IBM WebSphere eXtreme Scale
Version 8.6

Produktübersicht

Dezember 2012

8.6 Diese Ausgabe bezieht sich auf Version 8, Release 6 von WebSphere eXtreme Scale und alle nachfolgenden Releases und Modifikationen, bis dieser Hinweis in einer Neuausgabe geändert wird.

Diese Veröffentlichung ist eine Übersetzung des Handbuchs
IBM WebSphere eXtreme Scale Version 8.6 Product Overview,
herausgegeben von International Business Machines Corporation, USA

Informationen, die nur für bestimmte Länder Gültigkeit haben und für Deutschland, Österreich und die Schweiz nicht zutreffen, wurden in dieser Veröffentlichung im Originaltext übernommen.

Möglicherweise sind nicht alle in dieser Übersetzung aufgeführten Produkte in Deutschland angekündigt und verfügbar; vor Entscheidungen empfiehlt sich der Kontakt mit der zuständigen IBM Geschäftsstelle.

Änderung des Textes bleibt vorbehalten.

Herausgegeben von:

TSC Germany
Kst. 2877
Dezember 2012

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellen

Informationen zur *Produktübersicht*

Der Dokumentationssatz zu WebSphere eXtreme Scale umfasst drei Handbücher, die die erforderlichen Informationen zur Verwendung des Produkts WebSphere eXtreme Scale, zur Programmierung für das Produkt und zur Verwaltung des Produkts enthalten.

Bibliothek von WebSphere eXtreme Scale

Die Bibliothek von WebSphere eXtreme Scale enthält die folgenden Bücher:

- Die Veröffentlichung *Produktübersicht* enthält eine Übersicht über die Konzepte von WebSphere eXtreme Scale, einschließlich Anwendungsfallszenarien und Lernprogrammen.
- Im *Installationshandbuch* wird beschrieben, wie Sie allgemeine Topologien von WebSphere eXtreme Scale installieren.
- Die Veröffentlichung *Verwaltung* enthält die für Systemadministratoren erforderlichen Informationen, z. B. Planung von Anwendungsimplementierungen, Kapazitätsplanung, Installation und Konfiguration des Produkts, Starten und Stoppen von Servern, Überwachung der Umgebung und Sicherung der Umgebung.
- Die Veröffentlichung *Programmierung* enthält Informationen für Anwendungsentwickler zur Entwicklung von Anwendungen für WebSphere eXtreme Scale unter Verwendung der bereitgestellten API-Informationen.

Zum Herunterladen der Handbücher rufen Sie die Bibliotheksseite von WebSphere eXtreme Scale auf.

Dieselben Informationen wie in dieser Bibliothek finden Sie auch im Information Center von WebSphere eXtreme Scale Version 8.6. .

Veröffentlichungen offline verwenden

Alle Handbücher in der Bibliothek von WebSphere eXtreme Scale enthalten Links zum Information Center mit dem folgenden Stamm-URL: `http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r1m1` . Diese Links führen Sie direkt zu den zugehörigen Informationen. Wenn Sie jedoch offline arbeiten und auf einen dieser Links klicken, können Sie den Titel des Links in den anderen Veröffentlichungen in der Bibliothek suchen. Die API-Dokumentation, das Glossar und die Nachrichtenreferenzen sind in den PDF-Veröffentlichungen nicht verfügbar.

Zielgruppe

Dieses Buch ist für alle Benutzer bestimmt, die sich über das Produkt WebSphere eXtreme Scale informieren möchten.

Aktualisierungen für dieses Handbuch

Sie erhalten Aktualisierungen zu diesem Handbuch, indem Sie die jeweils aktuelle Version des Handbuchs von der Bibliotheksseite von WebSphere eXtreme Scale herunterladen.

Hinweise zu Rückmeldungen

Wenden Sie sich an das Dokumentationsteam. Haben Sie die benötigten Informationen gefunden? Sind die Informationen präzise und vollständig? Senden Sie Ihre Kommentare zu dieser Dokumentation per E-Mail an wasdoc@us.ibm.com.

Kapitel 1. Produktübersicht

WebSphere eXtreme Scale ist ein elastisches, skalierbares speicherinternes Datengrid. Das Datengrid übernimmt folgende Aufgaben: dynamische Zwischenspeicherung, Partitionierung, Replikation und Verwaltung von Anwendungsdaten und Geschäftslogik in mehreren Servern. WebSphere eXtreme Scale unterstützt die Verarbeitung hoher Transaktionsaufkommen mit hoher Effizienz und linearer Skalierbarkeit. Außerdem können Sie mit WebSphere eXtreme Scale Servicequalitäten wie Transaktionsintegrität, hohe Verfügbarkeit und voraussagbare Antwortzeiten erreichen.

Übersicht über WebSphere eXtreme Scale

WebSphere eXtreme Scale ist ein elastisches, skalierbares speicherinternes Datengrid. Das Datengrid übernimmt folgende Aufgaben: dynamische Zwischenspeicherung, Partitionierung, Replikation und Verwaltung von Anwendungsdaten und Geschäftslogik in mehreren Servern. WebSphere eXtreme Scale unterstützt die Verarbeitung hoher Transaktionsaufkommen mit hoher Effizienz und linearer Skalierbarkeit. Außerdem können Sie mit WebSphere eXtreme Scale Servicequalitäten wie Transaktionsintegrität, hohe Verfügbarkeit und voraussagbare Antwortzeiten erreichen.

WebSphere eXtreme Scale kann auf verschiedene Arten eingesetzt werden. Sie können das Produkt als äußerst leistungsfähigen Cache, als speicherinternen Datenbankverarbeitungsbereich für die Verwaltung von Anwendungsstatus oder zum Erstellen von XTP-Anwendungen (Extreme Transaction Processing) verwenden. Zur XTP-Funktionalität gehört eine Anwendungsinfrastruktur für die Unterstützung Ihrer anspruchsvollsten geschäftskritischen Anwendungen.

Elastische Skalierbarkeit

Eine elastische Skalierbarkeit wird durch die Verwendung eines verteilten Objekt-Cachings erreicht. Mit elastischer Skalierbarkeit überwacht und verwaltet das Datengrid sich selbst. Das Datengrid kann Server in der Topologie hinzufügen und entfernen und somit nach Bedarf die Speicher-, Netzdurchsatz- und Verarbeitungskapazität erhöhen und verringern. Beim Einleiten eines so genannten Scale-out-Prozesses wird dem aktiven Datengrid Kapazität hinzugefügt, ohne dass ein Neustart erforderlich ist. Beim so genannten Scale-in-Prozess werden Kapazitäten sofort entfernt. Außerdem ist das Datengrid selbst-heilend, d. h., es kann sich nach Fehlern automatisch wiederherstellen.

WebSphere eXtreme Scale und speicherinterne Datenbanken im Vergleich

WebSphere eXtreme Scale ist eigentlich keine speicherinterne Datenbank. Eine speicherinterne Datenbank ist zu einfach, um derartig komplexe Situationen zu verwalten, wie WebSphere eXtreme Scale es kann. Wenn eine speicherinterne Datenbank einen Server hat, der ausfällt, kann sie dieses Problem nicht beheben. Ein Ausfall kann katastrophal sein, wenn sich die gesamte Umgebung auf diesem einen Server befindet.

Zur Vermeidung dieser Art von Fehlern teilt eXtreme Scale die jeweilige Datenmenge in Partitionen auf, die Baumstrukturschemas mit Integritätsbedingungen

entsprechen. Baumstrukturschemas mit Integritätsbedingungen beschreiben die Beziehung zwischen Entitäten. Wenn Sie Partitionen verwenden, müssen die Entitätsbeziehungen eine Datenstruktur im Baumformat modellieren. In dieser Struktur ist der Kopf des Baums die Stammentität und die einzige Entität, die partitioniert wird. Alle untergeordneten Entitäten der Stammentität sind in derselben Partition wie die Stammentität gespeichert. Jede Partition ist als primäre Kopie oder Shard vorhanden. Eine Partition enthält auch Replikat-Shards für die Sicherung der Daten. Eine speicherinterne Datenbank kann diese Funktion nicht bereitstellen, weil sie diese Struktur und Dynamik nicht besitzt. Bei einer speicherinternen Datenbank müssen Sie die Operationen implementieren, die WebSphere eXtreme Scale automatisch ausführt. Sie können SQL-Operationen in speicherinternen Datenbanken ausführen und damit die Verarbeitungsgeschwindigkeit im Vergleich mit Datenbanken, die nicht im Speicher aufgeführt werden, verbessern. WebSphere eXtreme Scale hat anstelle der SQL-Unterstützung eine eigene Abfragesprache. Diese Abfragesprache ist elastischer, ermöglicht die Partitionierung von Daten und bietet eine zuverlässige Fehlerbehebung.

WebSphere eXtreme Scale mit Datenbanken

Mit dem Write-behind-Cache-Feature kann WebSphere eXtreme Scale als Front-End-Cache für eine Datenbank eingesetzt werden. Durch die Verwendung dieses Front-End-Caches können Sie den Durchsatz erhöhen und die Datenbanklast und Konkurrenzsituationen in der Datenbank verringern. WebSphere eXtreme Scale ermöglicht eine horizontale Skalierung einer Topologie (Scale-out und Scale-in) zu vorhersagbaren Verarbeitungskosten.

Die folgende Abbildung zeigt eine verteilte kohärente Cacheumgebung, in der die eXtreme-Scale-Clients Daten an das Datengrid senden und Daten aus dem Datengrid empfangen. Das Datengrid kann automatisch mit einem Back-End-Datenspeicher synchronisiert werden. Der Cache ist kohärent, weil alle Clients dieselben Daten im Cache sehen. Jede einzelne Information wird im Cache eines einzigen Servers gespeichert, auf den Schreibzugriff besteht. Durch die Verwendung einer einzigen Kopie jeder Information werden unnötige Datensatzkopien verhindert, die potenziell verschiedene Versionen der Daten enthalten könnten. Ein kohärenter Cache nimmt immer mehr Daten auf, je mehr Server dem Datengrid hinzugefügt werden. Die Skalierung des Caches erfolgt linear zum Wachstum des Datengrids. Die Daten können für zusätzliche Fehlertoleranz auch repliziert werden.

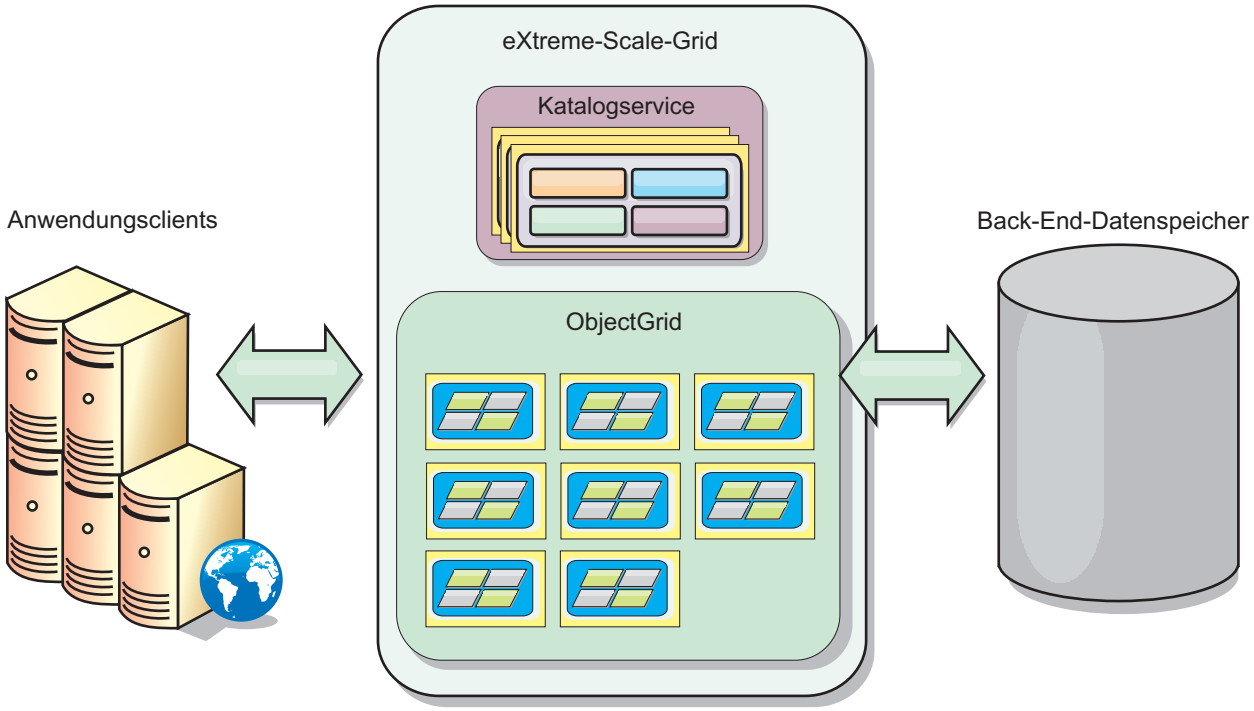


Abbildung 1. Übergeordnete Topologie

WebSphere eXtreme Scale hat Server, so genannte *Container-Server*, die das speicherinterne Datengrid bereitstellen. Die Server können in WebSphere Application Server oder in einfachen J2SE-JVMs ausgeführt werden. Es können mehrere Container-Server auf einem einzigen physischen Server ausgeführt werden. Deshalb kann das speicherinterne Datengrid groß sein. Das Datengrid wird weder durch den Hauptspeicher oder Adressraum der Anwendung bzw. des Anwendungsservers beschränkt noch hat es Einfluss auf diesen. Der Hauptspeicher kann die Summe mehrerer Hundert oder Tausend Java™ Virtual Machines sein, die auf vielen verschiedenen physischen Servern ausgeführt werden.

Als speicherinterner Datenbankverarbeitungsbereich kann WebSphere eXtreme Scale durch eine Platte, eine Datenbank oder beides gestützt werden.

Obwohl eXtreme Scale mehrere Java-APIs bereitstellt, erfordern die meisten Anwendungsfälle keine Benutzerprogrammierung, sondern müssen lediglich in der WebSphere-Infrastruktur konfiguriert und implementiert werden.

Übersicht über das Datengrid

Die einfachste eXtreme-Scale-Programmierschnittstelle ist die Schnittstelle "ObjectMap", die eine einfache Mapschnittstelle ist, die Folgendes bietet: eine Methode "map.put(key,value)", mit der ein Wert im Cache gespeichert werden kann, und eine Methode "map.get(key)", mit der der Wert später abgerufen werden kann.

Das grundlegende Datengridkonzept ist ein Schlüssel/Wert-Paar, in dem das Datengrid Werte (Java-Objekte) mit einem zugehörigen Schlüssel (einem weiteren Java-Objekt) speichert. Der Schlüssel wird später zum Abrufen des Werts verwendet. In eXtreme Scale setzt sich eine Map aus Einträgen solcher Schlüssel/Wert-Paare zusammen.

WebSphere eXtreme Scale bietet verschiedene Datengridkonfigurationen, angefangen von einem einzigen einfachen lokalen Cache bis hin zu einem großen verteilten Cache, mit mehreren JVMs und/oder Servern.

Zusätzlich zu einfachen Java-Objekten können Sie Objekte mit Beziehungen speichern. Zum Abrufen dieser Objekte können Sie eine Abfragesprache, die SQL gleicht, mit `SELECT-, FROM-, und WHERE`-Klauseln verwenden. Ein Bestellobjekt (Order) kann beispielsweise ein Kundenobjekt (Customer) und mehrere zugehörige Artikelobjekte (Item) haben. WebSphere eXtreme Scale unterstützt 1:1-, 1:n- und n:n-Beziehungen.

WebSphere eXtreme Scale unterstützt auch eine EntityManager-Programmierschnittstelle für das Speichern von Entitäten im Cache. Diese Programmierschnittstelle gleicht Entitäten in Java Enterprise Edition. Entitätsbeziehungen können automatisch über eine XML-Entitätsdeskriptordatei oder über Annotationen in Java-Klassen erkannt werden. Sie können eine Entität anhand des Primärschlüssels mit der Methode find in der Schnittstelle EntityManager aus dem Cache abrufen. Entitäten können innerhalb von Transaktionsgrenzen persistent im Datengrid festgeschrieben und aus diesem entfernt werden.

Stellen Sie sich eine verteilte Umgebung vor, in der der Schlüssel ein einfacher alphabetischer Name ist. Der Cache kann nach Schlüsseln in vier Partitionen aufgeteilt werden: Partition 1 für Schlüssel, die mit A-E beginnen, Partition 2 für Schlüssel, die mit F-L beginnen usw. Für die Verfügbarkeit besitzt eine Partition ein primäres Shard und ein Replikat-Shard. Änderungen an den Cachedaten werden am primären Shard vorgenommen und im Replikat-Shard repliziert. Sie konfigurieren die Anzahl der Server, die das Datengrid enthalten, und eXtreme Scale verteilt die Daten in Shards auf diese Serverinstanzen. Für die Verfügbarkeit werden Replikat-Shards auf anderen physischen Servern als die primären Shards gespeichert.

WebSphere eXtreme Scale verwendet einen Katalogservice, um das primäre Shard für jeden Schlüssel zu finden. Der Katalogserver verschiebt die Shards zwischen eXtreme-Scale-Servern, wenn die physischen Server ausfallen und später wiederhergestellt werden. Wenn beispielsweise der Server, der ein Replikat-Shards enthält, ausfällt, ordnet eXtreme Scale ein neues Replikat-Shard zu. Fällt ein Server aus, der ein primäres Shard enthält, wird das Replikat-Shard zum primären Shard hochgestuft. Wie zuvor wird ein neues Replikat-Shard erstellt.

Neuerungen in Version 8.6

WebSphere eXtreme Scale enthält zahlreiche neue Features in Version 8.6. Verwenden Sie diesen Abschnitt, um sich mit den neuesten Produktaktualisierungen vertraut zu machen.

Fortlaufende Abfrage

Wenn Sie Clientanwendungen entwickeln, die mit dem Datengrid interagieren, benötigen Sie möglicherweise Abfragen, die automatisch Echtzeitergebnisse abrufen, wenn neue Einträge eingefügt oder aktualisiert werden. Sie können die fortlaufende Abfrage verwenden, um in Ihrer Client-JVM (Java Virtual Machine) benachrichtigt zu werden, wenn Daten im Datengrid eingefügt oder aktualisiert werden. Dieses Feature vereinfacht das Grid- und Datenmanagement für Entwickler und/oder Administratoren. Lesen Sie den Abschnitt Weitere Informationen...

Plattenüberlauf

Sie können das Feature für Plattenüberlauf verwenden, um die Datengridkapazität zu erweitern, indem Cacheeinträge aus dem Hauptspeicher auf die Platte ausgelagert werden. Wenn Sie den Plattenüberlauf aktivieren, werden Einträge, die nicht mehr in den verfügbaren Hauptspeicher der Container-Server passen, auf der Platte gespeichert. Lesen Sie den Abschnitt Weitere Informationen...

Werte abgefragter Daten anzeigen

Sie können die Werte der Schlüssel in Datenabfragen, die Sie in der Überwachungskonsole oder mit dem Dienstprogramm **xscmd** erstellen, jetzt anzeigen. Lesen Sie den Abschnitt Weitere Informationen...

Unternehmensdatengrid

Unternehmensdatengrids verwenden den Transportmechanismus "eXtremeIO" und ein neues Serialisierungsformat. Mit dem neuen Transport und Serialisierungsformat können Sie Java- und .NET-Clients mit demselben Datengrid verbinden. Lesen Sie den Abschnitt Weitere Informationen...

Globaler Index

Der globale Index erweitert das integrierte HashIndex-Plug-in und wird auf Shards in einem verteilten, partitionierten Datengrid ausgeführt. Der globale Index überwacht die Position indexierter Attribute im Datengrid und bietet effiziente Möglichkeiten, um Partitionen, Schlüssel, Werte oder Einträge anhand von Attributen in großen, partitionierten Datengridumgebungen zu finden. Lesen Sie den Abschnitt Weitere Informationen...

Invalidierung mithilfe eines globalen Index

Sie können die Invalidierung mithilfe eines globalen Index aktivieren, um die Invalidierungseffizienz in einer großen partitionierten Umgebung (z. B. einer Umgebung mit mehr als 40 Partitionen) zu verbessern. Lesen Sie den Abschnitt Weitere Informationen...

High Performance Extensible Logging (HPEL)

Als Alternative zur Basisprotokoll- und -tracefunktion können Sie die Verwendung von HPEL in Katalog- und Container-Servern aktivieren. Lesen Sie den Abschnitt Weitere Informationen...

IBM® Support Assistant Data Collector

IBM Support Assistant Data Collector ist ein Tool, mit dem Sie Daten zur Problembestimmung aus Ihrer Umgebung von WebSphere eXtreme Scale erfassen können. Lesen Sie den Abschnitt Weitere Informationen...

Umgekehrter Bereichsindex

Sie können einen umgekehrten Bereichsindex mithilfe des integrierten InverseRangeIndex-Plug-ins konfigurieren. Lesen Sie den Abschnitt Weitere Informationen...

Liste der für die Verteilung inaktivierten Shard-Container

Wenn bei der Verteilung von Shards an einen bestimmten Shard-Container ein Problem auftritt, wird der Shard-Container in eine Liste aufgenommen, die verhindert, dass der Shard-Container weitere Verteilungsanforderungen enthält. Mit **xscmd**-Befehlen können Sie die inaktivierten Shard-Container auflisten und einen Shard-Container aus der Liste entfernen. Lesen Sie den Abschnitt Weitere Informationen...

Message Center

Das Message Center enthält eine zusammengefasste Ansicht von Ereignisbenachrichtigungen für Protokoll- und FFDC-Nachrichten (First-Failure Data Capture). Sie können diese Ereignisbenachrichtigungen mit dem Message Center in der Webkonsole, mit dem Dienstprogramm **xscmd** oder über das Programm mit MBeans anzeigen. Lesen Sie den Abschnitt Weitere Informationen...

Transaktionen, an denen mehrere Partitionen beteiligt sind

WebSphere eXtreme Scale Client unterstützt jetzt Transaktionsaktualisierungen auf mehreren Partitionen in einem Datengrid. Lesen Sie den Abschnitt Weitere Informationen...

Invalidierung des nahen Caches

Sie können die Invalidierung des nahen Caches konfigurieren, um veraltete Daten so schnell wie möglich aus dem nahen Cache zu entfernen. Wenn eine Aktualisierungs-, Lösch- oder Invalidierungsoperation für das ferne Datengrid ausgeführt wird, wird eine asynchrone Invalidierungsoperation im nahen Cache ausgelöst. Lesen Sie den Abschnitt Weitere Informationen...

WebSphere eXtreme Scale Client for .NET

Wenn Sie WebSphere eXtreme Scale Client for .NET installieren, können Sie .NET-Anwendungen implementieren, die auf das Datengrid zugreifen. Lesen Sie den Abschnitt Weitere Informationen...

Neue Methoden und APIs

- Methode upsert: Die Methoden upsert und upsertAll ersetzen Sie die ObjectMap-Methoden put und putAll. Lesen Sie den Abschnitt Weitere Informationen...
- Methode lock: Wenn Sie pessimistisches Sperren verwenden, können Sie die Sperrmethode verwenden, um Daten oder Schlüssel zu sperren, ohne dass Datenwerte zurückgegeben werden. Mit der Sperrmethode können Sie den Schlüssel im Grid oder den Schlüssel sperren und feststellen, ob der Wert im Grid vorhanden ist. In früheren Releases haben Sie die APIs "get" und "getForUpdate" zum Sperren von Schlüsseln im Datengrid verwendet. Lesen Sie den Abschnitt Weitere Informationen...
- sessionIdOverrideClass: Diese Klasse implementiert die Schnittstelle "com.ibm-.websphere.xs.sessionmanager.SessionIDOverride", um die von der Methode "HttpSession.getId()" abgerufene Standardbenutzer-ID zu überschreiben. Lesen Sie den Abschnitt Weitere Informationen...

Neue Befehle und Parameter für das Dienstprogramm `xscmd`

- Befehl **`xscmd -c getNotificationFilter`**: Führen Sie diesen Befehl aus, um die aktuellen Filter für neue Benachrichtigungen aus dem Message Center anzuzeigen. Lesen Sie den Abschnitt Weitere Informationen...
- Befehl **`xscmd -c listenForNotifications`**: Führen Sie diesen Befehl aus, um auf neue Benachrichtigungen aus dem Message Center zu warten. Lesen Sie den Abschnitt Weitere Informationen...
- Befehl **`xscmd -c setNotificationFilter`**: Führen Sie diesen Befehl aus, um einen Filter für neue Benachrichtigungen aus dem Message Center zu erstellen. Lesen Sie den Abschnitt Weitere Informationen...
- Befehl **`xscmd -c showLinkedDomains`**: Führen Sie diesen Befehl aus, um zu prüfen, welche Katalogservicedomänen mit Ihrer lokalen Katalogservicedomäne verlinkt sind. Lesen Sie den Abschnitt Weitere Informationen...
- Befehl **`xscmd -c showNotificationHistory`**: Führen Sie diesen Befehl aus, um die Ausgabe des Ereignisbenachrichtigungsprotokolls in tabellarischer Form anzuzeigen. Lesen Sie den Abschnitt Weitere Informationen...
- Parameter **`-to`** oder **`--timeout`**: Geben Sie diesen Parameter an, um das Zeitlimit zu reduzieren, um zu verhindern, dass während eines Netzbrownouts oder Systemfehlers System- oder andere Netzzeitlimits abgewartet werden. Lesen Sie den Abschnitt Weitere Informationen...
- Parameter **`-hc`** oder **`--linkHealthCheck`**: Verwenden Sie diesen Parameter mit dem Befehl **`xscmd -c showLinkedPrimaries`**, um zu überprüfen, ob die primären Shards die richtige Anzahl an Katalogservicedomänenlinks haben. Lesen Sie den Abschnitt Weitere Informationen...
- Befehl **`xscmd -c listDisabledForPlacement`**: Führen Sie diesen Befehl aus, um eine Liste mit Shard-Containern anzuzeigen, die für die Shard-Verteilung inaktiviert wurden. Lesen Sie den Abschnitt Weitere Informationen...
- Befehl **`xscmd -c listIndoubts`**: Führen Sie diesen Befehl aus, um die Liste unbestätigter Transaktionen anzuzeigen. Führen Sie diesen Befehl aus, um potenzielle Ausnahmen wegen Überschreitung der Sperrzeit auf einer Partition zu beheben. Lesen Sie den Abschnitt Weitere Informationen...
- Befehl **`xscmd -c enableForPlacement -ct <Shard-Container>`**: Führen Sie diesen Befehl aus, um einen Shard-Container zu reaktivieren, der für die Shardverteilung inaktiviert wurde. Lesen Sie den Abschnitt Weitere Informationen...
- Befehle **`xscmd -c showReplicationState`** und **`xscmd -c showDomainReplicationState`**: Führen Sie diese Befehle aus, um den Status der Revisionen in Ihren Katalogservern oder Katalogserverdomänen anzuzeigen. Lesen Sie den Abschnitt Weitere Informationen...
- Befehl **`xscmd -c showTransport`**: Führen Sie diesen Befehl aus, um den Transporttyp der Katalogservicedomäne anzuzeigen. Lesen Sie den Abschnitt Weitere Informationen...

Ferne Protokollierung

Sie können die ferne Protokollierung aktivieren, um Protokolleinträge auf einem fernen Server zu speichern. Es muss ein syslog-Server für die Überwachung und Erfassung von Ereignissen verfügbar sein. Lesen Sie den Abschnitt Weitere Informationen...

Unterstützung des REST-Gateways

Sie können das REST-Gateway (Representational State Transfer) verwenden, um auf einfache Datengrids zuzugreifen, die in einem Verbund enthalten sind. Dieses

REST-Gateway ist hilfreich, wenn Sie über Nicht-Java-Umgebungen auf Griddaten zugreifen müssen. Lesen Sie den Abschnitt Weitere Informationen...

Unterstützung für die Spezifikation Java Servlets 3.0

Die HTTP-Sitzungsverwaltungsfunktion von WebSphere eXtreme Scale unterstützt jetzt die Spezifikation Java Servlets 3.0. Wenn Sie Anwendungen für WebSphere eXtreme Scale in einer eigenständigen Umgebung schreiben, werden nur Rückrufe an Listener abgesetzt, die explizit in der Datei `web.xml` angegeben sind, wenn Sitzungen über die ferne Containerbereinigung von WebSphere eXtreme Scale invalidiert werden.

eXtreme Data Format (XDF)

XDF basiert auf dem MapSerializerPlugin-Plug-in und ist jetzt die Standardserialisierungstechnologie, die verwendet wird, wenn Sie IBM eXtremeIO (XIO) ausführen und der Kopiermodus für Ihre Maps auf `COPY_TO_BYTES` gesetzt ist. Wenn Sie dieses Feature aktivieren, können Java- und C#-Objekte Daten in demselben Datengrid gemeinsam nutzen. Lesen Sie den Abschnitt Weitere Informationen...

Feature "webApp"

Das Feature "webApp" von Liberty-Profil enthält die Funktion zum Erweitern der Webanwendung des Liberty-Profils. Fügen Sie das Feature "webApp" hinzu, wenn Sie HTTP-Sitzungsdaten für die Fehlertoleranz replizieren möchten. Lesen Sie den Abschnitt Weitere Informationen...

Feature "WebGrid"

Mit diesem Feature des Liberty-Profils kann ein Liberty-Profilserver ein Datengrid hosten, in dem Daten für Anwendungen zwischengespeichert werden, um HTTP-Sitzungsdaten für Fehlertoleranz zu replizieren. Lesen Sie den Abschnitt Weitere Informationen...

Releaseinformationen

Hier finden Sie Links zur Unterstützungswebsite für das Produkt, zur Produktdokumentation und zum letzten Stand der Updates, Einschränkungen und bekannten Problemen für das Produkt.

- „Zugriff auf den letzten Stand von Updates, Einschränkungen und bekannten Problemen"
- „Zugriff auf System- und Softwarevoraussetzungen" auf Seite 9
- „Zugriff auf die Produktdokumentation " auf Seite 9
- „Zugriff auf die Unterstützungswebsite für das Produkt " auf Seite 9
- „Kontaktaufnahme mit der IBM Softwareunterstützung " auf Seite 9

Zugriff auf den letzten Stand von Updates, Einschränkungen und bekannten Problemen

Die Releaseinformationen sind auf der Produktunterstützungssite als technische Hinweise verfügbar. Eine Liste aller technischen Hinweise für WebSphere eXtreme Scale finden Sie auf der Unterstützungswebseite. Wenn Sie auf die hier bereitgestellten Links klicken, wird die Unterstützungwebseite nach den relevanten Releaseinformationen durchsucht, die dann als Liste zurückgegeben werden.

- Eine Liste der Releaseinformationen für Version 8.6 finden Sie auf der Unterstützungswebseite.

Zugriff auf System- und Softwarevoraussetzungen

Die Hardware- und Softwarevoraussetzungen finden Sie auf den folgenden Webseiten:

- Detailed system requirements

Zugriff auf die Produktdokumentation

Das vollständige Informationsset finden Sie auf der Bibliotheksseite.

Zugriff auf die Unterstützungswebsite für das Produkt

Wenn Sie nach den neuesten technischen Hinweisen, Downloads und Korrekturen sowie nach Informationen zur Unterstützung suchen, rufen Sie das Unterstützungsportal auf.

Kontaktaufnahme mit der IBM Softwareunterstützung

Wenn ein Problem mit dem Produkt auftritt, versuchen Sie zuerst, die folgenden Aktionen auszuführen:

- Führen Sie die in der Produktdokumentation beschriebenen Schritte aus.
- Suchen Sie in der Onlinehilfe nach Referenzliteratur.
- Schlagen Sie die Fehlernachrichten in der Nachrichtenreferenz nach.

Sollten Sie das Problem nicht auf diese Weise lösen können, wenden Sie sich an die technische Unterstützung der IBM.

Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden.

Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Dienstleistungen von IBM verwendet werden können. Anstelle der IBM Produkte, Programme oder Services können auch andere, ihnen äquivalente Produkte, Programme oder Services verwendet werden, solange diese keine gewerblichen oder anderen Schutzrechte von IBM verletzen. Die Verantwortung für den Betrieb von Produkten, Programmen und Services anderer Anbieter liegt beim Kunden.

Für in diesem Handbuch beschriebene Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieser Dokumentation ist keine Lizenzierung dieser Patente verbunden. Lizenzanforderungen sind schriftlich an folgende Adresse zu richten (Anfragen an diese Adresse müssen auf Englisch formuliert werden):

IBM Director of Licensing
IBM Europe, Middle East & Africa
Tour Descartes

2, avenue Gambetta
92066 Paris La Defense
France

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die hier enthaltenen Informationen werden in regelmäßigen Zeitabständen aktualisiert und als Neuausgabe veröffentlicht. IBM kann ohne weitere Mitteilung jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Verweise in diesen Informationen auf Websites anderer Anbieter dienen lediglich als Benutzerinformationen und stellen keinerlei Billigung des Inhalts dieser Websites dar. Das über diese Websites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt. Die Verwendung dieser Websites geschieht auf eigene Verantwortung.

Werden an IBM Informationen eingesandt, können diese beliebig verwendet werden, ohne dass eine Verpflichtung gegenüber dem Einsender entsteht.

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängig voneinander erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

IBM Corporation
Mail Station P300
522 South Road
Poughkeepsie, NY 12601-5400
USA
Attention: Information Requests

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Lieferung des im Dokument aufgeführten Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt auf der Basis der IBM Rahmenvereinbarung bzw. der Allgemeinen Geschäftsbedingungen von IBM, der IBM Internationalen Nutzungsbedingungen für Programmpakete oder einer äquivalenten Vereinbarung.

Alle in diesem Dokument enthaltenen Leistungsdaten stammen aus einer kontrollierten Umgebung. Die Ergebnisse, die in anderen Betriebsumgebungen erzielt werden, können daher erheblich von den hier erzielten Ergebnissen abweichen. Einige Daten stammen möglicherweise von Systemen, deren Entwicklung noch nicht abgeschlossen ist. Eine Gewährleistung, dass diese Daten auch in allgemein verfügbaren Systemen erzielt werden, kann nicht gegeben werden. Darüber hinaus wurden einige Daten unter Umständen durch Extrapolation berechnet. Die tatsächlichen Ergebnisse können davon abweichen. Benutzer dieses Dokuments sollten die entsprechenden Daten in ihrer spezifischen Umgebung prüfen.

Alle Informationen zu Produkten anderer Anbieter stammen von den Anbietern der aufgeführten Produkte, deren veröffentlichten Ankündigungen oder anderen allgemein verfügbaren Quellen. IBM hat diese Produkte nicht getestet und kann

daher keine Aussagen zu Leistung, Kompatibilität oder anderen Merkmalen machen. Fragen zu den Leistungseigenschaftenn von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Aussagen über Pläne und Absichten von IBM unterliegen Änderungen oder können zurückgenommen werden und repräsentieren nur die Ziele von IBM.

Alle von IBM angegebenen Preise sind empfohlene Richtpreise und können jederzeit ohne weitere Mitteilung geändert werden. Händlerpreise können u. U. von den hier genannten Preisen abweichen.

Diese Veröffentlichung dient nur zu Planungszwecken. Die in dieser Veröffentlichung enthaltenen Informationen können geändert werden, bevor die beschriebenen Produkte verfügbar sind.

Diese Veröffentlichung enthält Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufes. Sie sollen nur die Funktionen des Lizenzprogramms illustrieren und können Namen von Personen, Firmen, Marken oder Produkten enthalten. Alle diese Namen sind frei erfunden; Ähnlichkeiten mit tatsächlichen Namen und Adressen sind rein zufällig.

COPYRIGHTLIZENZ:

Diese Veröffentlichung enthält Beispielanwendungsprogramme, die in Quellensprache geschrieben sind und Programmiertechniken in verschiedenen Betriebsumgebungen veranschaulichen. Sie dürfen diese Beispielprogramme kostenlos kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, zu verwenden, zu vermarkten oder zu verteilen, die mit der Anwendungsprogrammierschnittstelle für die Betriebsumgebung konform sind, für die diese Beispielprogramme geschrieben werden. Diese Beispiele wurden nicht unter allen denkbaren Bedingungen getestet. Daher kann IBM die Zuverlässigkeit, Wartungsfreundlichkeit oder Funktion dieser Programme weder zusagen noch gewährleisten. Die Beispielprogramme werden ohne Wartung (auf "as-is"-Basis) und ohne jegliche Gewährleistung zur Verfügung gestellt. IBM übernimmt keine Haftung für Schäden, die durch die Verwendung der Beispielprogramme entstehen.

Kopien oder Teile der Beispielprogramme bzw. daraus abgeleiteter Code müssen folgenden Copyrightvermerk beinhalten:

© (Name Ihrer Firma) (Jahr). Teile des vorliegenden Codes wurden aus Beispielprogrammen der IBM Corp. abgeleitet.

© Copyright IBM Corp. _Jahr/Jahre angeben_. Alle Rechte vorbehalten.

Informationen zu Programmierschnittstellen

In der vorliegenden Veröffentlichung werden in erster Linie Information en dokumentiert, die NICHT als Programmierschnittstellen von WebSphere eXtreme Scale verwendet dürfen. Außerdem sind in dieser Veröffentlichung vorgesehene Programmierschnittschnellen beschrieben, die es dem Kunden ermöglichen, Programme zu schreiben, mit denen die Services von WebSphere eXtreme Scale abgerufen werden können. Diese Informationen sind an den entsprechenden Stellen mit einer einführenden Anweisung zu einem Kapitel oder Abschnitt oder mit der folgenden Markierung gekennzeichnet: Informationen zu Programmierschnittstellen.

Marken

IBM, das IBM Logo und ibm.com sind Marken oder eingetragene Marken der IBM Corporation in den USA und/oder anderen Ländern. Weitere Produkt- und Servicenamen können Marken von IBM oder anderen Unternehmen sein. Eine aktuelle Liste der IBM Marken finden Sie auf der Webseite „Copyright and trademark information" unter www.ibm.com/legal/copytrade.shtml.

Bedingungen für Information Center

Die Berechtigung zur Nutzung dieser Veröffentlichungen wird Ihnen auf der Basis der folgenden Bedingungen gewährt.

Anwendbarkeit

Die Bedingungen gelten zusätzlich zu allen Nutzungsbedingungen für die IBM Website.

Persönliche Nutzung

Sie dürfen diese Veröffentlichungen für Ihre persönliche, nicht kommerzielle Nutzung unter der Voraussetzung vervielfältigen, dass alle Eigentumsvermerke erhalten bleiben. Sie dürfen diese Veröffentlichungen oder Teile der Veröffentlichungen ohne ausdrückliche Genehmigung von IBM nicht weitergeben, anzeigen oder abgeleitete Werke davon erstellen.

Kommerzielle Nutzung

Sie dürfen diese Veröffentlichungen nur innerhalb Ihres Unternehmens und unter der Voraussetzung, dass alle Eigentumsvermerke erhalten bleiben, vervielfältigen, weitergeben und anzeigen. Sie dürfen diese Veröffentlichungen oder Teile der Veröffentlichungen ohne ausdrückliche Genehmigung von IBM außerhalb Ihres Unternehmens nicht vervielfältigen, weitergeben, anzeigen oder abgeleitete Werke davon erstellen.

Rechte

Abgesehen von den hier gewährten Berechtigungen erhalten Sie keine weiteren Berechtigungen, Lizenzen oder Rechte (veröffentlicht oder stillschweigend) in Bezug auf die Veröffentlichungen oder darin enthaltene Informationen, Daten, Software oder geistiges Eigentum.

IBM behält sich das Recht vor, die in diesem Dokument gewährten Berechtigungen nach eigenem Ermessen zurückzuziehen, wenn sich die Nutzung der Veröffentlichungen für IBM als nachteilig erweist oder wenn die obigen Nutzungsbestimmungen nicht genau befolgt werden.

Sie dürfen diese Informationen nur in Übereinstimmung mit allen anwendbaren Gesetzen und Vorschriften, einschließlich aller US-amerikanischen Exportgesetze und Verordnungen, herunterladen und exportieren.

IBM übernimmt keine Gewährleistung für den Inhalt dieser Veröffentlichungen. Diese Veröffentlichungen werden auf der Grundlage des gegenwärtigen Zustands (auf "as-is"-Basis) und ohne eine ausdrückliche oder stillschweigende Gewährleistung für die Handelsüblichkeit, die Verwendungsfähigkeit für einen bestimmten Zweck oder die Freiheit von Rechten Dritter zur Verfügung gestellt.

IBM Marken

IBM, das IBM Logo und ibm.com sind Marken oder eingetragene Marken der IBM Corporation in den USA und/oder anderen Ländern. Weitere Produkt- und Servicenamen können Marken von IBM oder anderen Unternehmen sein. Eine aktuelle Liste der IBM Marken finden Sie auf der Webseite www.ibm.com/legal/copytrade.shtml.

Hardware- und Softwarevoraussetzungen

Dieser Abschnitt enthält eine Übersicht über die Hardware- und Betriebssystemvoraussetzungen. Sie müssen zwar keine bestimmte Version der Hardware oder des Betriebssystems für WebSphere eXtreme Scale verwenden, aber auf der Seite "Systems Requirements" der Produktunterstützungssite sind die formal unterstützten Hardware- und Softwareoptionen beschrieben. Sollte die Informationen im Information Center und auf der Seite "System Requirements" widersprüchlich sein, haben die Informationen auf der Website Vorrang. Die Informationen zu den Voraussetzungen im Information Center werden nur im Hinblick auf die Bedienerfreundlichkeit bereitgestellt.

Die offizielle Beschreibung der Hardware- und Softwarevoraussetzungen finden Sie auf der Webseite System Requirements.

Sie können das Produkt in Java-EE- und in Java-SE-Umgebungen installieren und implementieren. Außerdem können Sie die Clientkomponente direkt, ohne Integration mit WebSphere Application Server, mit JEE-Anwendungen bündeln.

Hardwarevoraussetzungen

WebSphere eXtreme Scale setzt keine bestimmte Hardwareversion voraus. Die Hardwarevoraussetzungen richten sich nach der unterstützten Hardware für die Installation der Java Platform, Standard Edition, die Sie für die Ausführung von WebSphere eXtreme Scale verwenden. Wenn Sie eXtreme Scale mit WebSphere Application Server oder einer anderen Java-EE-Implementierung (Java Platform, Enterprise Edition) verwenden, sind die Hardwarevoraussetzungen dieser Plattformen für WebSphere eXtreme Scale ausreichend.

Betriebssystemvoraussetzungen

.NET **8.6+** Einzelheiten zu den Voraussetzungen für eine .NET-Clientumgebung finden Sie unter Hinweise zu Microsoft .NET.

Java Jede Java-SE- und jede Java-EE-Implementierung setzt verschiedene Betriebssystemversionen oder -Fixes für Probleme voraus, die während des Testens der Java-Implementierung erkannt werden. Die von diesen Implementierungen vorausgesetzten Versionen sind für eXtreme Scale ausreichend.

Voraussetzungen von Installation Manager

Bevor Sie WebSphere eXtreme Scale installieren können, müssen Sie Installation Manager installieren. Sie können Installation Manager vom Produktdatenträger, über eine Datei, die Sie von der Website von Passport Advantage herunterladen, oder über eine Datei installieren, die die aktuellste Version von Installation Manager von der Download-Site von IBM Installation Manager enthält. Weitere Informa-

tionen finden Sie unter IBM Installation Manager und Produktangebote von WebSphere eXtreme Scale installieren.

Web-Browser-Voraussetzungen

Die Webkonsole unterstützt die folgenden Web-Browser:

- Mozilla Firefox Version 3.5.x und höher
- Microsoft Internet Explorer Version 7 und höher

Voraussetzungen in Bezug auf WebSphere Application Server

8.6+

- WebSphere Application Server Version 7.0.0.21 oder höher
- WebSphere Application Server Version 8.0.0.2 oder höher

Weitere Informationen finden Sie auf der Webseite mit den empfohlenen Fixes für WebSphere Application Server.

Java-Voraussetzungen

8.6+ Andere Java-EE-Implementierungen können die Laufzeitumgebung von eXtreme Scale als lokale Instanz oder als Client für Server von eXtreme Scale verwenden. Zum Implementieren von Java SE müssen Sie Version 6 oder höher verwenden.

Verzeichniskonventionen

Die folgenden Verzeichniskonventionen werden in der Dokumentation verwendet, um auf spezielle Verzeichnisse zu verweisen, wie z.B. *WXS-Installationsstammverzeichnis* und *WXS-Ausgangsverzeichnis*. Sie greifen in verschiedenen Szenarien, wie z. B. während der Installation oder der Verwendung der Befehlszeilenprogramme, auf diese Verzeichnisse zu.

WXS-Installationsstammverzeichnis

Das Verzeichnis *WXS-Installationsstammverzeichnis* ist das Stammverzeichnis, in dem die Produktdateien von WebSphere eXtreme Scale installiert sind. Das Verzeichnis *WXS-Installationsstammverzeichnis* kann das Verzeichnis sein, in dem das Testarchiv entpackt wurde, oder das Verzeichnis, in dem das Produkt WebSphere eXtreme Scale installiert ist.

- Beispiel für die entpackte Testversion:

  Beispiel: `/opt/IBM/WebSphere/eXtremeScale`

- Beispiel, wenn WebSphere eXtreme Scale in einem eigenständigen Verzeichnis installiert ist:

  UNIX **Beispiel:** `/opt/IBM/eXtremeScale`

  Windows **Beispiel:** `C:\Program Files\IBM\WebSphere\eXtremeScale`

- Beispiel, wenn WebSphere eXtreme Scale mit WebSphere Application Server integriert ist:

  Beispiel: `/opt/IBM/WebSphere/AppServer`

WXS-Ausgangsverzeichnis

Das Verzeichnis *WXS-Ausgangsverzeichnis* ist das Stammverzeichnis der Produktbibliotheken, Beispiele und Komponenten von WebSphere eXtreme Scale. Dieses Verzeichnis entspricht dem Verzeichnis *WXS-Installationsstammverzeichnis*, wenn die Testversion entpackt wurde. Bei ei-

genständigen Installationen ist das Verzeichnis *WXS-Ausgangsverzeichnis* das Unterverzeichnis `ObjectGrid` im Verzeichnis *WXS-Installationsstammverzeichnis*. Bei Installationen, die mit WebSphere Application Server integriert sind, ist dieses Verzeichnis das Verzeichnis `optionalLibraries/ObjectGrid` im Verzeichnis *WXS-Installationsstammverzeichnis*.

- Beispiel für die entpackte Testversion:

  Beispiel: `/opt/IBM/WebSphere/eXtremeScale`
- Beispiel, wenn WebSphere eXtreme Scale in einem eigenständigen Verzeichnis installiert ist:

  UNIX **Beispiel:** `/opt/IBM/eXtremeScale/ObjectGrid`

  Windows **Beispiel:** *WXS-Installationsstammverzeichnis*`\ObjectGrid`
- Beispiel, wenn WebSphere eXtreme Scale mit WebSphere Application Server integriert ist:

  Beispiel: `/opt/IBM/WebSphere/AppServer/optionalLibraries/ObjectGrid`

WAS-Stammverzeichnis

Das Verzeichnis *WAS-Stammverzeichnis* ist das Stammverzeichnis einer Installation von WebSphere Application Server:

Beispiel: `/opt/IBM/WebSphere/AppServer`

.NET 8.6+ **Ausgangsverzeichnis_des_net-Clients**

Das Verzeichnis *Ausgangsverzeichnis_des_net-Clients* ist das Stammverzeichnis einer .NET-Clientinstallation.

Beispiel: `C:\Program Files\IBM\WebSphere\eXtreme Scale .NET Client`

Ausgangsverzeichnis_des_REST-Service

Das Verzeichnis *Ausgangsverzeichnis_des_REST-Service* ist das Verzeichnis, in dem die Bibliotheken und Beispiele des REST-Datenservice von WebSphere eXtreme Scale enthalten sind. Dieses Verzeichnis hat den Namen `restservice` und ist ein Unterverzeichnis im Verzeichnis *WXS-Ausgangsverzeichnis*.

- Beispiel für eigenständige Implementierungen:

  Beispiel: `/opt/IBM/WebSphere/eXtremeScale/ObjectGrid/restservice`

  Beispiel: *WXS-Ausgangsverzeichnis*`\restservice`
- Beispiel für integrierte Implementierungen mit WebSphere Application Server:

  Beispiel: `/opt/IBM/WebSphere/AppServer/optionalLibraries/ObjectGrid/restservice`

Tomcat-Stammverzeichnis

Das *Tomcat-Stammverzeichnis* ist das Stammverzeichnis der Apache-Tomcat-Installation.

Beispiel: `/opt/tomcat5.5`

WASCE-Stammverzeichnis

Das *WASCE-Stammverzeichnis* ist das Stammverzeichnis der Installation von WebSphere Application Server Community Edition.

Beispiel: `/opt/IBM/WebSphere/AppServerCE`

Java-Ausgangsverzeichnis

Das *Java-Ausgangsverzeichnis* ist das Stammverzeichnis der Installation von Java Runtime Environment (JRE).

UNIX **Beispiel:** `/opt/IBM/WebSphere/eXtremeScale/java`

Windows **Beispiel:** *WXS-Installationsstammverzeichnis*`\java`

Ausgangsverzeichnis_für_Beispiele
Das *Ausgangsverzeichnis_für_Beispiele* ist das Verzeichnis, in dem Sie die Beispieldateien entpacken, die für die Lernprogramme verwendet werden.

UNIX **Beispiel:** *WXS-Ausgangsverzeichnis*`/samples`

Windows **Beispiel:** *WXS-Ausgangsverzeichnis*`\samples`

DVD-Stammverzeichnis
Das Verzeichnis *DVD-Stammverzeichnis* ist das Stammverzeichnis der DVD, die das Produkt enthält.

Beispiel: `dvd_root/docs/`

Equinox-Stammverzeichnis
Das Verzeichnis *Equinox-Stammverzeichnis* ist das Stammverzeichnis der Eclipse-Equinox-OSGi-Framework-Installation.

Beispiel:`/opt/equinox`

Benutzerausgangsverzeichnis
Das Verzeichnis *Benutzerausgangsverzeichnis* ist die Position, an der Benutzerdateien gespeichert werden, wie z. B. Sicherheitsprofile.

Windows `c:\Documents and Settings\`*Benutzername*

UNIX `/home/`*Benutzername*

Technische Übersicht über WebSphere eXtreme Scale

WebSphere eXtreme Scale ist ein elastisches, skalierbares speicherinternes Datengrid. Das Produkt übernimmt folgende Aufgaben: dynamische Zwischenspeicherung, Partitionierung, Replikation und Verwaltung von Anwendungsdaten und Geschäftslogik in mehreren Servern.

Da WebSphere eXtreme Scale keine speicherinterne Datenbank ist, müssen Sie spezielle Konfigurationsanforderungen berücksichtigen. Der erste Schritt bei der Implementierung eines Datengrids ist das Starten einer Stammgruppe und eines Katalogservice, der als Koordinator für alle anderen Java Virtual Machines auftritt, die am Datengrid teilnehmen und Konfigurationsdaten verwalten. Die Prozesse von WebSphere eXtreme Scale werden mit einfachen Befehlsscriptaufrufen über die Befehlszeile gestartet.

Der nächste Schritt ist das Starten der Serverprozesse von WebSphere eXtreme Scale für das Datengrid, um Daten zu speichern und abzurufen. Wenn Server gestartet werden, registrieren sie sich automatisch bei der Stammgruppe und beim Katalogservice, was ihnen eine Zusammenarbeit bei der Bereitstellung der Datengrid-Services ermöglicht. Je mehr Server gestartet werden, desto höher sind Kapazität und Zuverlässigkeit des Datengrids.

Ein lokales Datengrid ist ein einfaches Grid mit einer Instanz, d. h., alle Daten werden in einem einzigen Grid gespeichert. Wenn Sie WebSphere eXtreme Scale effektiv als speicherinternen Datenbankverarbeitungsbereich nutzen möchten, können Sie ein verteiltes Datengrid konfigurieren und implementieren. Die Daten im verteilten Grid werden so auf die verschiedenen eXtreme-Scale-Server verteilt, dass jeder Server einen Teil der Daten, eine so genannte Partition, enthält.

Ein wichtiger Konfigurationsparameter für verteilte Datengrids ist die Anzahl der Gridpartitionen. Die Griddaten werden in diese Anzahl von Untergruppen partitioniert, und jede dieser Untergruppen wird als Partition bezeichnet. Der Katalogservice sucht die Partition für eine bestimmte Information anhand des Schlüssels. Die Anzahl der Partitionen wirkt sich direkt auf die Kapazität und die Skalierbarkeit des Datengrids aus. Ein Server kann eine oder mehrere Datengridpartitionen enthalten. Somit ist die Größe einer Partition auf die Hauptspeicherkapazität des Servers beschränkt. Mit zunehmender Anzahl an Partitionen erhöht sich die Kapazität des Datengrids. Die maximale Kapazität eines Datengrids entspricht der Anzahl an Partitionen, multipliziert mit der Größe des verfügbaren Speichers jedes Servers. Ein Server kann eine JVM sein, aber Sie können Ihren eXtreme-Scale-Server so definieren, dass er für Ihre Implementierungsumgebung geeignet ist.

Die Daten einer Partition werden in einem Shard gespeichert. Für die Verfügbarkeit kann ein Datengrid mit Replikaten konfiguriert werden, die synchron oder asynchron sein können. Änderungen an den Griddaten werden im primären Shard vorgenommen und in den Replikat-Shards repliziert. Der von einem Datengrid belegte bzw. benötigte Gesamtspeicher kann mit Hilfe der folgenden Formel ermittelt werden: Größe des Datengrids mal (1 (für das primäre Shard) + Anzahl der Replikate).

WebSphere eXtreme Scale verteilt die Shards eines Datengrids auf die Server, aus denen sich das Grid zusammensetzt. Diese Server können sich auf demselben physischen Server oder auf unterschiedlichen physischen Servern befinden. Für die Verfügbarkeit werden Replikat-Shards auf anderen physischen Servern als die primären Shards gespeichert.

WebSphere eXtreme Scale überwacht den Status der zugehörigen Server und verschiebt Shards bei einem Ausfall und der anschließenden Wiederherstellung von Shards oder physischen Servern. Wenn beispielsweise der Server, der ein Replikat-Shard enthält, ausfällt, ordnet WebSphere eXtreme Scale ein neues Replikat-Shard zu und repliziert Daten vom primären Shard auf dem neuen Replikat. Wenn ein Server, der ein primäres Shard enthält, ausfällt, wird das Replikat-Shard als primäres Shard hochgestuft, und es wird ein neues Replikat-Shard erstellt. Beim Start eines weiteren Servers für das Datengrid, werden die Shards gleichmäßig auf alle Server verteilt. Die Neuverteilung wird als horizontale Vorwärtsskalierung (Scale-out) bezeichnet. Für eine horizontale Rückskalierung (Scale-in) können Sie einen der Server stoppen, um die von einem Datengrid konsumierten Ressourcen zu verringern. Daraufhin werden die Shards auf die verbleibenden Server verteilt.

Übersicht über Caching

WebSphere eXtreme Scale kann als speicherinterner Datenbankverarbeitungsbereich eingesetzt werden, den Sie verwenden können, um integriertes Caching für ein Datenbank-Back-End oder um einen Nebencache bereitzustellen. Beim integrierten Caching wird eXtreme Scale als primäres Mittel für die Interaktion mit den Daten verwendet. Wenn eXtreme Scale als Nebencache verwendet wird, wird das Back-End zusammen mit dem Datengrid verwendet. In diesem Abschnitt sind die verschiedenen Cachekonzepte und -szenarien sowie die verfügbaren Topologien für die Implementierung eines Datengrids beschrieben.

Caching-Architektur: Maps, Container, Clients und Kataloge

Mit WebSphere eXtreme Scale kann Ihre Architektur speicherinternes Daten-Caching oder verteiltes Client/Server-Daten-Caching verwenden.

WebSphere eXtreme Scale erfordert für den Betrieb eine minimale zusätzliche Infrastruktur. Die Infrastruktur setzt sich aus Scripts für das Installieren, Starten und Stoppen einer Java-EE-Anwendung auf einem Server zusammen. Die zwischengespeicherten Daten werden im Server von eXtreme Scale gespeichert, und Clients stellen über Fernzugriff eine Verbindung zum Server her.

Verteilte Caches bieten eine bessere Leistung, Verfügbarkeit und Skalierbarkeit und können unter Verwendung dynamischer Topologien konfiguriert werden, in denen Server automatisch verteilt werden. Zusätzliche Server können hinzugefügt werden, ohne die vorhandenen eXtreme Scale-Server erneut starten zu müssen. Sie können einfache Implementierungen erstellen oder große Implementierungen in Terabytegröße, in denen Tausende von Servern erforderlich sind.

Katalogservice

Der Katalogservice steuert die Verteilung von Shards und ermittelt und überwacht die Vitalität der Container-Server im Datengrid. Der Katalogservice enthält Logik, die inaktiv bleibt und nur geringen Einfluss auf die Skalierbarkeit hat. Der Katalogservice ist so konzipiert, dass er Hunderte gleichzeitig verfügbarer Container-Container bedienen kann, und führt Services für die Verwaltung der Container-Server aus.

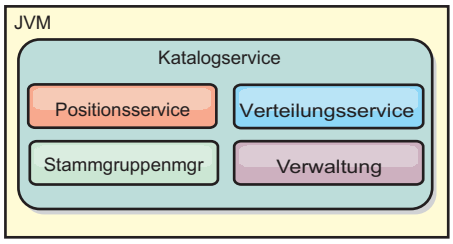


Abbildung 2. Katalogservice

Die Zuständigkeiten des Katalogservers setzen sich aus den folgenden Services zusammen:

Arbeitsumgebung
Der Positionsservice wird in den Membern des Datengrids ausgeführt, um Clients und Container-Servern Positionsdaten zu liefern. Container-Server registrieren sich beim Positionsservice, um die gehosteten Anwendungen zu registrieren. Clients können den Positionsservice dann verwenden, um Container-Server für das Hosten von Anwendungen zu suchen.

Verteilungsservice
Der Katalogservice verwaltet die Verteilung von Shards auf die verfügbaren Container-Server. Der Verteilungsservice ist für die Verwaltung der gleichmäßigen Verteilung der Shards auf physische Ressourcen und die Zuordnung einzelner Shards zu ihren Host-Container-Servern zuständig. Der Verteilungsservice wird als einer von N ausgewählten Services im Cluster und im Datengrid ausgeführt. Das bedeutet, dass genau eine Instanz des Verteilungsservice aktiv ist. Wenn eine Instanz ausfällt, wird ein anderer Prozess ausgewählt, der die Arbeit übernimmt. Aus Redundanzgründen wird der Status des Katalogservice in allen Servern, in denen der Katalogservice ausgeführt, repliziert.

Stammgruppenmanager
Die Stammgruppe verwaltet die Peergruppierung für die Verfügbarkeits-

überwachung, fasst Container-Server zu kleinen Servergruppen zusammen und bindet die Servergruppen automatisch ein.

Der Katalogservice verwendet den High Availability Manager (kurz HA-Manager), um Prozesse für die Überwachung der Verfügbarkeit zu gruppieren. Jede Prozessgruppierung ist eine Stammgruppe. Der Stammgruppenmanager gruppiert die Prozesse dynamisch. Diese Prozesse werden für die Skalierbarkeit klein gehalten. Jede Stammgruppe wählt ein führendes Member aus, das für das Senden von Überwachungssignalnachrichten an den Stammgruppenmanager zuständig ist. Diese Nachrichten erkennen, ob ein Member ausgefallen oder noch verfügbar ist. Der Überwachungssignalmechanismus wird auch verwendet, um festzustellen, ob alle Member einer Gruppe ausgefallen sind, was dazu führt, dass die Kommunikation mit dem führenden Member scheitert.

Der Stammgruppenmanager ist für die Organisation der Container in kleinen Servergruppen zuständig, die lose zu einem Datengrid zusammengefasst werden. Wenn ein Container-Server den ersten Kontakt zum Katalogservice herstellt, wartet er auf die Zuteilung einer neuen oder vorhandenen Gruppe. Eine Implementierung von eXtreme Scale setzt sich aus vielen solcher Gruppen zusammen, und diese Gruppierung ist ein Enabler für die Skalierbarkeit von Schlüsseln. Jede Gruppe setzt sich aus Java Virtual Machines zusammen. Ein ausgewähltes führendes Member verwendet den Überwachungssignalmechanismus, um die Verfügbarkeit der anderen Gruppen zu überwachen. Das führende Member leitet die Verfügbarkeitsinformationen an den Katalogservice weiter, damit dieser durch Neuzuordnung und Routenweiterleitung auf Fehler reagieren kann.

Verwaltung
Der Katalogservice ist auch der logische Einstiegspunkt für die Systemverwaltung. Der Katalogservice enthält eine Managed Bean (MBean) und stellt JMX-URLs (Java Management Extensions) für alle vom Katalogservice verwalteten Server bereit.

Für die hohe Verfügbarkeit konfigurieren Sie eine Katalogservicedomäne. Eine Katalogservicedomäne setzt sich aus mehreren Java Virtual Machines, einschließlich einer Master-JVM und einer Reihe von Ausweich-JVMs zusammen. Weitere Informationen finden Sie unter „Katalogservice mit hoher Verfügbarkeit" auf Seite 111.

Container-Server, Partitionen und Shards

Der Container-Server speichert Anwendungsdaten für das Datengrid. Diese Daten werden gewöhnlich in mehrere Teile, so genannte Partitionen aufgeteilt. Partitionen werden auf mehrere Shard-Container verteilt. Jeder Container-Server wiederum hostet einen Teil der Gesamtdaten. Eine JVM kann einen oder mehrere Shard-Container hosten, und jeder Shard-Container kann mehrere Shards hosten.

Hinweis: Planen Sie die Größe des Heapspeichers für die Container-Server, die alle Daten hosten. Konfigurieren Sie die Einstellungen für den Heapspeicher entsprechend.

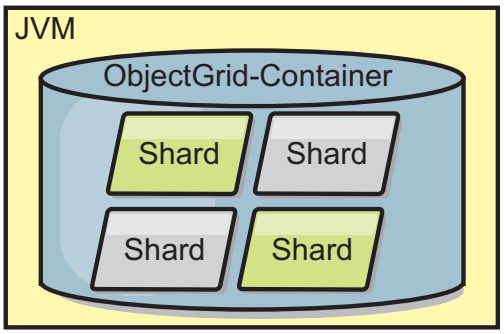


Abbildung 3. Container-Server

Partitionen enthalten einen Teil der Daten des Grids. WebSphere eXtreme Scale stellt automatisch mehrere Partitionen in einen einzigen Shard-Container und verteilt die Partitionen auf weitere Container-Server, sobald diese verfügbar werden.

Wichtig: Sie müssen die Anzahl der Partitionen vor der endgültigen Implementierung sorgfältig auswählen, da sie nicht dynamisch geändert werden kann. Ein Hash-Mechanismus wird verwendet, um Partitionen im Netz zu suchen, und eXtreme Scale hat nach der Implementierung der Daten keine Möglichkeit, ein erneutes Hashing für die gesamten Daten durchzuführen. Als allgemeine Regel gilt, dass die Anzahl der Partitionen zu hoch geschätzt werden kann.

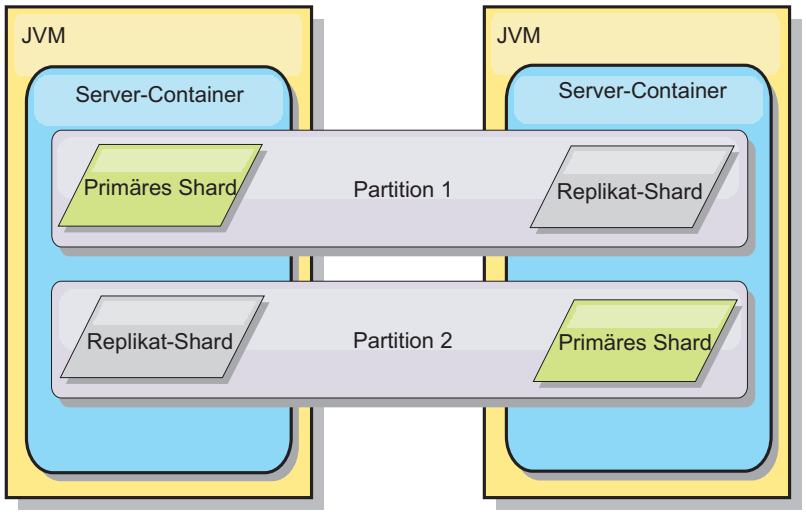


Abbildung 4. Partition

Shards sind Instanzen von Partitionen und haben eine von zwei Rollen: primäres Shard oder Replikat-Shard. Das primäre Shard und seine Replikate bilden die physische Manifestation der Partition. Jede Partition hat mehrere Shards, die jeweils alle in dieser Partition vorhandenen Daten enthalten. Ein Shard ist das primäre Shard, und die anderen Shards sind Replikate oder Replikat-Shards, d. h. redundante Kopien der Daten im primären Shard. Ein primäres Shard ist die einzige Partitionsinstanz, die Transaktionen das Schreiben in den Cache erlaubt. Ein Replikat-Shard ist eine "gespiegelte" Instanz der Partition. Es empfängt synchron oder asynchron Aktualisierungen vom primären Shard. Das Replikat-Shard erlaubt Transaktionen nur das Lesen von Daten aus dem Cache. Replikate befinden sich nie in demselben Container-Server wie das primäre Shard und normalerweise nicht

einmal auf derselben Maschine wie das primäre Shard.

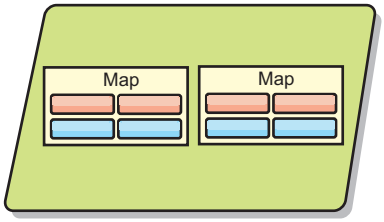


Abbildung 5. Shard

Um die Verfügbarkeit der Daten oder die Persistenzgarantien zu erhöhen, replizieren Sie die Daten. Die Replikation bedeutet jedoch einen höheren Aufwand für die Transaktion und damit Leistungseinbußen zu Gunsten der Verfügbarkeit. Mit eXtreme Scale können Sie den Aufwand steuern, da synchrone und asynchrone Replikation sowie Hybridreplikationsmodelle unterstützt werden, die synchrone und asynchrone Replikationsmodi verwenden. Ein synchrones Replikat-Shard empfängt Aktualisierungen im Rahmen der Transaktion des primären Shards, um die Datenkonsistenz zu gewährleisten. Ein synchrones Replikat kann die Antwortzeit verdoppeln, da die Transaktion sowohl im primären Shard als auch im synchronen Replikat festgeschrieben werden muss, bevor sie beendet werden kann. Ein asynchrones Replikat-Shard empfängt Aktualisierungen nach der Transaktionsfestschreibung, um die Auswirkungen auf die Leistung zu begrenzen, birgt aber das Risiko eines Datenverlusts, da das asynchrone Replikat mehrere Transaktionen hinter dem primären Shard zurückliegen kann.

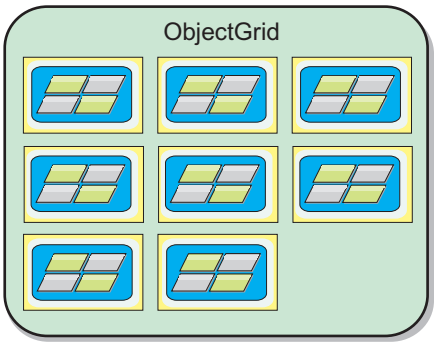


Abbildung 6. ObjectGrid

Maps

Eine Map ist ein Container für Schlüssel/Wert-Paare, in der eine Anwendung einen Wert nach einem Schlüssel indexiert speichern kann. Maps unterstützen Indizes, die Indexattributen im Schlüssel oder Wert hinzugefügt werden können. Diese Indizes werden automatisch von der Abfragelaufzeitumgebung verwendet, um die effizienteste Methode für die Durchführung einer Abfrage zu bestimmen.

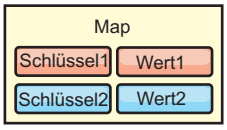


Abbildung 7. Map

Ein Mapset ist eine Sammlung von Maps mit einem gemeinsamen Partionierungsalgorithmus. Die Daten in den Maps werden auf der Basis der Richtlinie repliziert, die im Mapset definiert ist. Ein Mapset wird nur für verteilte Topologien verwendet und wird für lokale Topologien nicht benötigt.

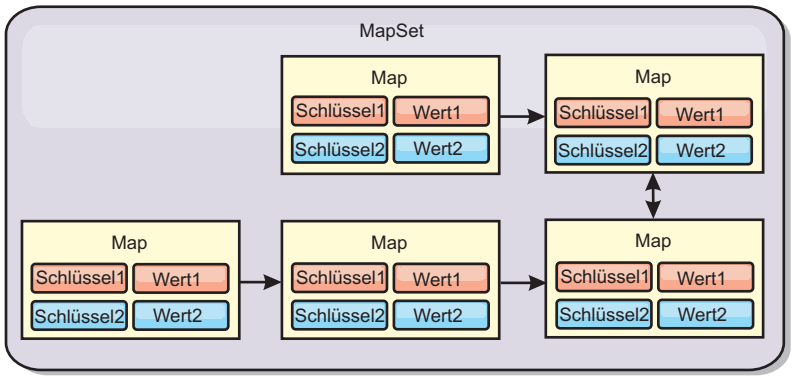


Abbildung 8. Mapsets

Einem Mapset kann ein Schema zugeordnet sein. Ein Schema setzt sich aus den Metadaten zusammen, die die Beziehungen zwischen den einzelnen Maps beschreiben, wenn homogene Objekttypen oder Entitäten verwendet werden.

WebSphere eXtreme Scale kann über die Anwendungsprogrammierschnittstelle (API, Application Programming Interface) "ObjectMap" serialisierbare Java-Objekte in jeder der Maps speichern. Es kann ein Schema für die Maps definiert werden, um die Beziehungen zwischen den Objekten in den Maps zu identifizieren, wobei jede Map Objekte eines einzelnen Typs enthält. Die Definition eines Schemas für Maps ist erforderlich, um den Inhalt der Mapobjekte abzufragen. In WebSphere eXtreme Scale können mehrere Mapschemas definiert werden.

Außerdem kann WebSphere eXtreme Scale über die API "EntityManager" Entitäten speichern. Jede Entität ist einer Map zugeordnet. Das Schema für ein Entitäts-Mapset wird automatisch über eine XML-Entitätsdeskriptordatei oder über annotierte Java-Klassen erkannt. Jede Entität besitzt einen Satz von Schlüsselattributen und einen Satz von Attributen ohne Schlüsselfunktion. Eine Entität kann außerdem Beziehungen zu anderen Entitäten haben. WebSphere eXtreme Scale unterstützt 1:1-, 1:N-, N-1- und N:N-Beziehungen. Jede Entität ist physisch einer einzigen Map im Mapset zugeordnet. Entitäten ermöglichen Anwendungen die problemlose Verwendung komplexer Objektgraphen, die sich über mehrere Maps erstrecken. Eine verteilte Topologie kann mehrere Entitätsschemas haben.

Weitere Informationen finden Sie unter Caching von Objekten und ihren Beziehungen (API "EntityManager").

Clients

Clients stellen eine Verbindung zu einem Katalogservice her, rufen eine Beschreibung der Servertopologie ab und kommunizieren bei Bedarf mit jedem Server direkt. Wenn sich die Servertopologie ändert, weil neue Server hinzugefügt werden oder vorhandene Server ausfallen, leitet der dynamische Katalogservice den Client an den Server weiter, der die Daten enthält. Clients müssen die Schlüssel der Anwendungsdaten untersuchen, um festzustellen, an welche Partition die Anforderung weitergeleitet werden muss. Clients können in einer einzigen Transaktion Daten aus mehreren Partitionen lesen. Sie können jedoch innerhalb einer einzigen Transaktion nur eine einzige Partition aktualisieren. Nachdem der Client einige Einträge aktualisiert hat, muss die Clienttransaktion diese Partition für Aktualisierungen verwenden.

8.6+ Sie können zwei Typen von Clients verwenden: Java-Clients und .NET-Clients. Sie können nur Java-Clients, nur .NET-Clients oder beide Typen verwenden, um auf denselben Katalogserver und dasselbe Datengrid zuzugreifen.

Java-Clients

Java-Clientanwendungen werden in der Java Virtual Machines (JVM) ausgeführt und stellen eine Verbindung zum Katalogservice und zu Container-Servern her.

- Ein Katalogservice existiert in einem eigenen Datengrid von JVMs. Sie können einen einzigen Katalogservice verwenden, um mehrere Clients oder Container-Server zu verwalten.
- Ein Container-Server kann in einer eigenen JVM gestartet oder zusammen mit anderen Containern für verschiedene Datengrids in eine beliebige JVM geladen werden.
- Ein Client kann in einer beliebigen JVM existieren und mit einem oder mehreren Datengrids kommunizieren. Ein Client kann auch in derselben JVM wie ein Container-Server existieren.

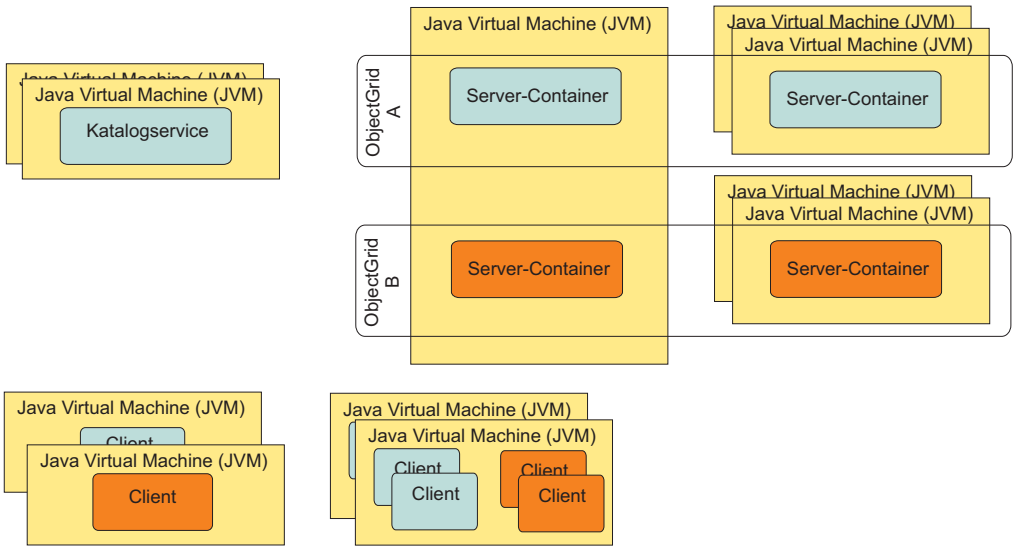


Abbildung 9. Mögliche Topologien

8.6+

.NET-Clients

.NET-Clients arbeiten ähnlich wie Java-Clients, werden aber nicht in JVMs ausgeführt. .NET-Clients werden remote über Ihre Katalog- und Container-Server installiert. Sie stellen die Verbindung zum Katalogservice über die Anwendung her. Mit einer .NET-Clientanwendung können Sie eine Verbindung zu demselben Datengrid wie Ihre Java-Clients herstellen. Weitere Informationen zur gemeinsamen Verwendung von Java- und .NET-Clients finden Sie unter „Szenario: Unternehmensdatengrid konfigurieren” auf Seite 193.

Übersicht über Unternehmensdatengrids

Unternehmensdatengrids verwenden den Transportmechanismus "eXtremeIO" und ein neues Serialisierungsformat. Mit dem neuen Transport und Serialisierungsformat können Sie Java- und .NET-Clients mit demselben Datengrid verbinden.

Mit dem Unternehmensdatengrid können Sie mehrere Typen von Anwendungen erstellen, die in verschiedenen Programmiersprachen geschrieben sind, um auf dieselben Objekte im Datengrid zuzugreifen. In früheren Releases durften Datengridanwendungen nur in der Programmiersprache Java geschrieben werden. Mit der Unternehmensdatengridfunktion können Sie .NET-Anwendungen schreiben, die Objekte in demselben Datengrid erstellen, abrufen, aktualisieren und löschen, wie die Java-Anwendung.

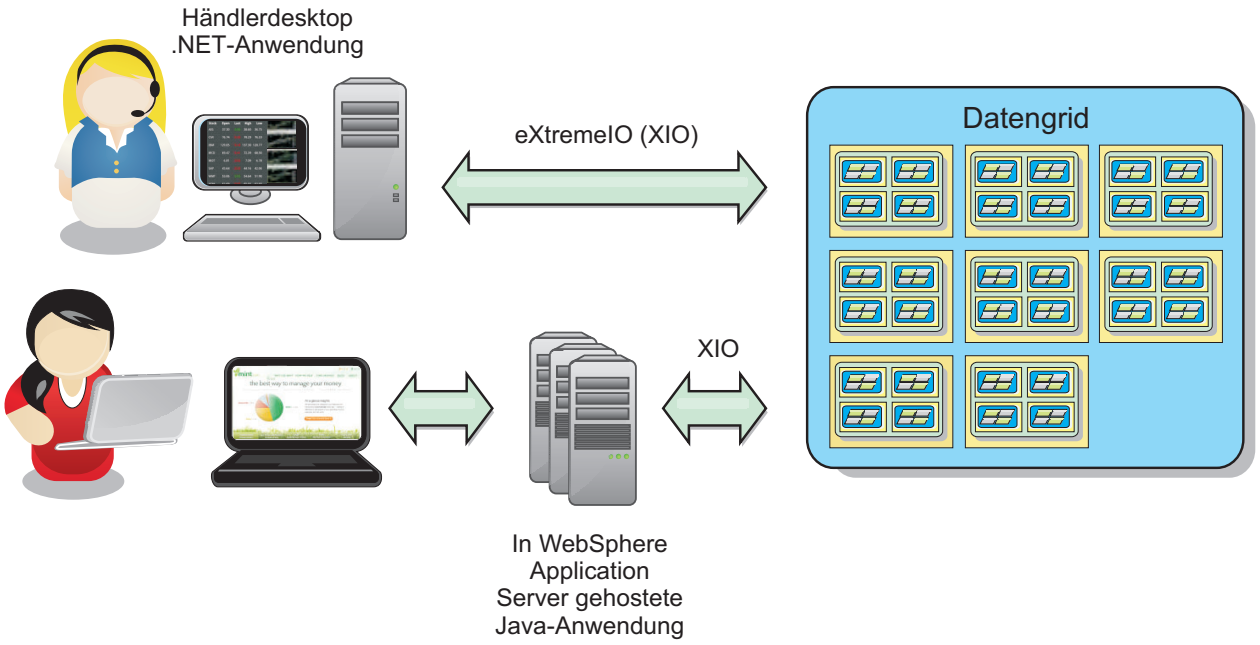


Abbildung 10. Allgemeine Übersicht über Unternehmensdatengrids

Objektaktualisierungen über verschiedene Anwendungen

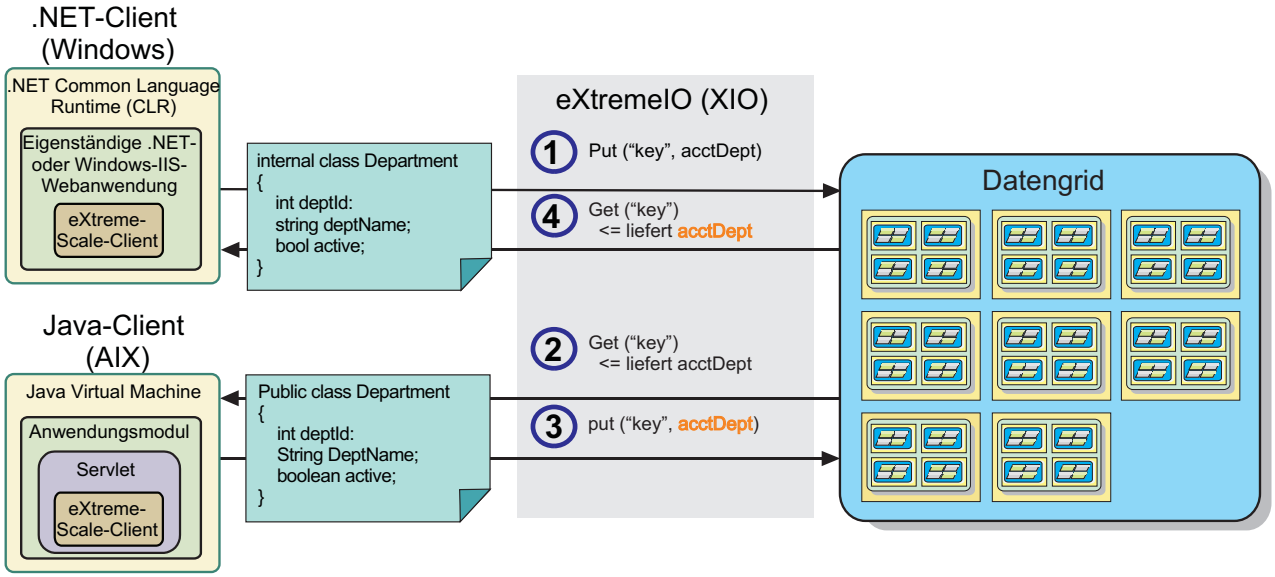


Abbildung 11. Ablauf der Objektaktualisierung in einem Unternehmensdatengrid

1. Der .NET-Client speichert Daten in seinem eigenen Format im Datengrid.
2. Die Daten werden in einem universellen Format gespeichert, sodass die Daten bei Anforderung durch den Java-Client in das Java-Format konvertiert werden können.
3. Der Java-Client aktualisiert die Daten und speichert sie dann erneut.
4. Beim Zugriff auf die aktualisierten Daten durch den .NET-Client werden die Daten in das .NET-Format konvertiert.

Transportmechanismus

eXtremeIO (XIO) ist ein plattformübergreifendes Transportprotokoll XIO ersetzt den Java-gebundenen Object Request Broker (ORB). Mit dem ORB wird WebSphere eXtreme Scale an native Java-Clientanwendungen gebunden. XIO ist ein angepasster Transportmechanismus, der speziell für das Datencaching bestimmt ist und Clientanwendungen, die in anderen Programmiersprachen geschrieben sind, ermöglicht, die Verbindung zum Datengrid herzustellen.

Serialisierungsformat

XDF (eXtreme Data Format) ist ein plattformübergreifendes Serialisierungsformat. XDF ersetzt die Java-Serialisierung in Maps, in denen das copyMode-Attribut in der ObjectGrid-XML-Deskriptordatei auf den Wert `COPY_TO_BYTES` gesetzt ist. Mit XDF ist die Leistung höher, und die Daten sind kompakter. Außerdem ermöglicht die Einführung von XDF Clientanwendungen, die in verschiedenen Programmiersprachen geschrieben sind, eine Verbindung zu demselben Datengrid herzustellen.

Weiterentwicklung von Klassen

eXtreme Data Format (XDF) lässt die Weiterentwicklung von Klassen zu. Bei der Weiterentwicklung von Klassen können Sie die Klassendefinitionen, die im Datengrid verwendet werden, ohne Beeinflussung älterer Anwendungen, die frühere Versionen der Klassen verwenden, weiterentwickeln. Diese älteren Klassen greifen auf Daten in derselben Map wie die neuen Anwendungen zu.

Übersicht

Die Weiterentwicklung von Klassen ist eine weitere Erweiterung der Identifikation von Klassen und Feldern, die bestimmen, ob zwei Typen ausreichend kompatibel sind, um miteinander zu funktionieren. Klassen können miteinander funktionieren, wenn eine der Klassen weniger Felder als die andere Klasse hat. Die folgenden Benutzerszenarien sind in der XDF-Implementierung abgebildet:

Mehrere Versionen derselben Objektklasse
In diesem Szenario gibt es eine Map in einer Einzelhandelsanwendung, die zur Überwachung von Kunden verwendet wird. Diese Map hat zwei verschiedene Schnittstellen. Eine Schnittstelle ist für die Webeinkäufe bestimmt. Die zweite Schnittstelle ist für Telefoneinkäufe bestimmt. In Version 2 dieser Einzelhandelsanwendung entscheiden Sie, Webeinkäufern basierend auf deren Kaufgewohnheiten Rabatte einzuräumen. Dieser Rabatt wird zusammen mit dem Kundenobjekt (Customer) gespeichert. Die Mitarbeiter für Telefoneinkäufe verwenden weiterhin Version 1 der Anwendung, die das neue Rabattfeld in der Webversion nicht kennt. Sie möchten, dass Kundenobjekte aus Version 2 der Anwendung mit Kundenobjekten funktionieren, die mit Version 1 der Anwendung erstellt wurden, und umgekehrt.

Mehrere Versionen einer anderen Objektklasse
In diesem Szenario gibt es eine Einzelhandelsanwendung, die in Java geschrieben ist und eine Map von Kundenobjekten (Customer) verwaltet. Außerdem gibt es eine andere Anwendung, die in C# geschrieben ist und dazu verwendet wird, den Bestand im Lager zu verwalten und Waren an Kunden auszuliefern. Diese Klassen sind momentan basierend auf den Namen der Klassen, Felder und Typen kompatibel. In der Java-Einzelhandelsanwendung möchten Sie nun dem Kundendatensatz eine Option hinzufügen, um den Verkäufer einem Kundenkonto zuzuordnen. Sie möchten die Lageranwendung jedoch nicht mit diesem Feld aktualisieren, weil das Feld im Lager nicht erforderlich ist.

Mehrere inkompatible Versionen derselben Klasse
In diesem Szenario enthalten die Einzelhandelsanwendung und die Anwendung zur Bestandsführung jeweils ein Kundenobjekt (Customer). Die Anwendung zur Bestandsführung verwendet ein ID-Feld mit dem Datentyp "String", und die Einzelhandelsanwendung verwendet ein ID-Feld mit dem Datentyp "Integer". Diese Datentypen sind nicht kompatibel. Deshalb werden die Objekte wahrscheinlich nicht in derselben Map gespeichert. Diese Objekte müssen von der XDF-Serialisierung verarbeitet und als zwei unterschiedliche Datentypen behandelt werden. Obwohl dieses Szenario eigentlich keine Weiterentwicklung von Klassen ist, muss es während des Gesamtanwendungsentwurfs berücksichtigt werden.

Entschluss zur Weiterentwicklung

XDF versucht, eine Klasse weiterzuentwickeln, wenn die Klassennamen übereinstimmen und die Feldnamen keine widersprüchlichen Typen haben. Die Verwendung der Annotationen "ClassAlias" und "FieldAlias" sind hilfreich, wenn Sie versuchen, Klassen von C#- und Java-Anwendungen abzugleichen, in denen die Namen der Klassen oder Felder geringfügig verschieden sind. Sie können diese Annotationen in die Java- und/oder C#-Anwendung einfügen. Die Suche der Klasse in der Java-Anwendung kann jedoch weniger effizient sein als die Definition der Annotation "ClassAlias" in der C#-Anwendung. Weitere Informationen zu den An-

notationen "ClassAlias" und "FieldAlias" finden Sie unter „ClassAlias- und FieldAlias-Annotationen” auf Seite 200.

Auswirkung fehlender Felder in serialisierten Daten

Der Konstruktor der Klasse wird während der Deserialisierung nicht aufgerufen. Deshalb haben alle fehlenden Felder einen Standardwert, der den Feldern basierend auf der Sprache zugeordnet wird. Die Anwendung, die neue Felder hinzufügt, muss in der Lage sein, die fehlenden Felder zu erkennen und zu reagieren, wenn eine ältere Version der Klasse abgerufen wird.

Aktualisierung der Daten ist die einzige Möglichkeit für ältere Anwendungen, die neueren Felder zu verwalten

Eine Anwendung könnte eine Abrufoperation ausführen und die Map mit einer älteren Version der Klasse aktualisieren, in der einige Felder im serialisierten Wert des Clients fehlen. Der Server führt dann die Werte auf dem Server zusammen und bestimmt, ob Felder aus der ursprünglichen Version in den neuen Datensatz eingefügt werden müssen. Wenn eine Anwendung eine Abrufoperation ausführt und dann einen Eintrag entfernt und einfügt, gehen die Felder aus dem ursprünglichen Wert verloren.

Zusammenführungsfunktionen

Objekte in einem Array oder in einer Sammlung werden von XDF nicht zusammengeführt. Es ist nicht immer klar, ob eine Aktualisierung eines Arrays oder einer Sammlung dazu bestimmt ist, die Elemente dieses Arrays oder den Typ zu ändern. Wenn beim Verschieben eines Eintrags im Array eine Zusammenführung basierend auf der Positionierung stattfindet, kann XDF Felder zusammenführen, die nicht für eine Zuordnung bestimmt sind. Deshalb versucht XDF nicht, den Inhalt von Arrays oder Sammlungen zusammenzuführen. Wenn Sie jedoch ein Array in eine neuere Version einer Klassendefinition hinzufügen, wird das Array wieder mit der früheren Version der Klasse zusammengeführt.

IBM eXtremeMemory

IBM eXtremeMemory ermöglicht die Speicherung von Objekten im nativen Speicher anstatt im Java-Heapspeicher. Durch die Auslagerung von Objekten aus dem Java-Heapspeicher können Sie Garbage-Collection-Pausen vermeiden und damit eine konstantere Leistung und vorhersehbare Antwortzeiten erzielen.

Die JVM (Java Virtual Machine) stützt sich bei der Erfassung, Verkleinerung und Vergrößerung des Prozessspeichers auf Nutzungsheuristik. Der Garbage-Collector führt diese Operationen aus. Durch die Garbage-Collection entstehen jedoch Kosten. Die Kosten für die Garbage-Collection nehmen mit der Größe des Java-Heapspeichers und der Anzahl der Objekte im Datengrid zu. Die JVM stellt verschiedene heuristische Verfahren für verschiedene Anwendungsfälle und Ziele bereit: optimaler Durchsatz, optimale Pausenzeiten, auf dem Objektalter basierende Garbage-Collection, gleichmäßige Verteilung und Garbage-Collection in Echtzeit. Kein heuristisches Verfahren ist perfekt. Ein einziges heuristisches Verfahren kann nicht für alle Konfigurationen geeignet sein.

WebSphere eXtreme Scale verwendet Datencaching mit verteilten Maps, die Einträge mit einem bekannten Lebenszyklus haben. Dieser Lebenszyklus enthält die folgenden Operationen: GET, INSERT, DELETE und UPDATE. Durch die Verwendung dieser bekannten Maplebenszyklen kann eXtremeMemory die

Hauptspeicherbelegung für Datengridobjekte in Container-Servern effizienter verwalten als der Standard-JVM-Garbage-Collector.

Die folgende Abbildung veranschaulicht, wie mit eXtremeMemory konsistentere relative Antwortzeiten in der Umgebung erzielt werden. Im Bereich der hohen Perzentile sind die relativen Antwortzeiten bei Anforderungen, die eXtremeMemory verwenden, wesentlich kürzer. In der Abbildung sind die 95-100 Perzentile dargestellt.

.

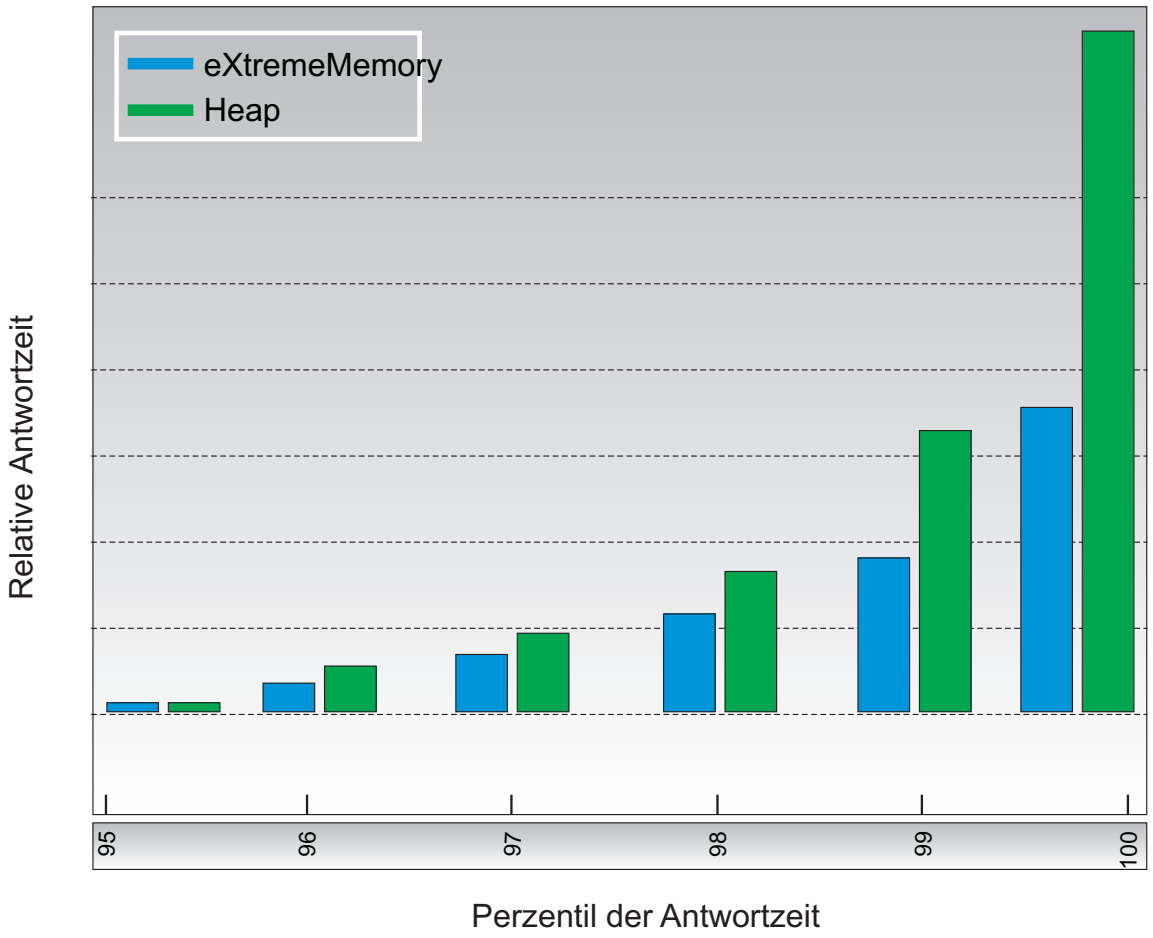


Abbildung 12. Antwortzeiten von eXtremeMemory und Heapspeicher im Vergleich

Zonen

Mit Zonen können Sie die Shardverteilung steuern. Zonen sind benutzerdefinierte logische Gruppierungen physischer Server. Im Folgenden finden Sie Beispiele für verschiedene Zonentypen: verschiedene Blade-Server, Blade-Server-Gehäuse, Stockwerke, Gebäude oder verschiedene geografische Standorte in einer Umgebung mit mehreren Rechenzentren. Ein weiterer Anwendungsfall für Zonen sind virtualisierte Umgebungen, in denen viele Serverinstanzen, jeweils mit einer eindeutigen IP-Adresse, auf demselben physischen Server ausgeführt werden.

Zwischen Rechenzentren definierte Zonen

Der klassische Beispiel- und Anwendungsfall für Zonen sind zwei oder mehr Rechenzentren, die über verschiedene geografische Standorte verteilt sind. Bei verteilten Rechenzentren wird das Datengrid über verschiedene Standorte verteilt, um bei Fehlern in den Rechenzentrum eine Möglichkeit der Wiederherstellung zu haben. Sie können beispielsweise sicherstellen, dass ein vollständiger Satz asynchroner Replikat-Shards für Ihr Datengrid in einem fernen Rechenzentrum verfügbar ist. Diese Strategie ermöglicht Ihnen nach einem Fehler im lokalen Rechenzentrum eine transparente Wiederherstellung ohne Datenverlust. Rechenzentren selbst haben Hochgeschwindigkeitsnetze mit geringen Latenzzeiten. Die Kommunikation zwischen zwei Rechenzentren hat jedoch höhere Latenzzeiten. Synchrone Replikate werden in jedem Rechenzentrum verwendet, in dem die geringen Latenzzeiten die Auswirkungen der Replikation auf die Antwortzeiten minimiert. Die asynchrone Replikation verringert die Auswirkungen auf die Antwortzeiten. Die geografische Entfernung gewährleistet die Verfügbarkeit der Daten, falls im lokalen Rechenzentrum ein Fehler auftritt.

Im folgenden Beispiel haben die primären Shards für die Zone "Chicago" Replikate in der Zone "London". Primäre Shards für die Zone "London" haben Replikate in der Zone "Chicago".

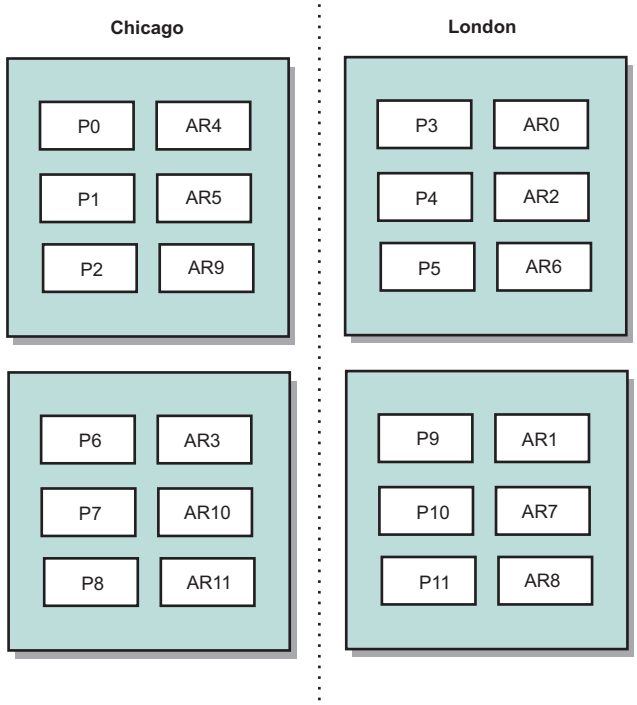


Abbildung 13. Primäre Shards und Replikate in Zonen

Die Shard-Verteilung wird mit drei Konfigurationseinstellungen in eXtreme Scale gesteuert:

- Implementierungsdatei
- Gruppencontainer
- Regeln

In den folgenden Abschnitten sind die verschiedenen Optionen, flexibel in aufsteigender Reihenfolge ihrer Komplexität, beschrieben.

Entwicklungsmodus inaktivieren

Definieren Sie in Ihrer XML-Implementierungsdatei `developmentMode="false"`.

Dieser einfache Schritt aktiviert die erste Shard-Verteilungsrichtlinie in eXtreme Scale.

Weitere Informationen zur XML-Datei finden Sie unter XML-Deskriptordatei für Implementierungsrichtlinie.

Richtlinie 1: Shards für dieselbe Partition werden auf separate physische Server verteilt.

Stellen Sie sich eine einfaches Beispiel eines Datengrids mit einem einzigen Replikat-Shard vor. Bei dieser Richtlinie befinden sich die primären Shards und Replikat-Shards für jede Partition auf verschiedenen physischen Servern. Wenn einer der physischen Server ausfällt, gehen keine Daten verloren. Das primäre Shard oder Replikat-Shard für jede Partition befindet sich auf unterschiedlichen Servern, die nicht ausgefallen sind, oder beide Shards befinden sich auf einem anderen physischen Server, der nicht ausgefallen ist.

Die hohe Verfügbarkeit und Einfachheit dieser Richtlinie macht die Richtlinie zur effizientesten Einstellung für alle Produktionsumgebungen. In vielen Fällen ist die Anwendung dieser Richtlinie der einzige erforderliche Schritt für eine effiziente Steuerung der Shard-Verteilung in Ihrer Umgebung.

Wenn Sie diese Richtlinie anwenden, wird ein physischer Server anhand einer IP-Adresse identifiziert. Shards werden an Container-Server verteilt. Container-Server haben eine IP-Adresse, z. B. den Parameter **`-listenerHost`** im Script zum Starten des Servers. Mehrere Container-Server können dieselbe IP-Adresse haben.

Da ein physischer Server mehrere IP-Adressen hat, können Sie den nächsten Schritt ausführen, um Ihre Umgebung noch differenzierter zu steuern.

Zonen für Gruppen-Container-Server definieren

Container-Server werden mit dem Parameter **`-zone`** im Script zum Starten des Servers Zonen zugeordnet. In einer Umgebung von WebSphere Application Server werden Zonen über Knotengruppen mit einem bestimmten Namensformat definiert: `Replikationszone<Zone>`. Auf diese Weise wählen Sie den Namen und die Zugehörigkeit Ihrer Zonen aus. Weitere Informationen finden Sie unter Zonen für Container-Server definieren.

Richtlinie 2: Shards für dieselbe Partition werden separaten Zonen zugeordnet.

Angenommen, Sie erweitern das Datengridbeispiel mit einem einzigen Replikat-Shard, indem Sie es auf zwei Rechenzentren verteilen. Definieren Sie jedes Rechenzentrum als unabhängige Zone. Verwenden Sie den Zonennamen "DC1" für die Container-Server im ersten Rechenzentrum und den Namen "DC2" für die Container-Server im zweiten Rechenzentrum. Bei dieser Richtlinie befinden sich die primären Shards und die Replikat-Shards für jede Partition in verschiedenen Rechenzentren. Wenn ein Rechenzentrum ausfällt, gehen keine Daten verloren. Das primäre Shard oder das Replikat-Shard jeder Partition befindet sich im anderen Rechenzentrum.

Mit dieser Richtlinie können Sie die Shard-Verteilung durch die Definition von Zonen steuern. Sie wählen Ihre physische oder logische Grenze bzw. gewünschte Gruppierung aus. Anschließend wählen Sie einen eindeutigen Zonennamen für jede Gruppe aus und starten die Container-Server in jeder Ihrer Zonen mit dem Namen der entsprechenden Zone. Damit verteilt eXtreme Scale die Shards so, dass Shards für dieselbe Partition separaten Zonen zugeordnet werden.

Zonenregeln definieren

Die differenzierteste Steuerungsebene für die Shard-Verteilung wird mit Zonenregeln erreicht. Zonenregeln werden im Element "zoneMetadata" des XML-Implementierungsrichtliniendeskriptor von eXtreme Scale definiert. Eine Zonenregel definiert eine Gruppe von Zonen, auf die die Shards verteilt werden. Ein Element "shardMapping" ordnet ein Shard einer Zonenregel zu. Das Attribut "shard" des Elements "shardMapping" gibt den Shard-Typ an:

- P gibt das primäre Shard an.
- S gibt synchrone Replikat-Shards an.
- A gibt asynchrone Replikat-Shards an.

Wenn mehrere synchrone oder asynchrone Replikate vorhanden sind, müssen Sie shardMapping-Elemente des entsprechenden Shard-Typs angeben. Das Attribut "exclusivePlacement" des Elements "zoneRule" bestimmt die Verteilung der Shards derselben Partition auf die Zonen. Die gültigen Werte für das Attribut "exclusivePlacement" sind folgende:

- `true` (ein Shard kann nicht derselben Zone wie ein anderes Shard derselben Partition zugeordnet werden)

  Hinweis: Wenn Sie den Wert "true" angeben, müssen Sie mindestens so viele Zonen in der Regel wie Shards haben, die diese verwenden. Auf diese Weise wird sichergestellt, dass jedes Shard einer eigenen Zone zugeordnet werden kann.
- `false` (Shards aus derselben Partition können derselben Zone zugeordnet werden)

Die Standardeinstellung ist "true".

Weitere Informationen finden Sie im Abschnitt Beispiel: Zonendefinitionen in der XML-Implementierungsrichtliniendeskriptordatei.

Erweiterte Anwendungsfälle

Im Folgenden werden verschiedene Anwendungsfälle für Shard-Verteilungsstrategien beschrieben:

Schrittweise Upgrades

Stellen Sie sich ein Szenario vor, in dem Sie schrittweise Upgrades auf Ihre physischen Server anwenden möchten, einschließlich Wartungspaketen, die einen Neustart der Implementierung erfordern. Angenommen, Sie haben in diesem Beispiel ein Datengrid, das auf 20 physische Server mit einem einzigen synchronen Replikat verteilt ist. Jetzt möchten Sie 10 der physischen Server für Wartungszwecke gleichzeitig herunterfahren.

Wenn Sie Gruppen von 10 physischen Servern herunterfahren, darf keine Partition ihre primären und Replikat-Shards auf den Servern haben, die Sie beenden. Andernfalls gehen die Daten dieser Partition verloren.

Die einfachste Lösung ist die Definition einer dritten Zone. Anstelle von zwei Zonen mit jeweils 10 physischen Servern verwenden Sie drei Zonen: zwei mit sieben physischen Servern und eine mit sechs. Die Verteilung der Daten auf mehrere Zonen ermöglicht ein besseres Failover im Hinblick auf die Verfügbarkeit.

Eine Alternative zur Definition einer weiteren Zone ist das Hinzufügen eines Replikats.

Upgrade von eXtreme Scale durchführen

Wenn Sie das schrittweise Upgrade der eXtreme-Scale-Software mit Datengrids durchführen, die Livedaten enthalten, müssen Sie die folgenden Punkte berücksichtigen. Die Softwareversion des Katalogservice muss größer-gleich den Softwareversionen des Container-Servers sein. Aktualisieren Sie bei einer schrittweisen Strategie zuerst alle Katalogserver. Weitere Informationen zum Durchführen des Upgrades für Ihre Implementierung finden Sie im Abschnitt eXtreme-Scale-Server aktualisieren.

Datenmodell ändern

Ein zugehöriger Aspekt ist die Vorgehensweise beim Ändern des Datenmodells oder Schemas der Objekte, die im Datengrid gespeichert werden, ohne Ausfallzeiten zu verursachen. Es würde zu Unterbrechungen des Betriebs kommen, wenn Sie das Datenmodell ändern, indem Sie das Datengrid stoppen und anschließend mit den aktualisierten Datenmodellklassen im Klassenpfad des Container-Servers erneut starten, woraufhin das Datengrid erneut geladen wird. Alternativ könnten Sie ein neues Datengrid mit dem neuen Schema starten, die Daten aus dem alten Datengrid in das neue Schema kopieren und das alte Datengrid beenden.

Jeder dieser Prozesse führt zu Unterbrechungen des Betriebs und damit zu Ausfallzeiten. Wenn Sie das Datenmodell ohne Ausfallzeiten ändern möchten, speichern Sie das Objekt in einem der folgenden Formate:

- XML (als Wert
- BLOB (Binary Large Object, das mit Google protobuf erstellt wird)
- JSON (JavaScript Object Notation)

Schreiben Sie Serialisierungsmethoden (Serializer), um die Umstellung von POJO (Plain Old Java Object) zu einem dieser Formate auf der Clientseite ohne großen Aufwand durchzuführen. Schemaänderungen werden einfacher.

Virtualisierung

Cloud-Computing und Virtualisierung sind gängige aufstrebende Technologien. eXtreme Scale stellt standardmäßig sicher, dass zwei Shards für dieselbe Partition niemals derselben IP-Adresse zugeordnet werden (siehe die Beschreibung der Richtlinie 1). Wenn Sie die Implementierung in virtuellen Images wie VMware durchführen, können viele Serverinstanzen mit jeweils einer eindeutigen IP-Adresse auf demselben physischen Server ausgeführt werden. Um sicherzustellen, dass Replikate nur separaten physischen Servern zugeordnet werden, können Sie Zonen verwenden, um das Problem zu lösen. Gruppieren Sie Ihre physischen Server in Zonen, und verwenden Sie Zonenverteilungsregeln, damit die primären Shards und die Replikat-Shards in separaten Zonen verwaltet werden.

Zonen für Weitverkehrsnetze (WAN)

Sie können ein einzelnes eXtreme-Scale-Datengrid auf mehrere Gebäude oder Rechenzentren mit langsameren Netzverbindungen implementieren. Langsamere Netzverbindungen führen zu einer geringeren Bandbreite und zu einer höheren Latenzzeit. Das Risiko von Netzpartitionen ist in diesem Modus aufgrund der Netzüberlastung und anderen Faktoren ebenfalls erhöht.

Zur Bewältigung dieser Risiken organisiert der eXtreme-Scale-Katalogservice Container-Server in Stammgruppen, die Überwachungssignale austauschen, um Ausfälle von Container-Servern zu erkennen. Diese Stammgruppen können nicht auf mehrere Zonen verteilt werden. Ein führendes Member in jeder Stammgruppe überträgt die Zugehörigkeitsdaten mit Push an den Katalogservice. Der Katalogservice prüft alle gemeldeten Fehler, bevor er auf die Zugehörigkeitsinformationen reagiert, indem er Überwachungssignale mit dem fraglichen Container-Server austauscht. Wenn der Katalogservice eine falsche Fehlererkennung feststellt, führt er keine Aktionen aus. Die Stammgruppenpartition wird schnell wiederhergestellt. Außerdem sendet der Katalogservice in regelmäßigen Abständen Überwachungssignale an das führende Member jeder Stammgruppe, um Fälle von Stammgruppenisolation zu behandeln.

Bereinigungsprogramme

Java

Bereinigungsprogramme (Evictor) entfernen Daten aus dem Datengrid. Sie können ein zeitbasiertes Bereinigungsprogramm definieren. Da Bereinigungsprogramme BackingMaps zugeordnet werden, verwenden Sie die Schnittstelle "BackingMap", um das Plug-in-Bereinigungsprogramm anzugeben.

Bereinigungsprogrammtypen

Mit jeder dynamischen BackingMap wird ein TTL-Standardbereinigungsprogramm erstellt. Das Bereinigungsprogramm entfernt Einträge, die auf einem TTL-Konzept basieren.

Ohne

Gibt an, dass Einträge nicht verfallen und deshalb nicht aus der Map entfernt werden.

Erstellungszeit

Gibt an, dass Einträge auf der Basis der Erstellungszeit entfernt werden.

Wenn Sie das Bereinigungsprogramm "Erstellungszeit" (CREATION_TIME ttlType) verwenden möchten, löscht das Bereinigungsprogramm einen Eintrag, wenn dessen Lebensdauer ab Erstellung seinem TTL-Wert (Attribut TimeToLive) entspricht, das in Ihrer Anwendungskonfiguration in Millisekunden definiert ist. Wenn Sie den TTL-Wert (Attribut TimeToLive) auf 10 Sekunden setzen, wird der Eintrag automatisch zehn Sekunden nach seinem Einfügen gelöscht.

Gehen Sie beim Definieren dieses Werts für den Bereinigungsprogrammtyp "Erstellungszeit" (ttlType CREATION_TIME) mit Vorsicht vor. Die Verwendung dieses Bereinigungsprogramms empfiehlt sich, wenn dem Cache sehr viele Einträge hinzugefügt werden, die nur für eine bestimmte Zeit verwendet werden. Mit dieser Strategie werden alle erstellten Einträge nach der festgelegten Zeit entfernt.

Der Bereinigungsprogrammtyp "Erstellungszeit" (ttlType CREATION_TIME) ist in Szenarien wie der Aktualisierung von Börsennotierungen in

einem 20-Minuten-Intervall hilfreich. Angenommen, eine Webanwendung ruft Börsennotierungen ab. Die Topaktualität der Notierungen ist jedoch nicht kritisch. In diesem Fall werden die Börsennotierungen 20 Minuten lang in einem Datengrid zwischengespeichert. Nach 20 Minuten verfallen die Mapeinträge und werden daraufhin entfernt. Ungefähr alle zwanzig Minuten verwendet das Datengrid das Loader-Plug-in, um die Daten mit Daten aus der Datenbank zu aktualisieren. Die Datenbank wird alle 20 Minuten mit den topaktuellen Börsennotierungen aktualisiert.

Letzte Zugriffszeit

Gibt an, dass Einträge auf der Basis der letzten Zugriffszeit (Lese- oder Aktualisierungszugriff) entfernt werden.

Letzte Aktualisierungszeit

Gibt an, dass Einträge auf der Basis der Uhrzeit der letzten Aktualisierung entfernt werden.

Wenn Sie den Bereinigungsprogrammtyp "Letzte Zugriffszeit" (LAST_ACCESS_TIME) oder "Letzte Aktualisierungszeit" (Attribut ttlType LAST_UPDATE_TIME) verwenden, setzen Sie den TTL-Wert (Attribut TimeToLive) auf eine niedrigere Zahl als beim Bereinigungsprogrammtyp "Erstellungszeit" (ttlType CREATION_TIME). Die Einträge (Attribut TimeToLive) bei jedem Zugriff zurückgesetzt. Anders ausgedrückt, wenn der Wert des Attributs TimeToLive kleiner als 15 ist und ein Eintrag eine Lebensdauer von 14 Sekunden hat, wenn der Zugriff erfolgt, bleibt der Eintrag für weitere 15 Sekunden gültig. Wenn Sie den TTL-Wert auf einen relativ hohen Wert setzen, werden viele Einträge nie gelöscht. Setzen Sie "TimeToLive" jedoch auf einen Wert von ungefähr 15 Sekunden, können Einträge entfernt werden, wenn nicht sehr häufig auf sie zugegriffen wird.

Die Bereinigungsprogrammtypen "Letzte Zugriffszeit" (LAST_ACCESS_TIME) und "Letzte Aktualisierungszeit" (LAST_UPDATE_TIME ttlType) sind in Szenarien wie dem Speichern von Sitzungsdaten eines Clients mithilfe einer Daten-Grid-Map hilfreich. Sitzungsdaten müssen gelöscht werden, wenn der Client die Sitzungsdaten innerhalb eines bestimmten Zeitraums nicht verwendet. Die Sitzungsdaten verfallen beispielsweise, wenn innerhalb von 30 Minuten keine Aktivität des Clients erfolgt. In diesem Fall eignet sich die Verwendung des Bereinigungsprogrammtyps "Letzte Zugriffszeit" (LAST_ACCESS_TIME) oder "Letzte Aktualisierungszeit" (LAST_UPDATE_TIME) mit einem TTL-Wert von 30 Minuten für diese Anwendung.

Sie können aber auch eigene Bereinigungsprogramme schreiben. Weitere Informationen hierzu finden Sie unter Angepassten Evictor schreiben.

Plug-in-Bereinigungsprogramm

Das TTL-Standardbereinigungsprogramm verwendet eine Bereinigungsrichtlinie, die auf Zeit basiert, und die Anzahl der Einträge in der BackingMap hat keine Auswirkung auf die Verfallszeit eines Eintrags. Sie können ein optionales Plug-in-Bereinigungsprogramm verwenden, um Einträge auf der Basis der Anzahl vorhandener Einträge und nicht auf der Basis der Zeit zu entfernen.

Die folgenden optionalen Plug-in-Bereinigungsprogramme stellen einige gängige Algorithmen bereit, mit denen entschieden werden kann, welche Einträge entfernt werden sollen, wenn eine BackingMap ihr Größenlimit erreicht.

- Das Bereinigungsprogramm "LRUEvictor" verwendet einen LRU-Algorithmus (Least Recently Used), um zu entscheiden, welche Einträge entfernt werden sollen, wenn die BackingMap eine maximale Anzahl an Einträgen überschreitet.
- Das Bereinigungsprogramm "LFUEvictor" verwendet einen LFU-Algorithmus (Least Frequently Used), um zu entscheiden, welche Einträge entfernt werden sollen, wenn die BackingMap eine maximale Anzahl an Einträgen überschreitet.

Die BackingMap informiert ein Bereinigungsprogramm, wenn Einträge in einer Transaktion erstellt, geändert oder entfernt werden. Die BackingMap verfolgt diese Einträge und entscheidet, wann Einträge aus der BackingMap-Instanz entfernt werden müssen.

Eine BackingMap-Instanz hat keine Konfigurationsdaten für eine maximale Größe. Stattdessen werden Eigenschaften des Bereinigungsprogramms definiert, um das Verhalten des Bereinigungsprogramms zu steuern. Die Bereinigungsprogramme "LRUEvictor" und "LFUEvictor" haben beide eine Eigenschaft für die maximale Größe, die das Bereinigungsprogramm anweist, mit dem Entfernen von Einträgen zu beginnen, wenn die maximale Größe überschritten wird. Wie das TTL-Bereinigungsprogramm entfernen auch die Bereinigungsprogramme "LRUEvictor" und "LFUEvictor" einen Eintrag möglicherweise nicht sofort, wenn die maximale Anzahl an Einträgen erreicht ist, um die Auswirkungen auf die Leistung zu minimieren.

Wenn der LRU- bzw. LFU-Bereinigungsalgorithmus für eine bestimmte Anwendung nicht angemessen ist, können Sie eigene Bereinigungsprogramme schreiben, um eine eigene Bereinigungsstrategie zu erstellen.

Speicherbasierte Bereinigung

Wichtig: Die speicherbasierte Bereinigung wird nur in Java Platform, Enterprise Edition Version 5 und höher unterstützt.

Alle integrierten Bereinigungsprogramme unterstützten die speicherbasierte Bereinigung, die in der Schnittstelle "BackingMap" aktiviert werden kann, indem das Attribut "evictionTriggers" der BackingMap auf `MEMORY_USAGE_THRESHOLD` gesetzt wird. Weitere Informationen zum Definieren des Attributs "evictionTriggers" in der BackingMap finden Sie unter Schnittstelle "BackingMap" und unter ObjectGrid-XML-Deskriptordatei.

Die speicherbasierte Bereinigung basiert auf einem Schwellenwert für die Auslastung des Heapspeichers. Wenn die speicherbasierte Bereinigung in der BackingMap aktiviert ist und die BackingMap ein integriertes Bereinigungsprogramm hat, wird der Schwellenwert für die Auslastung auf einen Standardprozentsatz des Gesamtspeichers gesetzt, wenn noch kein Schwellenwert definiert wurde.

Wenn Sie die speicherbasierte Bereinigung verwenden, müssen Sie den Schwellenwert für die Garbage-Collection auf denselben Wert wie die Zielauslastung des Heapspeichers setzen. Beispiel: Wenn der Schwellenwert für die speicherbasierte Bereinigung auf 50 Prozent und der Schwellenwert für die Garbage-Collection auf den Standardwert von 70 Prozent gesetzt wird, kann die Auslastung des Heapspeichers auf maximal 70 Prozent ansteigen. Diese Erhöhung der Heapspeicherauslastung findet statt, weil die speicherbasierte Bereinigung erst nach einem Garbage-Collection-Zyklus ausgelöst wird.

Wenn Sie den Standardprozentsatz für den Auslastungsschwellenwert ändern möchten, setzen Sie die Eigenschaft "memoryThresholdPercentage" in den Contai-

ner- und Servereigenschaftendateien für eXtreme-Scale-Serverprozesse. Zum Definieren des Schwellenwerts für die Zielauslastung in einem Clientprozess können Sie die MBean "MemoryPoolMXBean" verwenden.

Der von WebSphere eXtreme Scale verwendete Algorithmus für die speicherbasierte Bereinigung reagiert auf das Verhalten des verwendeten Algorithmus für die Garbage-Collection. Der beste Algorithmus für die speicherbasierte Bereinigung ist der IBM Standarddurchsatz-Collector. Algorithmen für eine Garbage-Collection nach Objektalter können ein unerwünschtes Verhalten bewirken, und deshalb sollten Sie diese Algorithmen nicht für die speicherbasierte Bereinigung verwenden.

Wenn Sie den Prozentsatz für den Auslastungsschwellenwert ändern möchten, setzen Sie die Eigenschaft "memoryThresholdPercentage" in den Container- und Servereigenschaftendateien für eXtreme-Scale-Serverprozesse.

Wenn die Speicherbelegung zur Laufzeit den Schwellenwert für die Zielauslastung überschreitet, beginnen speicherbasierte Bereinigungsprogramme mit dem Entfernen von Einträgen und versuchen, die Speicherbelegung unterhalb des Schwellenwerts für die Zielauslastung zu halten. Es gibt jedoch keine Garantien, dass die Bereinigung schnell genug ist, um potenzielle abnormale Speicherbedingungen zu verhindern, wenn die Laufzeitumgebung des Systems weiterhin so schnell Speicher belegt.

Übersicht über das OSGi-Framework

OSGi definiert ein dynamisches Modulsystem für Java. Die OSGi-Serviceplattform hat eine Schichtenarchitektur und ist für die Ausführung in verschiedenen Java-Standardprofilen bestimmt. Sie können Server und Client von WebSphere eXtreme Scale in einem OSGi-Container starten.

Vorteile der Ausführung von Anwendungen im OSGi-Container

Die OSGi-Unterstützung von WebSphere eXtreme Scale ermöglicht Ihnen, das Produkt im Eclipse-Equinox-OSGi-Framework zu implementieren. Zum Aktualisieren der von eXtreme Scale verwendeten Plug-ins mussten Sie früher die Java Virtual Machine (JVM) erneut starten, um die neuen Versionen der Plug-ins anzuwenden. Mit der Funktionalität für dynamische Aktualisierungen, die das OSGi-Framework bereitstellt, können Sie die Plug-in-Klassen jetzt ohne Neustart der JVM aktualisieren. Diese Plug-ins werden über Benutzerbundles als Services exportiert. WebSphere eXtreme Scale greift über eine Suche in der OSGi-Registry auf die Services zu.

eXtreme-Scale-Container lassen Sich durch Konfiguration mit dem OSGi-Konfigurationsverwaltungssservice oder mit OSGi Blueprint einfacher und dynamischer starten. Wenn Sie ein neues Datengrid mit der zugehörigen Verteilungsstrategie implementieren möchten, können Sie eine OSGi-Konfiguration erstellen oder ein Bundle mit den eXtreme-Scale-XML-Deskriptordateien implementieren. Mit der OSGi-Unterstützung können Anwendungsbundles, die die Konfigurationsdaten von eXtreme Scale enthalten, installiert, gestartet, gestoppt, aktualisiert und deinstalliert werden, ohne das gesamte System erneut starten zu müssen. Mit dieser Funktionalität können Sie ein Upgrade der Anwendung ohne Unterbrechung des Datengrids durchführen.

Plug-in-Beans und -Services können mit angepassten Shard-Geltungsbereichen konfiguriert werden. Dies bietet Ihnen differenzierte Integrationsmöglichkeiten mit anderen Services, die im Datengrid ausgeführt werden. Jedes Plug-in kann OSGi-

Blueprint-Rankings verwenden, um sicherzustellen, dass jede Instanz des Plug-ins mit der richtigen Version aktiviert wird. Mit der bereitgestellten OSGi-Managed-Bean (MBean) und dem bereitgestellten Dienstprogramm **xscmd** können Sie die OSGi-Services des eXtreme-Plug-ins und deren Rankings abfragen.

Auf diese Weise können Administratoren potenzielle Konfigurations- und Verwaltungsfehler schnell erkennen und die Plug-in-Service-Rankings, die von eXtreme Scale verwendet werden, aktualisieren.

OSGi-Bundles

Für die Interaktion mit und die Implementierung von Plug-ins im OSGi-Framework müssen Sie *Bundles* verwenden. In der OSGi-Serviceplattform ist ein Bundle eine JAR-Datei (Java-Archiv), das Java-Code, Ressourcen und ein Manifest enthält, das das Bundle und dessen Abhängigkeiten beschreibt. Das Bundle ist die Implementierungseinheit für eine Anwendung. Das Produkt eXtreme Scale unterstützt die folgenden Bundletypen:

Server-Bundle
Das Server-Bundle ist die Datei `objectgrid.jar`, die mit der eigenständigen eXtreme-Scale-Serverinstallation installiert wird und die für die Ausführung von eXtreme-Scale-Servern erforderlich ist. Das Server-Bundle kann auch für die Ausführung von eXtreme-Scale-Clients oder lokalen speicherinternen Caches verwendet werden. Die Bundle-ID für die Datei `objectgrid.jar` ist "com.ibm.websphere.xs.server_<Version>", wobei Version das folgende Format hat: `<Version>.<Release>.<Modifikation>`. Das Server-Bundle für eXtreme Scale Version 7.1.1 ist beispielsweise com.ibm-.websphere.xs.server_7.1.1.

Client-Bundle
Das Client-Bundle ist die Datei `ogclient.jar` und wird mit eigenständigen und Clientinstallationen von eXtreme Scale installiert. Es wird verwendet, um eXtreme-Scale-Clients oder lokale Speichercaches auszuführen. Die Bundle-ID für die Datei `ogclient.jar` ist "com.ibm.websphere.xs.client-_Version", wobei Version das folgende Format hat: `<Version>.<Release>.<Modifikation>`. Das Client-Bundle für eXtreme Scale Version 7.1.1 ist beispielsweise "com.ibm.websphere.xs.client_7.1.1".

Einschränkungen

Sie können das eXtreme-Scale-Bundle nicht erneut starten, weil Sie den Object Request Broker (ORB) oder eXtremeIO (XIO) nicht erneut starten können. Zum erneuten Starten des eXtreme-Scale-Servers müssen Sie das OSGi-Framework erneut starten.

Übersicht über die Cacheintegration

Das entscheidende Element, das WebSphere eXtreme Scale eine solche Vielseitigkeit und Zuverlässigkeit ermöglicht, ist die Anwendung von Caching-Konzepten für die Optimierung der Persistenz und erneuten Erfassung von Daten in praktisch jeder Implementierungsumgebung.

Spring-Cache-Provider

Spring Framework Version 3.1 führt eine neue Cacheabstraktion ein. Mit dieser neuen Abstraktion können Sie einer vorhandenen Spring-Anwendung Caching transparent hinzufügen. Sie können WebSphere eXtreme Scale als Cache-Provider für die Cacheabstraktion nutzen.

Liberty-Profil

Das Liberty-Profil ist eine dynamische Anwendungsserverlaufzeitumgebung die viele Kompositionsmöglichkeiten bietet und sich schnell starten lässt.

Sie installieren das Liberty-Profil, wenn Sie WebSphere eXtreme Scale mit WebSphere Application Server Version 8.5 installieren. Da das Liberty-Profil keine Java Runtime Environment (JRE) enthält, müssen Sie eine JRE von Oracle oder IBM installieren.

Weitere Informationen zu unterstützten Java-Umgebungen und -Positionen finden Sie im Abschnitt zu den unterstützten Java-Mindestversionen im Information Center von WebSphere Application Server.

Dieser Server unterstützt zwei Modelle der Anwendungsimplementierung:

- Implementierung einer Anwendung durch Ablegen im Verzeichnis `dropins`
- Implementierung einer Anwendung durch Hinzufügen zur Serverkonfiguration

Das Liberty-Profil unterstützt ein Subset der folgenden Teile des vollständigen Programmiermodells von WebSphere Application Server:

- Webanwendungen
- OSGi-Anwendungen
- Java Persistence API (JPA)

Zugeordnete Services wie Transaktionen und Sicherheit werden nur in dem Maße unterstützt, das für diese Anwendungstypen und für JPA erforderlich ist.

Features sind die Funktionseinheiten, mit denen Sie die Komponenten der Laufzeitumgebung steuern können, die in einen bestimmten Server geladen werden. Das Liberty-Profil enthält die folgenden Hauptfeatures:

- Bean Validation
- Blueprint
- Java API for RESTful Web Services
- Java Database Connectivity (JDBC)
- Java Naming and Directory Interface
- Java Persistence API (JPA)
- JavaServer Faces (JSF)
- JavaServer Pages (JSP)
- Lightweight Directory Access Protocol (LDAP)
- Lokaler Connector (für JMX-Clients (Java Management Extensions))
- Überwachung
- OSGi-JPA (JPA-Unterstützung für OSGi-Anwendungen)
- Ferne Connector (für JMX-Clients)
- Secure Sockets Layer (SSL)
- Sicherheit
- Servlet

- Sitzungspersistenz
- Transaktion
- Webanwendungsbundle (Web Application Bundle, WAB)
- z/OS-Sicherheit
- z/OS-Transaktionsmanagement

Sie können mit der Laufzeitumgebung direkt oder über WebSphere Application Server Developer Tools for Eclipse arbeiten.

Auf verteilten Plattformen stellt das Liberty-Profil eine Entwicklungs- und Betriebsumgebung bereit. Auf dem Mac stellt es eine Entwicklungsumgebung bereit.

Liberty-Profil mit einer JRE eines anderen Anbieters ausführen

Wenn Sie eine von Oracle bereitgestellte JRE verwenden, sind spezielle Aspekte bei der Ausführung von WebSphere eXtreme Scale mit dem Liberty-Profil zu beachten.

Deadlock im Klassenladeprogramm

Es kann ein Deadlock im Klassenladeprogramm auftreten, das mit den folgenden JVM_ARGS-Einstellungen umgangen wurde. Wenn ein Deadlock in der BundleLoader-Logik auftritt, fügen Sie die folgenden Argumente hinzu:

```
export JVM_ARGS="$JVM_ARGS -XX:+UnlockDiagnosticVMOptions -XX:+UnsyncloadClass"
```

IBM ORB

WebSphere eXtreme Scale setzt die Verwendung des IBM ORB voraus, der in einer Installation von WebSphere Application Server enthalten ist, aber nicht im Liberty-Profil. Sie müssen die empfohlenen Verzeichnisse mit der Java-Systemeigenschaft `java.endorsed.dirs` setzen, um das Verzeichnis mit den JAR-Dateien für den IBM ORB hinzuzufügen. Die IBM ORB-JAR-Dateien sind in der eXtreme-Scale-Installation im Verzeichnis `wlp\wxs\lib\endorsed` enthalten.

Features des Servers von WebSphere eXtreme Scale für das Liberty-Profil

Features sind die Funktionseinheiten, mit denen Sie die Komponenten der Laufzeitumgebung steuern können, die in einen bestimmten Server geladen werden.

Die folgende Liste enthält Informationen zu den wichtigsten verfügbaren Features. Der Einschluss eines Features in die Konfiguration kann dazu führen, dass automatisch weitere Features geladen werden. Zu jedem Feature wird eine Kurzbeschreibung und ein Beispiel zur Deklaration des Features bereitgestellt.

Server-Feature

Das Server-Feature enthält die Funktionalität zur Ausführung eines Servers von eXtreme Scale (Katalog- und Container-Server). Fügen Sie das Server-Feature hinzu, wenn Sie einen Katalogserver im Liberty-Profil ausführen oder eine Grid-Anwendung im Liberty-Profil implementieren möchten.

Datei *<WLP-Installationsstammverzeichnis>*`/usr/server/wxsserver/server.xml`

```
<server description="WebSphere eXtreme Scale Server">

  <featureManager>
    <feature>eXtremeScale.server-1.1</feature>
```

```
  </featureManager>

  <com.ibm.ws.xs.server.config />
</server>
```

Client-Feature

Das Client-Feature enthält den größten Teil des Programmiermodells für eXtreme Scale. Fügen Sie das Client-Feature hinzu, wenn Sie eine Anwendung haben, die im Liberty-Profil ausgeführt wird, die APIs von eXtreme Scale verwendet.

Datei *<WLP-Installationsstammverzeichnis>*/usr/server/wxsclient/server.xml

```
<server description="WebSphere eXtreme Scale Client">

  <featureManager>
    <feature>eXtremeScale.client-1.1</feature>
  </featureManager>

  <com.ibm.ws.xs.client.config />
</server>
```

Web-Feature

 Das Web-Feature ist veraltet. Verwenden Sie das Feature "webapp", wenn Sie HTTP-Sitzungsdaten für die Fehlertoleranz replizieren möchten.

Das Web-Feature enthält die Funktionalität zum Erweitern der Webanwendung des Liberty-Profils. Fügen Sie das Web-Feature hinzu, wenn Sie HTTP-Sitzungsdaten für die Fehlertoleranz replizieren möchten.

Datei *<WLP-Installationsstammverzeichnis>*/usr/server/wxsweb/server.xml

```
<server description="WebSphere eXtreme Scale enabled Web Server">

  <featureManager>
    <feature>eXtremeScale.web-1.1</feature>
  </featureManager>

  <com.ibm.ws.xs.web.config />
</server>
```

8.6+ Feature "webApp"

Das Feature "webApp" enthält die Funktionalität zum Erweitern der Webanwendung des Liberty-Profils. Fügen Sie das Feature "webApp" hinzu, wenn Sie HTTP-Sitzungsdaten für die Fehlertoleranz replizieren möchten.

Datei *<WLP-Installationsstammverzeichnis>*/usr/server/wxswebapp/server.xml

```
<WLP-Installationsstammverzeichnis>/usr/server/wxswebapp/server.xml file

<server description="WebSphere eXtreme Scale enabled Web Server">

<featureManager>
<feature>eXtremeScale.webApp-1.1</feature>
</featureManager>

<com.ibm.ws.xs.webapp.config />
</server>
```

8.6+ Feature "WebGrid"

Ein Liberty-Profilserver kann ein Datengrid hosten, in dem Daten für Anwendungen zwischengespeichert werden, um HTTP-Sitzungsdaten für die Fehlertoleranz zu replizieren.

Datei *<WLP-Installationsstammverzeichnis>*/usr/server/wxswebgrid/server.xml

```
<WLP-Installationsstammverzeichnis>/usr/server/wxswebgrid/server.xml file

<server description="WebSphere eXtreme Scale enabled Web Server">

<featureManager>
<feature>eXtremeScale.webGrid-1.1</feature>
</featureManager>

<com.ibm.ws.xs.webgrid.config />
</server>
```

8.6+ Feature "Dynamischer Cache"

Ein Liberty-Profilserver kann ein Datengrid hosten, in dem Daten für Anwendungen zwischengespeichert werden, in denen der dynamische Cache aktiviert ist.

Datei *<WLP-Installationsstammverzeichnis>*/usr/server/wxsweb/server.xml

```
<server description="WebSphere eXtreme Scale enabled Web Server">

  <featureManager>
    <feature>eXtremeScale.dynacacheGrid-1.1</feature>
  </featureManager>

  <com.ibm.ws.xs.xsDynacacheGrid.config />
</server>
```

8.6+ Feature "JPA"

Verwenden Sie das Feature "Java Persistence API (JPA)" für Ihre Anwendungen, die JPA im Liberty-Profil verwenden.

Datei *<WLP-Installationsstammverzeichnis>*/usr/server/wxsjpa/server.xml

```
<WLP-Installationsstammverzeichnis>/usr/server/wxsjpa/server.xml file

<server description="WebSphere eXtreme Scale enabled Web Server">

<featureManager>
<feature>eXtremeScale.jpa-1.1</feature>
</featureManager>

<com.ibm.ws.xs.jpa.config />
</server>
```

8.6+ Feature "REST"

Verwenden Sie das REST-Gateway (Representational State Transfer), um auf einfache Datengrids zuzugreifen, die von einem Verbund im Liberty-Profil gehostet werden.

Datei *<WLP-Installationsstammverzeichnis>*/usr/server/wxsrest/server.xml

```
<WLP-Installationsstammverzeichnis>/usr/server/wxsrest/server.xml file

<server description="WebSphere eXtreme Scale enabled Web Server">

<featureManager>
<feature>eXtremeScale.rest-1.1</feature>
</featureManager>

<com.ibm.ws.xs.rest.config />
</server>
```

JPA-L2-Cache-Plug-in

Java

WebSphere eXtreme Scale enthält Cache-Plug-ins der Stufe 2 (L2) für die JPA-Provider OpenJPA und Hibernate. Wenn Sie eines dieser Plug-ins verwenden, verwendet Ihre Anwendung die JPA-API. Es wird ein Datengrid zwischen Anwendung und Datenbank eingeführt, das die Antwortzeiten verbessert.

Die Verwendung von eXtreme Scale als L2-Cache-Provider erhöht die Leistung beim Lesen und Abfragen von Daten und reduziert die Last der Datenbank. WebSphere eXtreme Scale bietet im Vergleich mit integrierten Cacheimplementierungen verschiedene Vorteile, weil der Cache automatisch in allen Prozessen repliziert wird. Wenn ein Client einen Wert zwischenspeichert, können alle anderen Clients den zwischengespeicherten Wert, der sich lokal im Speicher befindet, verwenden.

Sie können die Topologie und die Eigenschaften für den L2-Cache-Provider in der Datei `persistence.xml` konfigurieren. Weitere Informationen zum Konfigurieren dieser Eigenschaften finden Sie unter JPA-Cachekonfigurationseigenschaften für Hibernate 4.0.

Tipp: Das JPA-L2-Cache-Plug-in erfordert eine Anwendung, die die JPA-APIs verwendet. Wenn Sie APIs von WebSphere eXtreme Scale für den Zugriff auf eine JPA-Datenquelle verwenden möchten, verwenden Sie den JPA-Loader. Weitere Informationen finden Sie unter „JPA-Loader” auf Seite 75.

Hinweise zur JPA-L2-Cachetopologie

Die folgenden Faktoren haben Auswirkungen auf den zu konfigurierenden Topologietyp:

1. **Wie viele Daten werden schätzungsweise zwischengespeichert?**
   - Wenn die Daten in einen einzigen JVM-Heapspeicher passen, verwenden Sie „Integrierte Topologie” auf Seite 44 oder „Domäneninterne Topologie” auf Seite 43.
   - Wenn die Daten nicht in einen einzigen JVM-Heapspeicher passen, verwenden Sie „Integrierte, partitionierte Topologie” auf Seite 45 oder „Ferne Topologie” auf Seite 47.
2. **Welches Verhältnis zwischen Lese- und Schreiboperationen erwarten Sie?**

   Das Verhältnis zwischen Lese- und Schreiboperationen wirkt sich auf die Leistung des L2-Caches aus. Jede Topologie verarbeitet Lese- und Schreiboperationen anders.
   - „Integrierte Topologie” auf Seite 44: lokale lesen, fern schreiben
   - „Domäneninterne Topologie” auf Seite 43: lokal lesen, lokal schreiben

- „Integrierte, partitionierte Topologie" auf Seite 45: Partitioniert: fern lesen, fern schreiben
- „Ferne Topologie" auf Seite 47: fern lesen, fern schreiben

Anwendungen, die größtenteils schreibgeschützt sind, sollten, sofern möglich, integrierte und domäneninterne Topologien verwenden. Anwendungen, die mehr Schreiboperationen durchführen, sollten domäneninterne Topologien verwenden.

3. **Wie ist der Prozentsatz abgefragter Daten im Vergleich zum Prozentsatz anhand eines Schlüssels gefundener Daten?**

   Wenn der JPA-Abfragecache aktiviert ist, wird er von Abfrageoperationen genutzt. Aktivieren Sie den JPA-Abfragecache nur für Anwendungen mit einem hohen Lese/Schreib-Verhältnis, z. B., wenn Stand der Leseoperationen 99 % erreicht. Wenn Sie den JPA-Abfragecache verwenden, müssen Sie den „Integrierte Topologie" auf Seite 44 oder den „Domäneninterne Topologie" verwenden.

   Die Find-by-key-Operation (Suchen nach Schlüsseln) ruft eine Zielentität ab, wenn die Zielentität keine Beziehung hat. Wenn die Zielentität Beziehungen mit dem Abruftyp EAGER hat, werden diese Beziehungen zusammen mit der Zielentität abgerufen. Im JPA-Datencache verursacht der Abruf dieser Beziehungen einige wenige Cachetreffer, um alle Beziehungsdaten abzurufen.

4. **Welcher Veraltungsstand der Daten wird toleriert?**

   In einem System mit wenigen JVMs treten Latenzzeiten bei Schreiboperationen während der Datenreplikation auf. Das Ziel des Caches ist die Verwaltung einer synchronisierten Datenansicht in allen JVMs. Wenn Sie die domäneninterne Topologie verwenden, treten bei Schreiboperationen Verzögerungen während der Datenreplikation auf. Anwendungen, die diese Topologie verwenden, müssen veraltete Leseoperationen und gleichzeitige Schreiboperationen tolerieren, die Daten überschreiben.

Domäneninterne Topologie

Bei einer domäneninternen Topologie werden primäre Shards an jeden Container-Server in der Topologie verteilt. Diese primären Shards enthalten die vollständigen Daten für die Partition. Alle primären Shards können auch Schreiboperationen im Cache ausführen. Diese Konfiguration schaltet Engpässe in der integrierten Topologie aus, in der alle Schreiboperationen im Cache über ein einziges primäres Shard erfolgen.

In einer domäneninternen Topologie werden keine Replikat-Shards erstellt, selbst wenn Sie Replikate in Ihren Konfigurationsdateien definiert haben. Jedes redundante primäre Shard enthält eine vollständige Kopie der Daten, sodass jedes primäre Shard auch als Replikat-Shard betrachtet werden kann. Diese Konfiguration verwendet ähnlich wie in der integrierten Topologie eine einzige Partition.

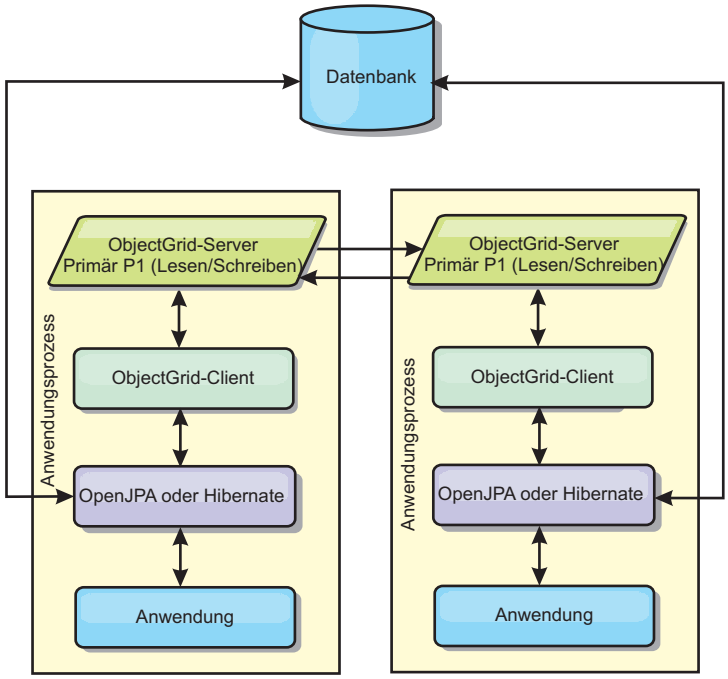


Abbildung 14. Domäneninterne JPA-Topologie

Zugehörige JPA-Cachekonfigurationseigenschaften für die domäneninterne Topologie:

```
ObjectGridName=ObjectGrid-Name,ObjectGridType=EMBEDDED,PlacementScope=CONTAINER_SCOPE,PlacementScopeTopology=HUB | RING
```

Vorteile:

- Lese- und Aktualisierungsoperationen im Cache sind lokal.
- Die Konfiguration ist einfach.

Einschränkungen:

- Diese Topologie eignet sich optimal, wenn die Container-Server alle Partitionsdaten enthalten.
- Replikat-Shards werden, selbst wenn sie konfiguriert sind, nie verteilt, weil jeder Container-Server ein primäres Shard hostet. Alle primären Shards werden auf den anderen primären Shards repliziert, sodass diese primären Shards zu gegenseitigen Replikaten werden.

Integrierte Topologie

Tipp: Für eine optimale Leistung sollten Sie eine domäneninterne Topologie in Erwägung ziehen.

Eine integrierte Topologie erstellt einen Container-Server im Prozessbereich jeder Anwendung. OpenJPA und Hibernate lesen die Speicherkopie des Caches direkt und schreiben in alle anderen Kopien. Sie können die Schreibleistung durch den Einsatz asynchroner Replikation verbessern. Diese Standardtopologie liefert die beste Leistung, wenn die zwischengespeicherte Datenmenge in einen einzigen Prozess passt. Bei einer integrierten Topologie erstellen Sie eine einzige Partition für die Daten.

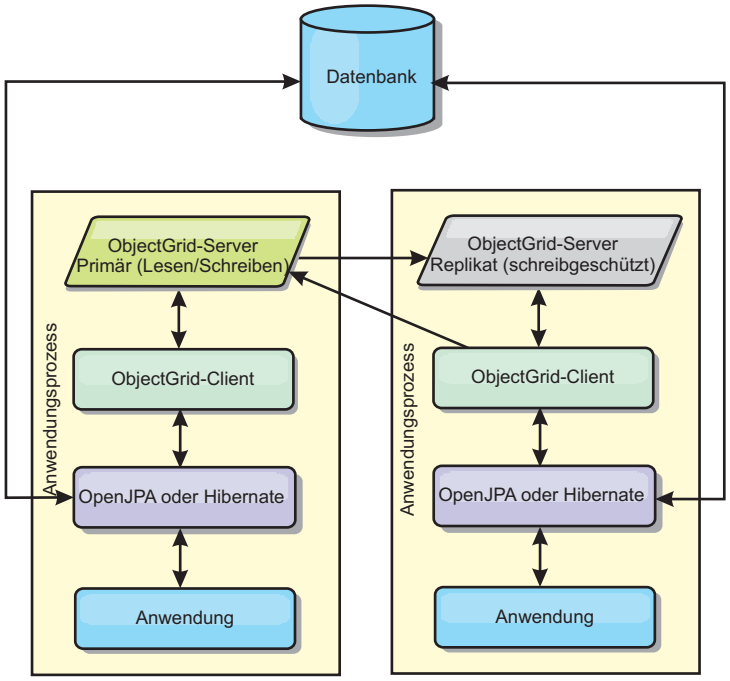


Abbildung 15. Integrierte JPA-Topologie

Zugehörige JPA-Cachekonfigurationseigenschaften für die integrierte Topologie:

```
ObjectGridName=ObjectGrid-Name,ObjectGridType=EMBEDDED,MaxNumberOfReplicas=Anzahl_der_Replikate,ReplicaMode=SYNC | ASYNC | NONE
```

Vorteile:

- Alle Leseoperationen im Cache sind schnelle lokale Zugriffe.
- Die Konfiguration ist einfach.

Einschränkungen:

- Das Datenvolumen ist auf die Größe des Prozesses beschränkt.
- Alle Cacheaktualisierungen werden über ein einziges primäres Shard gesendet, woraufhin ein Engpass entsteht.

Integrierte, partitionierte Topologie

Tipp: Für eine optimale Leistung sollten Sie eine domäneninterne Topologie in Erwägung ziehen.

Vorsicht:
Verwenden Sie den JPA-Abfragecache nicht für eine integrierte partitionierte Topologie. Im Abfragecache werden Abfrageergebnisse gespeichert, die eine Sammlung von Entitätsschlüsseln sind. Der Abfragecache rurft alle Entitätsdaten aus dem Datencache ab. Da der Datencache auf mehrere Prozesse verteilt ist, können diese zusätzlichen Aufrufe die Vorteile des L2-Caches aufheben.

Wenn die zwischengespeicherten Daten nicht in einen einzigen Prozess passen, können Sie die integrierte partitionierte Topologie verwenden. In dieser Topologie werden die Daten auf mehrere Prozesse verteilt. Die Daten werden so auf die primären Shards verteilt, dass jedes primäre Shard einen Teil der Daten enthält. Sie können diese Option auch verwenden, wenn die Latenzzeit der Datenbank hoch ist.

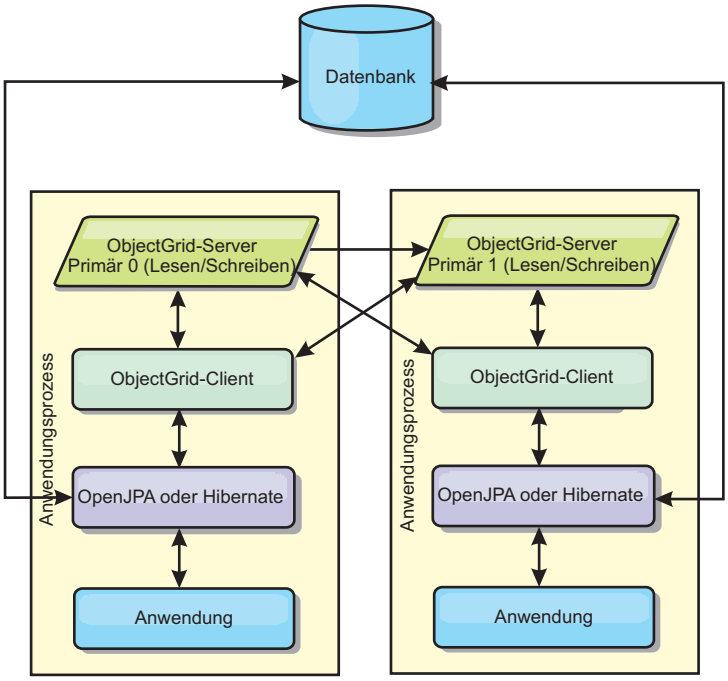


Abbildung 16. Integrierte, partitionierte JPA-Topologie

Zugehörige JPA-Cachekonfigurationseigenschaften für die integrierte partitionierte Topologie:

```
ObjectGridName=ObjectGrid-Name,ObjectGridType=EMBEDDED_PARTITION,ReplicaMode=SYNC | ASYNC | NONE,
NumberOfPartitions=Anzahl_Partitionen,ReplicaReadEnabled=TRUE | FALSE
```

Vorteile:

- Es können große Datenvolumen gespeichert werden.
- Die Konfiguration ist einfach.
- Cacheaktualisierungen werden auf mehrere Prozesse verteilt.

Einschränkungen:

- Die meisten Lese- und Aktualisierungsoperationen im Cache werden über Fernzugriff durchgeführt.

Um beispielsweise 10 GB Daten mit maximal 1 GB pro JVM zu speichern, sind zehn Java Virtual Machines erforderlich. Die Anzahl der Partitionen muss daher auf mindestens 10 gesetzt werden. Im Idealfall wird die Anzahl der Partitionen auf eine Primzahl gesetzt, sodass in jedem Shard eine angemessene Speichermenge zugeteilt wird. Gewöhnlich entspricht der Wert der Einstellung "numberOfPartitions" der Anzahl der Java Virtual Machines. Bei dieser Einstellung enthält jede JVM eine Partition. Wenn Sie die Replikation aktivieren, müssen Sie die Anzahl der Java Virtual Machines im System erhöhen. Andernfalls wird in jeder JVM zusätzlich eine Replikatpartition gespeichert, die genauso viel Speicher belegt wie eine primäre Partition.

Lesen Sie die Informationen zur Berechnung der Speicherkapazität und der Partitionsanzahl in der Veröffentlichung *Verwaltung*, um die Leistung der von Ihnen ausgewählten Konfiguration zu maximieren.

In einem System mit vier Java Virtual Machines und einem numberOfPartitions-Wert von 4 beispielsweise enthält jede JVM eine primäre Partition. Bei einer Lese-

operation besteht eine Chance von 25 %, dass die Daten aus einer lokal verfügbaren Partition abgerufen werden, was im Vergleich mit dem Abruf der Daten aus einer fernen JVM wesentlich schneller ist. Wenn eine Leseoperation, z. B. eine Abfrage, eine Sammlung von Daten abrufen muss, die gleichmäßig auf vier Partitionen verteilt sind, sind 75 % der Aufrufe ferne und 25 % der Aufrufe lokale Aufrufe. Wenn die Einstellung "ReplicaMode" auf `SYNC` oder `ASYNC` und die Einstellung "ReplicaReadEnabled" auf `true` gesetzt wird, werden vier Relikatpartitionen erstellt und auf vier Java Virtual Machines verteilt. Jede JVM enthält eine primäre Partition und eine Replikatpartition. Die Chance, dass die Leseoperation lokal ausgeführt wird, erhöht sich auf 50 %. Die Leseoperation, die eine Sammlung von Daten abrufen muss, die gleichmäßig auf vier Partitionen verteilt sind, hat 50 % ferne Aufrufe und 50% lokale Aufrufe. Lokale Aufrufe sind wesentlich schneller als ferne Aufrufe. Mit jedem fernen Aufruf nimmt die Leistung ab.

Ferne Topologie

Vorsicht:
Verwenden Sie den JPA-Abfragecache nicht für eine ferne Topologie. Im Abfragecache werden Abfrageergebnisse gespeichert, die eine Sammlung von Entitätsschlüsseln sind. Der Abfragecache rurft alle Entitätsdaten aus dem Datencache ab. Da der Datencache fern ist, können diese zusätzlichen Aufrufe die Vorteile des L2-Caches aufheben.

Tipp: Für eine optimale Leistung sollten Sie eine domäneninterne Topologie in Erwägung ziehen.

In einer fernen Topologie werden alle zwischengespeicherten Daten in einem oder mehreren gesonderten Prozessen gespeichert, was die Speicherbelegung der Anwendungsprozesse verringert. Sie können Ihre Daten auf unterschiedliche Prozesse verteilen, indem Sie ein partitioniertes, repliziertes eXtreme-Scale-Datengrid implementieren. Im Gegensatz zu den integrierten und integrierten partitionierten Konfigurationen, die in den vorherigen Abschnitten beschrieben wurden, müssen Sie ein fernes Datengrid unabhängig von der Anwendung und vom JPA-Provider verwalten.

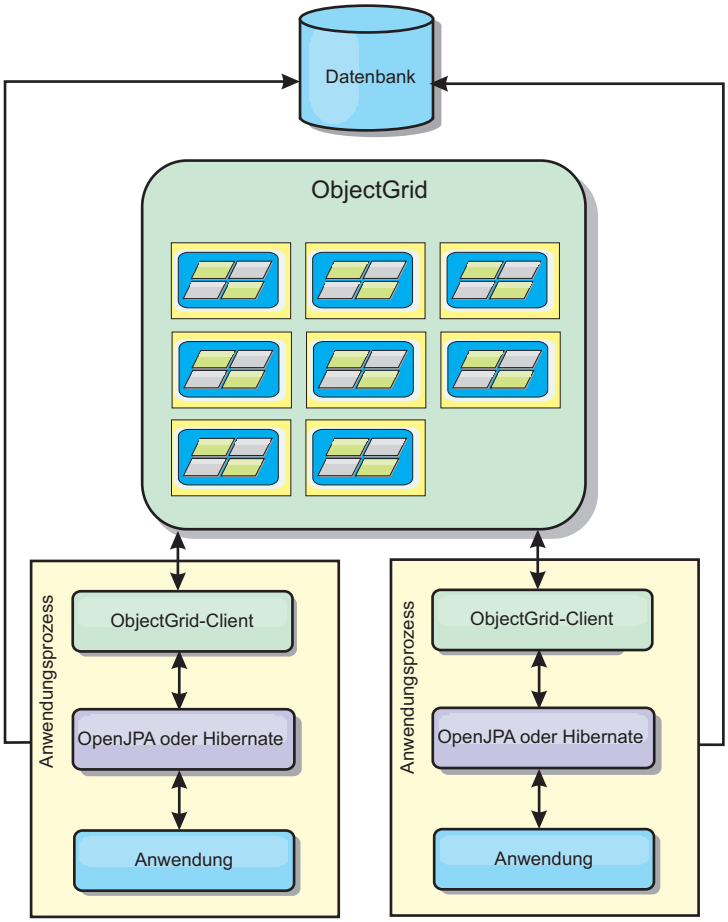


Abbildung 17. Ferne JPA-Topologie

Zugehörige JPA-Cachekonfigurationseigenschaften für die ferne Topologie:
`ObjectGridName=`*`ObjectGrid-Name`*`,ObjectGridType=REMOTE`

Der ObjectGrid-Typ REMOTE erfordert keine Eigenschaftseinstellungen, weil das ObjectGrid und die Implementierungsrichtlinie gesondert von der JPA-Anwendung definiert werden. Das JPA-Cache-Plug-in stellt über Fernzugriff eine Verbindung zu einem vorhandenen fernen ObjectGrid her.

Da alle Interaktionen mit dem ObjectGrid über Fernzugriff erfolgen, hat diese Topologie die geringste Leistung von allen ObjectGrid-Typen.

Vorteile:

- Es können große Datenvolumen gespeichert werden.
- Der Anwendungsprozess ist frei von zwischengespeicherten Daten.
- Cacheaktualisierungen werden auf mehrere Prozesse verteilt.
- Flexible Konfigurationsoptionen

Einschränkungen:

- Alle Lese- und Aktualisierungsoperationen im Cache werden über Fernzugriff durchgeführt.

Verwaltung von HTTP-Sitzungen

Der mit WebSphere eXtreme Scale bereitgestellte Sitzungsreplikationsmanager kann mit dem Standardsitzungsmanager in WebSphere Application Server zusammenarbeiten. Die Sitzungsdaten eines Prozesses werden in einem anderen Prozess repliziert, um die hohe Verfügbarkeit der Benutzersitzungsdaten zu unterstützen.

Features

Der Sitzungsmanager wurde so konzipiert, dass er in jedem Container der Java Platform, Enterprise Edition Version 5 ausgeführt werden kann. Da der Sitzungsmanager keine Abhängigkeiten von WebSphere-APIs aufweist, kann er verschiedene Versionen von WebSphere Application Server sowie Anwendungsserverumgebungen anderer Anbieter unterstützen.

Der HTTP-Sitzungsmanager stellt Sitzungsreplikationsfunktionen für eine zugeordnete Anwendung bereit. Der Sitzungsreplikationsmanager arbeitet mit dem Sitzungsmanager für den Web-Container. Der Sitzungsmanager und der Web-Container erstellen gemeinsam HTTP-Sitzungen und verwalten die Lebenszyklen von HTTP-Sitzungen, die der Anwendung zugeordnet sind. Zu diesen Verwaltungsaktivitäten für den Lebenszyklus gehören das Invalidieren von Sitzungen auf der Basis eines Zeitlimits oder eines expliziten Servlet- oder JSP-Aufrufs (JavaServer Pages) und der Aufruf von Sitzungslistenern, die der Sitzung bzw. der Webanwendung zugeordnet sind. Der Sitzungsmanager speichert seine Sitzungen in einem vollständig replizierten, in Clustern zusammengefassten und partitionierten Datengrid. Die Verwendung des Sitzungsmanagers von WebSphere eXtreme Scale ermöglicht dem Sitzungsmanager, die Unterstützung für HTTP-Sitzungsübernahme bereitzustellen, wenn Anwendungsserver heruntergefahren oder unerwartet beendet werden. Der Sitzungsmanager kann auch in Umgebungen eingesetzt werden, die keine Affinität unterstützen, wenn die Affinität nicht durch eine Lastausgleichsfunktionsschicht erzwungen wird, die Anforderungen auf die Anwendungsserver verteilt.

Einsatzszenarien

Der Sitzungsmanager kann in den folgenden Szenarien verwendet werden:

- In Umgebungen, die Anwendungsserver verschiedener Versionen von WebSphere Application Server verwenden, z. B. in einem Migrationsszenario.
- In Implementierungen, die Anwendungsserver verschiedener Anbieter verwenden. Ein Beispiel hierfür sind Anwendungen, die unter Open-Source-Anwendungsservern entwickelt werden und dann in WebSphere Application Server eingesetzt werden. Ein weiteres Beispiel sind Anwendungen, die von der Bereitstellung auf die Produktion hochgestuft werden. Eine nahtlose Migration dieser Anwendungsserverversionen ist möglich, während alle HTTP-Sitzungen aktiv und bedient werden.
- In Umgebungen, die erfordern, dass der Benutzer Sitzungen mit höheren Servicequalitätsstufen persistent speichert. Die Sitzungsverfügbarkeit ist während eines Server-Failovers besser gewährleistet als mit den Standardservicequalitätsstufen von WebSphere Application Server.
- In einer Umgebung, in der die Sitzungsaffinität nicht garantiert werden kann, oder in Umgebungen, in denen die Affinität von einer anbieterspezifischen Lastausgleichsfunktion verwaltet wird. Mit einer anbieterspezifischen Lastausgleichsfunktion muss der Affinitätsmechanismus an diese Lastausgleichsfunktion angepasst werden.
- In einer Umgebung, in der die Verwaltung und Speichern von Sitzungen in einen externen Java-Prozess ausgelagert werden muss.

- In mehreren Zellen, in denen die Sitzungsübernahme zwischen den Zellen aktiviert werden muss.
- In mehreren Rechenzentren oder mehreren Zonen.

Funktionsweise des Sitzungsmanagers

Der Sitzungsreplikationsmanager verwendet einen Sitzungslistener, um auf die Änderungen von Sitzungsdaten zu warten. Der Sitzungsreplikationsmanager schreibt die Sitzungsdaten persistent in einer ObjectGrid-Instanz fest - lokal oder fern. Mit Hilfe der in WebSphere eXtreme Scale bereitgestellten Tools können Sie den Sitzungslistener und den Servlet-Filter jedem Webmodul in Ihrer Anwendung hinzufügen. Sie können diese Listener und Filter auch dem Webimplementierungsdeskriptor Ihrer Anwendung hinzufügen.

Dieser Sitzungsreplikationsmanager arbeitet mit dem Webcontainersitzungsmanager jedes Anbieters zusammen, um Sitzungsdaten in Java Virtual Machines zu replizieren. Wenn der ursprüngliche Server abstürzt, können Benutzer die Sitzungsdaten von anderen Servern abrufen.

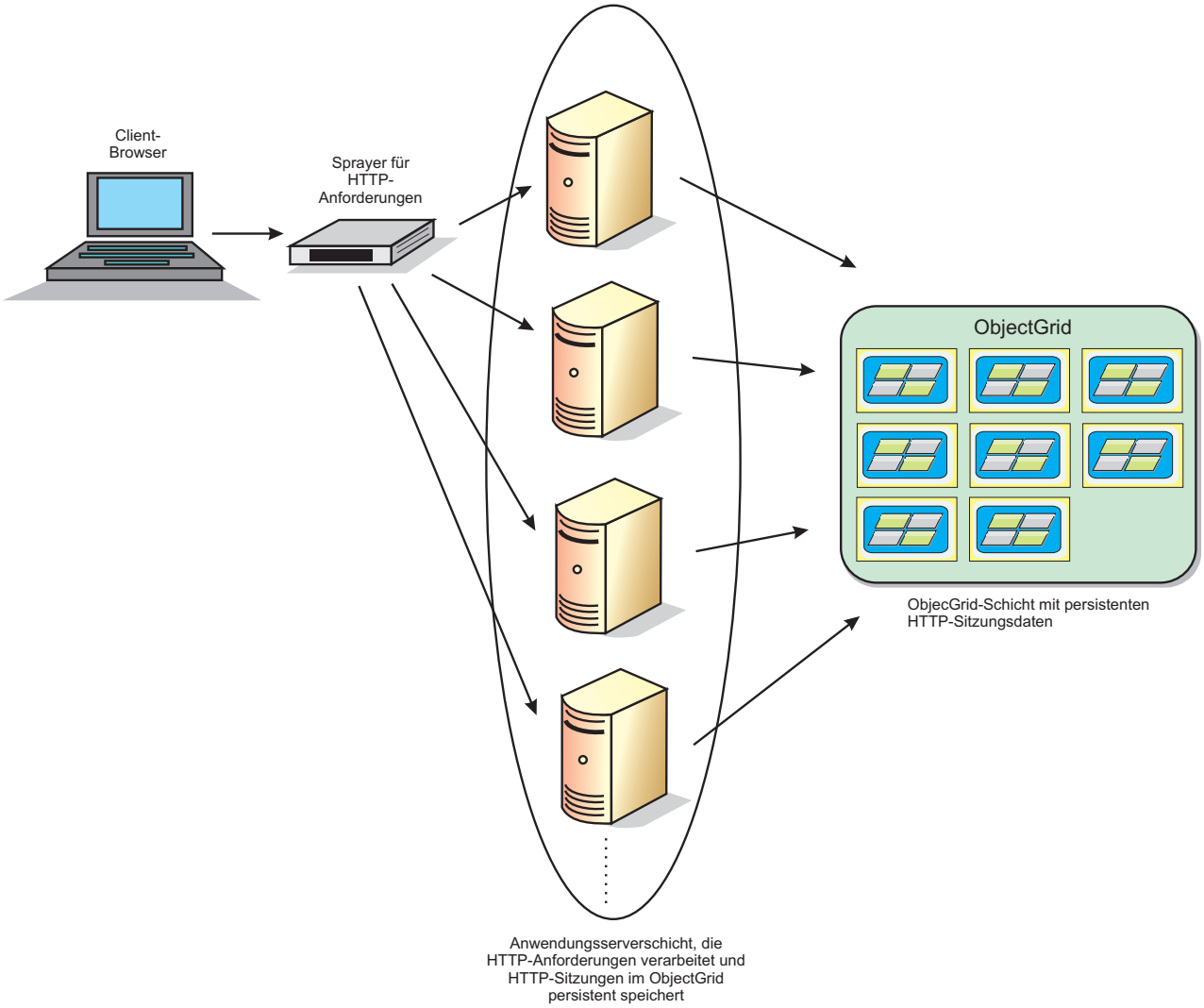


Abbildung 18. Topologie für die HTTP-Sitzungsverwaltung mit einer fernen Containerkonfiguration

Implementierungstopologien

Es gibt zwei dynamische Implementierungsszenarien für die Konfiguration des Sitzungsmanagers:

Integriertes Netz, das Container-Servern von eXtreme Scale zugeordnet ist
In diesem Szenario werden die Server von eXtreme Scale in denselben Prozessen wie die Servlets zusammengefasst. Der Sitzungsmanager kann direkt mit der lokalen ObjectGrid-Instanz kommunizieren, wodurch teure Verzögerungen bei der Netzübertragung vermieden werden. Dieses Szenario ist bevorzugt zu verwenden, wenn mit Affinität gearbeitet wird und die Leistung ein kritischer Faktor ist.

Fernes Netz, das Container-Servern von eXtreme Scale zugeordnet ist
In diesem Szenario werden die Server von eXtreme Scale in Prozessen ausgeführt, die von dem Prozess abgesondert sind, in dem die Servlets ausgeführt werden. Der Sitzungsmanager kommuniziert mit einem fernen eXtreme-Scale-Server-Grid. Dieses Szenario ist bevorzugt zu verwenden, wenn die Webcontainerschicht nicht den erforderlichen Speicher zum Speichern der Sitzungsdaten besitzt. Die Sitzungsdaten werden auf eine separate Schicht ausgelagert, was zu einer geringeren Speicherbelegung der Web-Container-Schicht führt. Die Latenzzeiten sind höher, weil sich die Daten an einem fernen Standort befinden.

Start generischer integrierter Container

eXtreme Scale startet einen integrierten ObjectGrid-Container automatisch in einem Anwendungsserverprozess, wenn der Webcontainer den Sitzungslistener oder den Servlet-Filter initialisiert, falls die Eigenschaft "objectGridType" den Wert `EMBEDDED` hat. Weitere Einzelheiten finden Sie im Abschnitt Initialisierungsparameter für den Servlet-Kontext.

Sie müssen eine Datei `ObjectGrid.xml` und eine Datei `objectGridDeployment.xml` nicht in die WAR-Datei der Webanwendung oder EAR-Datei packen. Die Standarddateien `ObjectGrid.xml` und `objectGridDeployment.xml` werden in die Produkt-JAR-Datei gepackt. Für verschiedene Webanwendungskontexte werden standardmäßig dynamische Maps erstellt. Statische eXtreme-Scale-Maps werden weiterhin unterstützt.

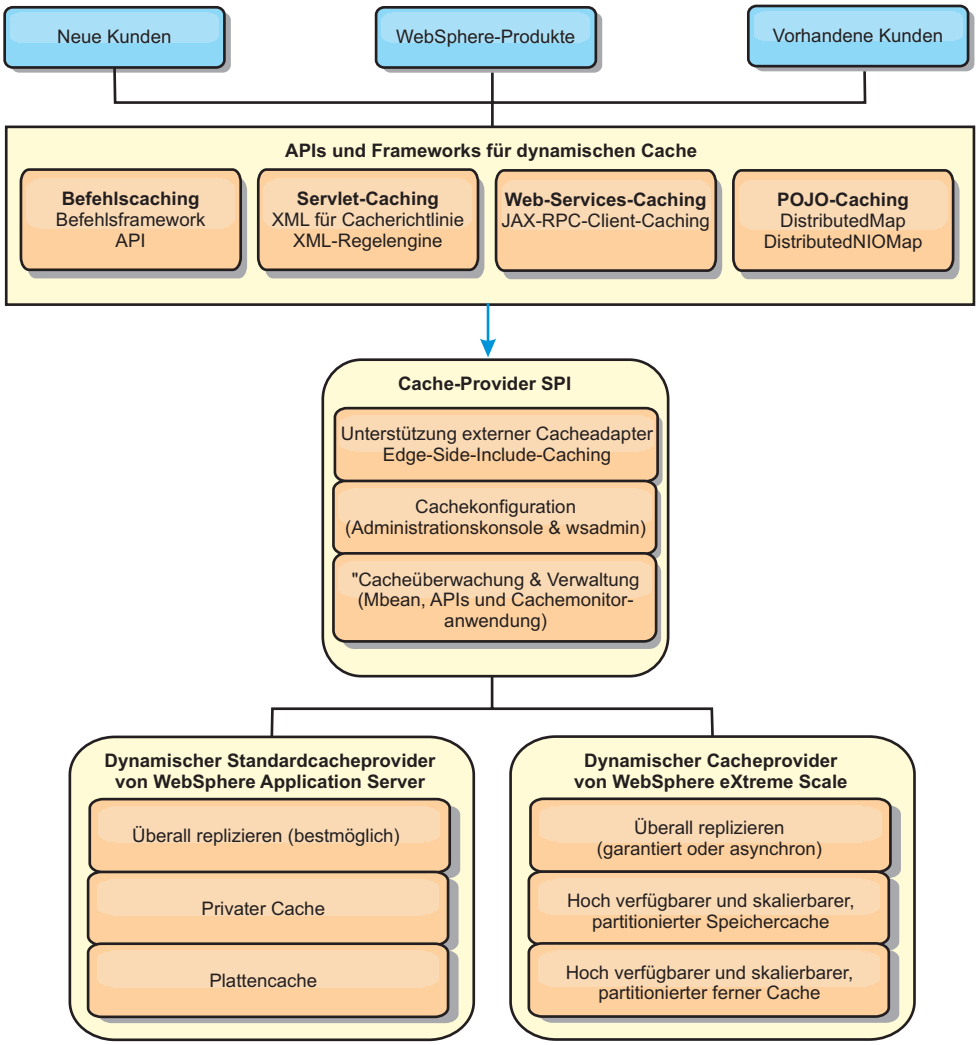
Dieser Ansatz für das Starten integrierter ObjectGrid-Container ist für jeden Typ von Anwendungsserver gültig. Die Ansätze, an denen eine Komponente von WebSphere Application Server oder WebSphere Application Server Community Edition GBean beteiligt ist, sind veraltet.

Übersicht über den dynamischen Cache-Provider

WebSphere Application Server stellt einen dynamischen Cache-Service bereit, der implementierten Java EE zur Verfügung steht. Dieser Service wird verwendet, um Daten wie die Ausgabe von Servlets, JSPs oder Befehlen sowie Objektdaten, die über das Programm in einer Unternehmensanwendung angegeben werden, mit DistributedMap-APIs zwischenzuspeichern.

Anfänglich war der einzige Serviceprovider für den dynamischen Cache-Service die in WebSphere Application Server integrierte Standardengine für dynamischen Cache. Jetzt können Kunden auch WebSphere eXtreme Scale als Cache-Provider für eine Cacheinstanz angeben. Wenn Sie diese Funktion konfigurieren, können Sie

Anwendungen, die den dynamischen Cache-Service verwenden, ermöglichen, die Features und Leistungsfunktionen von WebSphere eXtreme Scale zu verwenden.



Sie können den dynamischen Cache-Provider, wie im Abschnitt Standardinstanz des dynamischen Caches (baseCache) konfigurieren beschrieben, installieren und konfigurieren.

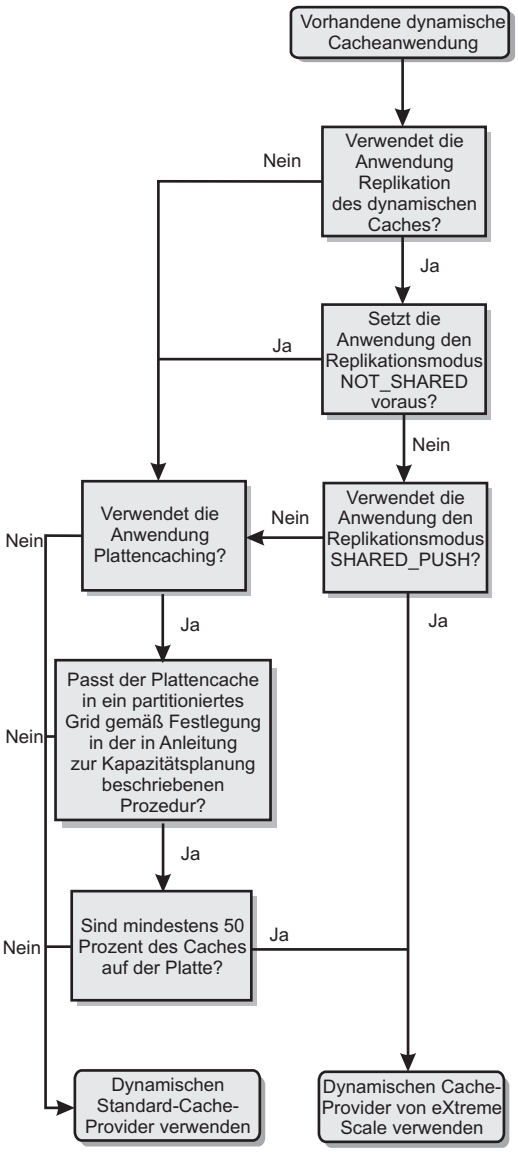
Einsatz von WebSphere eXtreme Scale festlegen

Die verfügbaren Features in WebSphere eXtreme Scale erweitern die verteilten Funktionen des dynamischen Cache-Service weit über das hinaus, was der Standardprovider für dynamischen Cache und der Datenreplikationsservice bieten. Mit eXtreme Scale können Sie Caches erstellen, die wirklich auf mehrere Server verteilt, anstatt nur repliziert und unter den Servern synchronisiert werden. Außerdem sind die Caches von eXtreme Scale transaktionsorientiert und hoch verfügbar, wodurch sichergestellt wird, dass jeder Server denselben Inhalt für den dynamischen Cacheservice sieht. WebSphere eXtreme Scale bietet eine höhere Servicequalität für die Cachereplikation über den Datenreplikationsservice.

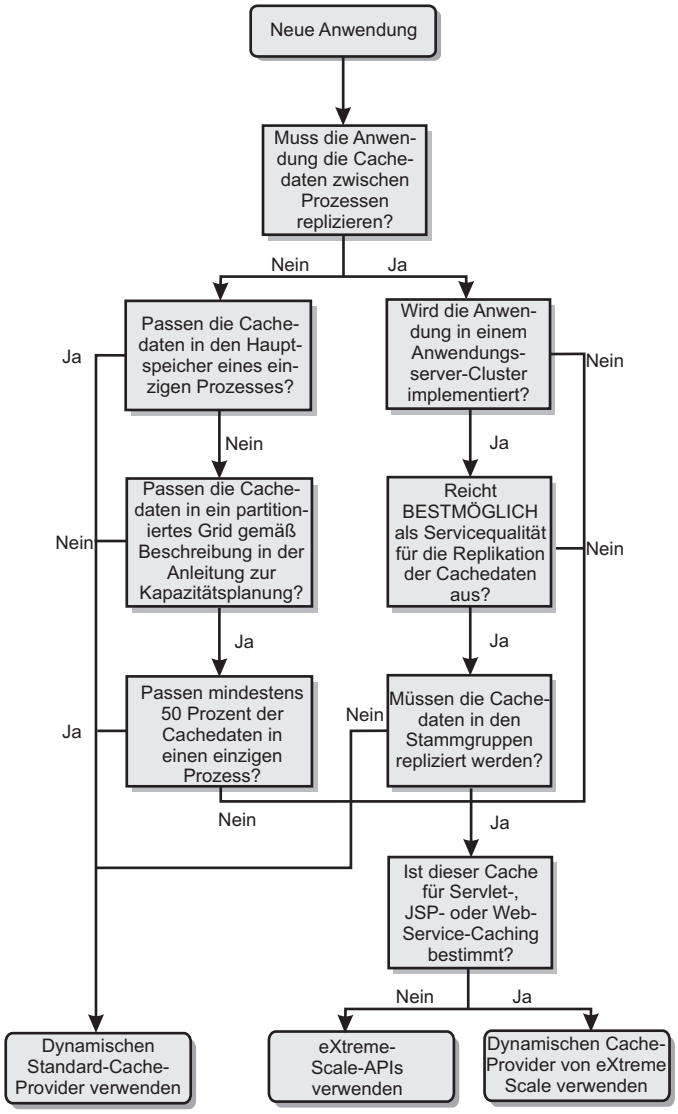
Diese Vorteile bedeuten jedoch nicht, dass der dynamische Cache-Provider von eXtreme Scale die richtige Wahl für jede Anwendung ist. Verwenden Sie die folgen-

den Entscheidungsbäume und Featurevergleichsmatrizes, um festzustellen, welche Technologie sich am besten für Ihre Anwendung eignet.

Entscheidungsbaum für die Migration vorhandener Anwendungen mit dynamischem Cache



Entscheidungsbaum für die Auswahl eines Cache-Providers für neue Anwendungen



Featurevergleich

Tabelle 1. Featurevergleich

Cache-Features	Standardprovider	eXtreme-Scale-Provider	eXtreme-Scale-API
Lokales Speicher-Caching	Ja	Über die Funktion für nahen Cache	Über die Funktion für nahen Cache
Verteiltes Caching	Über den Datenreplikationsservice	Ja	Ja
Linear skalierbar	Nein	Ja	Ja
Zuverlässige Replikation (synchron)	Nein	Ja	Ja
Plattenüberlauf	Ja	Nicht zutreffend	Nicht zutreffend

Tabelle 1. Featurevergleich (Forts.)

Cache-Features	Standardprovider	eXtreme-Scale-Provider	eXtreme-Scale-API
Bereinigung	LRU-/TTL-/Heapbasiert	LRU/TTL (pro Partition)	LRU/TTL (pro Partition)
Invalidierung	Ja	Ja	Ja
Beziehungen	Abhängigkeit/Vorlagen-ID-Beziehungen	Ja	Nein (keine anderen Beziehungen möglich)
Suchoperationen ohne Schlüssel	Nein	Nein	Über Abfrage und Index
Back-End-Integration	Nein	Nein	Über Ladeprogramme
Transaktionsorientiert	Nein	Ja	Ja
Schlüsselbasierter Speicher	Ja	Ja	Ja
Ereignisse und Listener	Ja	Nein	Ja
Integration von WebSphere Application Server	Nur einzelne Zelle	Mehrere Zellen	Zellenunabhängig
Unterstützung von Java Standard Edition	Nein	Ja	Ja
Überwachung und Statistiken	Ja	Ja	Ja
Sicherheit	Ja	Ja	Ja

Eine ausführlichere Beschreibung zur Funktionsweise der verteilten Caches in eXtreme Scale finden Sie in den Informationen zu Implementierungskonfigurationen in der Veröffentlichung *Verwaltung*.

Anmerkung: In einem verteilten eXtreme-Scale-Cache können nur Einträge gespeichert werden, bei denen sowohl der Schlüssel als auch der Wert die Schnittstelle "java.io.Serializable" implementieren.

Topologietypen

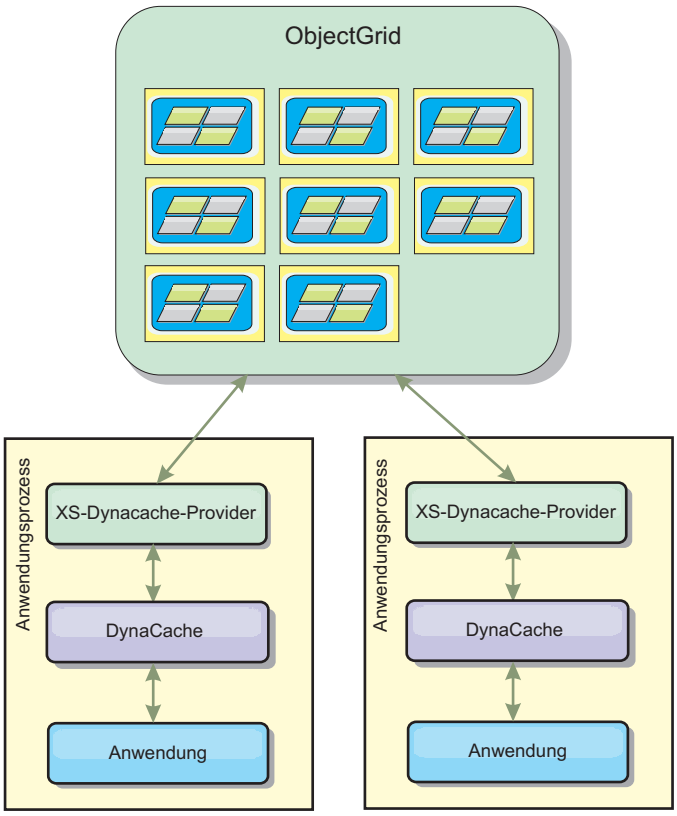
Veraltet: 8.6+ Die Topologietypen "local", "embedded" und "embedded-partitioned" sind veraltet.

Ein dynamischer Cache-Service, der mit eXtreme Scale als Provider erstellt wurde, kann in einer fernen Topologie implementiert werden.

Ferne Topologie

Bei der fernen Topologie ist kein Plattencache erforderlich. Alle Cachedaten werden außerhalb der Prozesse von WebSphere Application Server gespeichert. WebSphere eXtreme Scale unterstützt eigenständige Containerprozesse für Cachedaten. Diese Containerprozesse haben geringere Kosten als ein Prozess von WebSphere Application Server und sind auch nicht auf die Verwendung einer bestimmten Java Virtual Machine (JVM) beschränkt. Die Daten für einen dynamischen Cacheservice, auf

den mit einem 32-Bit-Prozess von WebSphere Application Server zugegriffen wird, könnte sich beispielsweise in einem eXtreme-Scale-Containerprozess befinden, der in einer 64-Bit-JVM ausgeführt wird. Dies ermöglicht Benutzern, die höhere Speicherkapazität von 64-Bit-Prozessen für das Caching zu nutzen, ohne das zusätzliche Kosten für den 64-Bit-Betrieb bei den Anwendungsserverprozessen anfallen. Die ferne Topologie wird in der folgenden Abbildung veranschaulicht:



Funktionale Unterschiede zwischen der dynamischen Cache-Engine und eXtreme Scale

Dem Benutzer sollten eigentlich keine funktionalen Unterschiede zwischen den beiden Caches auffallen, abgesehen davon, dass von WebSphere eXtreme Scale unterstützte Caches keine Auslagerung auf die Platte oder Statistiken und Operationen unterstützen, die sich auf die Größe des Caches im Speicher beziehen.

Die von den meisten Aufrufen der API "Dynamic Cache" zurückgegebenen Ergebnisse unterscheiden sich kaum, unabhängig davon, ob der Kunde den dynamischen Standard-Cache-Provider oder den Cache-Provider von eXtreme Scale verwendet. Für einige Operationen kann das Verhalten der dynamischen Cache-Engine nicht mit eXtreme Scale emuliert werden.

Statistiken zum dynamischen Cache

Statistische Daten für einen dynamischen Cache von WebSphere eXtreme Scale können mit den Überwachungstools von eXtreme Scale abgerufen werden. Weitere Informationen finden Sie im Abschnitt Überwachung.

MBean-Aufrufe

Der dynamische Cache-Provider von WebSphere eXtreme Scale unterstützt kein Platten-Caching. Alle MBean-Aufrufe, die sich auf Platten-Caching beziehen, funktionieren nicht.

Zuordnung einer Replikationsrichtlinie für den dynamischen Cache

Die ferne Topologie des dynamischen Cache-Providers von eXtreme Scale unterstützt eine Replikationsrichtlinie, die den Richtlinien SHARED_PULL und SHARED_PUSH_PULL am ehesten entspricht (und die Terminologie verwendet, die vom dynamischen Cache-Provider von WebSphere Application Server verwendet wird). In einem dynamischen Cache von eXtreme Scale ist der verteilte Status des Caches auf allen Servern vollkommen konsistent.

8.6+ Invalidierung mithilfe eines globalen Index

Sie können einen globalen Index verwenden, um die Invalidierungseffizienz in großen partitionierten Umgebungen zu verbessern, z. B. in Umgebungen mit mehr als 40 Partitionen. Ohne das globale Indexfeature müssen die Vorlage für dynamische Caches und die Verarbeitung der Abhängigkeitsinvalidierung Anforderungen ferner Agenten an alle Partitionen senden, was zu Leistungseinbußen führt. Wenn Sie einen globalen Index konfigurieren, werden Invalidierungsagenten nur an Partitionen gesendet, die Cacheeinträge enthalten, die sich auf die Vorlage oder die Abhängigkeits-ID beziehen. Die potenzielle Leistungsverbesserung ist in Umgebungen mit sehr vielen konfigurierten Partitionen höher. Sie können einen globalen Index mit den Indizes für Abhängigkeits-IDs und Vorlagen-IDs konfigurieren, die in den Beispiel-ObjectGrid-XML-Deskriptordateien für den dynamischen Cache verfügbar sind. Weitere Informationen finden Sie unter „Unternehmensdatengrid für dynamisches Caching in einer eigenständigen Umgebung konfigurieren” auf Seite 292.

Sicherheit

Wenn ein Cache in einer fernen Topologie ausgeführt wird, kann ein eigenständiger extreme-Scale-Client eine Verbindung zum Cache herstellen und den Inhalt der dynamischen Cacheinstanz beeinflussen. Deshalb ist es wichtig, dass sich Server von WebSphere eXtreme Scale, die die dynamischen Cacheinstanzen enthalten, in einem internen Netz hinter der so genannten Netz-DMZ (Demilitarized Zone) befinden.

Lesen Sie in der Dokumentation zu eXtreme Scale die Informationen zur „Übersicht über die Sicherheit” auf Seite 148, wenn SSL- oder Clientauthentifizierung erforderlich ist.

Naher Cache

Eine dynamische Cacheinstanz kann so konfiguriert werden, dass ein naher Cache erstellt wird, der sich lokal in der JVM des Anwendungsservers befindet und ein Subset der Einträge enthält, die in der fernen dynamischen Cacheinstanz enthalten sind. Sie können die nahe Cacheinstanz mit einer Datei namens "dynacache-nearCache-ObjectGrid.xml" konfigurieren. Weitere Informationen finden Sie unter „Unternehmensdatengrid für dynamisches Caching in einer eigenständigen Umgebung konfigurieren” auf Seite 292. Außerdem gibt es angepasste Eigenschaften für die Optimierung des nahen Caches. Weitere Informationen hierzu finden Sie unter

Angepasste Eigenschaften des dynamischen Caches.

Weitere Informationen

- Redbook zum dynamischen Cache
- Dokumentation zum dynamischen Cache
  - WebSphere Application Server 7.0
- Dokumentation zum Datenreplikationsservice
  - WebSphere Application Server 7.0

Datenbankintegration: Write-behind-, Inline- und Neben-Caching

WebSphere eXtreme Scale wird als Front-End für eine traditionelle Datenbank verwendet und macht Leseaktivitäten überflüssig, die normalerweise an die Datenbank übertragen werden. Ein kohärenter Cache kann direkt oder indirekt über einen ORM (Object Relational Mapper) mit einer Anwendung verwendet werden. Der kohärente Cache kann dann die Datenbank bzw. das Back-End von Leseaktivitäten entlasten. In einem geringfügig komplexeren Szenario, wie z. B. beim transaktionsorientierten Zugriff auf einen Datenbestand, in dem nur einige der Daten traditionelle Persistenzgarantien erfordern, können Sie Filter verwenden, um selbst die Schreibtransaktionen auszulagern.

Sie können WebSphere eXtreme Scale als hoch flexiblen speicherinternen Datenbankverarbeitungsbereich konfigurieren. WebSphere eXtreme Scale ist jedoch kein ORM. Das Produkt weiß nicht, woher die Daten im Datengrid stammen. Eine Anwendung oder ein ORM kann Daten in einem eXtreme-Scale-Server ablegen. Die Datenquelle ist dafür verantwortlich sicherzustellen, dass sie mit der Datenbank, aus der die Daten stammen, konsistent bleibt. Das bedeutet, dass eXtreme Scale Daten, die automatisch aus einer Datenbank extrahiert werden, nicht ungültig machen kann. Die Anwendung bzw. der Mapper muss diese Funktion bereitstellen und die Daten verwalten, die in eXtreme Scale gespeichert werden.

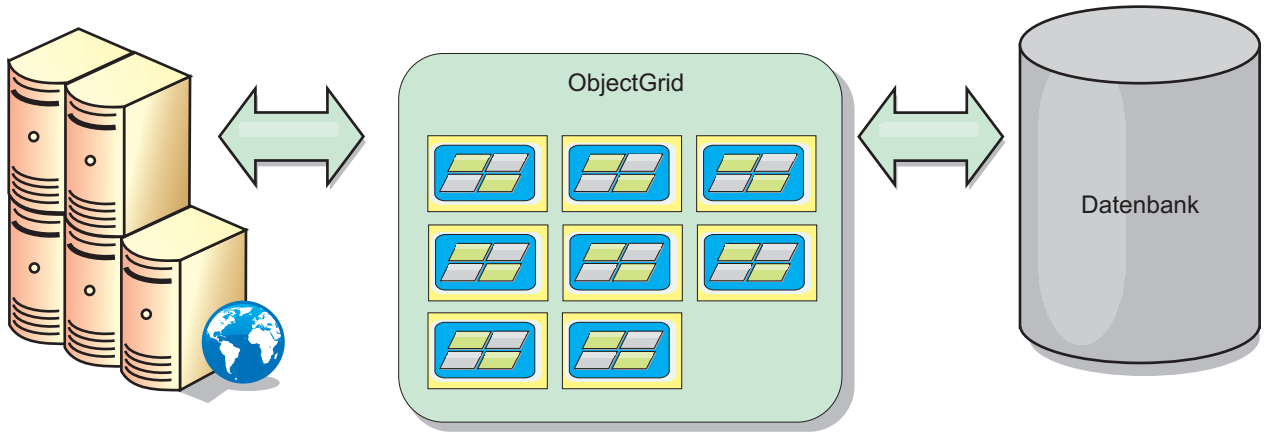


Abbildung 19. ObjectGrid als Datenbankpuffer

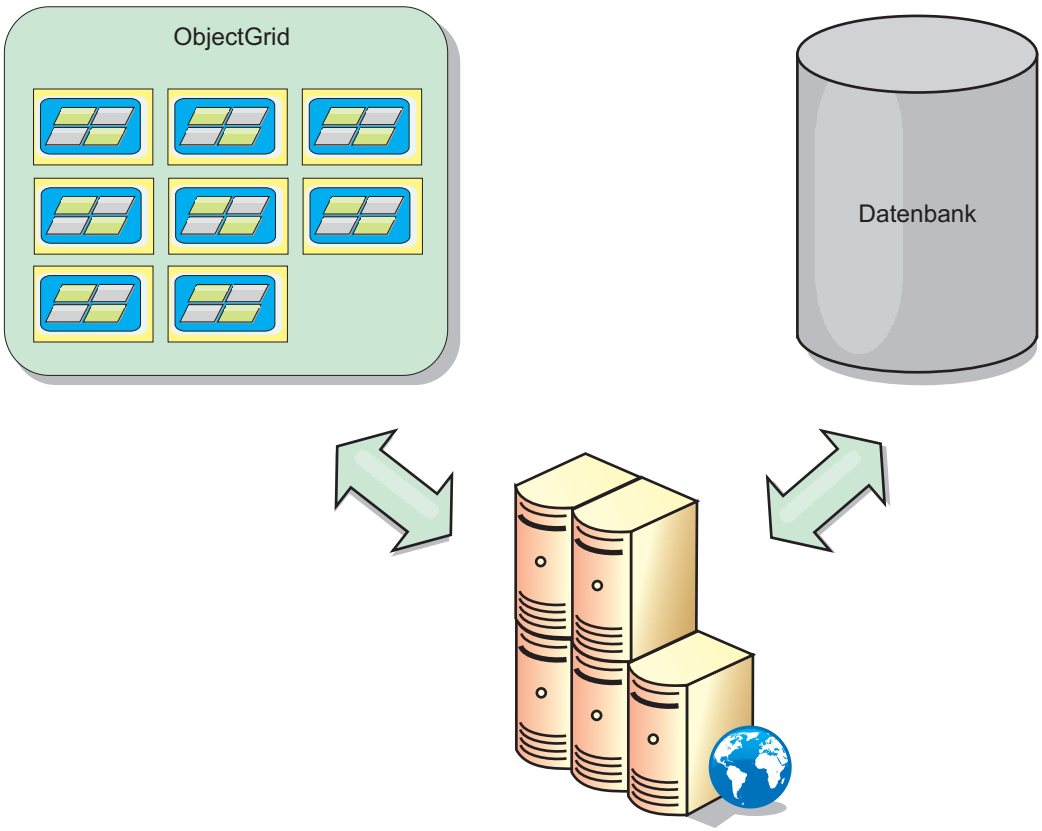


Abbildung 20. ObjectGrid als Nebencache

Teilcache und vollständiger Cache

WebSphere eXtreme Scale kann als Teilcache oder als vollständiger Cache eingesetzt werden. In einem Teilcache wird nur ein Teil der gesamten Daten gespeichert, wohingegen in einem vollständigen Cache alle Daten gespeichert werden. Ein Teilcache kann nach und nach bedarfsgesteuert gefüllt werden. Der Zugriff auf Teilcaches erfolgt gewöhnlich über Schlüssel (und nicht über Indizes oder Abfragen), da die Daten nur teilweise verfügbar sind.

Teilcache

Wenn ein Schlüssel nicht im Teilcache vorhanden ist oder wenn die Daten nicht verfügbar sind und ein Cachefehler auftritt, wird die nächste Schicht aufgerufen. Die Daten werden beispielsweise aus der Datenbank abgerufen und in die Cacheschicht des Datengrids eingefügt. Bei der Verwendung einer Abfrage oder eines Index wird nur auf die derzeit geladenen Werte zugegriffen, und die Anforderungen werden nicht an die anderen Schichten weitergeleitet.

Vollständiger Cache

Ein vollständiger Cache enthält alle erforderlichen Daten, und der Zugriff kann über Attribute ohne Schlüsselfunktion mit Indizes oder Abfragen erfolgen. Ein vollständiger Cache wird vorher mit Daten aus der Datenbank geladen, bevor die Anwendung versucht, auf die Daten zuzugreifen. Ein vollständiger Cache kann als Datenbankersatz dienen, nachdem die Daten geladen wurden. Da alle Daten ver-

fügbar sind, können Abfragen und Indizes verwendet werden, um Daten zu suchen und zusammenzufassen.

Nebencache

Wenn WebSphere eXtreme Scale als Nebencache verwendet wird, wird das Back-End für das Datengrid verwendet.

Nebencache

Sie können das Produkt als Nebencache für die Datenzugriffsschicht einer Anwendung konfigurieren. In diesem Szenario wird WebSphere eXtreme Scale verwendet, um Objekte temporär zu speichern, die normalerweise aus einer Back-End-Datenbank abgerufen werden. Anwendungen prüfen, ob das Datengrid die Daten enthält. Wenn die Daten im Datengrid enthalten sind, werden die Daten an den Aufrufenden zurückgegeben. Wenn die Daten nicht vorhanden sind, werden die Daten aus der Back-End-Datenbank abgerufen. Anschließend werden die Daten in das Datengrid eingefügt, damit die nächste Anforderung die zwischengespeicherte Kopie verwenden kann. Die folgende Abbildung veranschaulicht, wie WebSphere eXtreme Scale als Nebencache mit einer beliebigen Datenzugriffsschicht wie OpenJPA oder Hibernate verwendet werden kann.

Nebencache-Plug-ins für Hibernate und OpenJPA

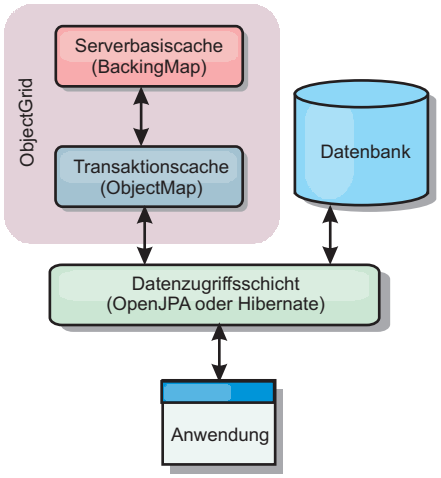


Abbildung 21. Nebencache

Cache-Plug-ins für OpenJPA und Hibernate sind in WebSphere eXtreme Scale enthalten. Damit können Sie das Produkt als automatischen Nebencache verwenden. Durch die Verwendung von WebSphere eXtreme Scale als Cache-Provider kann die Leistung beim Lesen und Abfragen von Daten verbessert und die Belastung der Datenbank verringert werden. WebSphere eXtreme Scale bietet im Vergleich mit integrierten Cacheimplementierungen verschiedene Vorteile, weil der Cache automatisch in allen Prozessen repliziert wird. Wenn ein Client einen Wert zwischenspeichert, können alle andere Clients den zwischengespeicherten Wert verwenden.

Inline-Cache

Sie können das Inline-Caching für ein Datenbank-Back-End oder als Nebencache für eine Datenbank konfigurieren. Beim Inline-Caching wird eXtreme Scale als pri-

märes Mittel für die Interaktion mit den Daten verwendet. Bei der Verwendung von eXtreme Scale als Inline-Cache interagiert die Anwendung über ein Loader-Plug-in mit dem Back-End.

Inline-Cache

Bei Verwendung als Inline-Cache interagiert WebSphere eXtreme Scale über ein Loader-Plug-in mit dem Back-End. Dieses Szenario kann den Datenzugriff vereinfachen, weil Anwendungen direkt auf die APIs von eXtreme Scale zugreifen können. Es werden verschiedene Caching-Szenarien in eXtreme Scale unterstützt, um sicherzustellen, dass die Daten im Cache und die Daten im Back-End synchronisiert sind. Die folgende Abbildung veranschaulicht, wie ein Inline-Cache mit der Anwendung und dem Back-End interagiert.

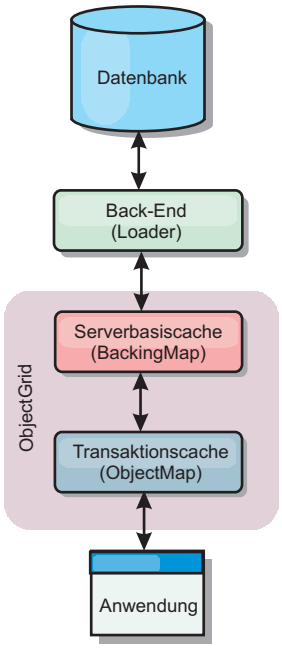


Abbildung 22. Inline-Cache

Die Option für Inline-Caching vereinfacht den Datenzugriff, weil sie Anwendungen den direkten Zugriff auf die eXtreme-Scale-APIs ermöglicht. WebSphere eXtreme Scale unterstützt mehrere Szenarien mit Inline-Caching:

- Read-through
- Write-through
- Write-behind

Szenario mit Read-through-Caching

Ein Read-through-Cache ist ein Teilcache, in den nach und nach Dateneinträge nach Schlüssel geladen werden, wenn diese angefordert werden. Dies geschieht, ohne dass der Aufrufende wissen muss, wie die Einträge geladen werden. Wenn die Daten nicht im eXtreme-Scale-Cache gefunden werden, ruft eXtreme Scale die fehlenden Daten vom Loader-Plug-in ab, das die Daten aus der Back-End-Datenbank lädt und in den Cache einfügt. Nachfolgende Anforderungen für denselben Datenschlüssel werden im Cache gefunden, bis der Eintrag gelöscht, ungültig gemacht oder durch Bereinigung entfernt wird.

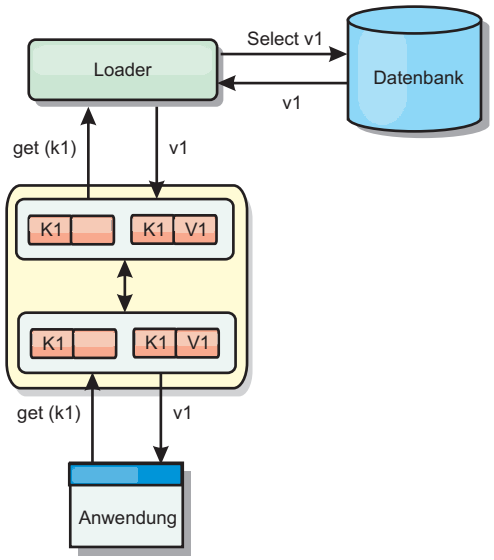


Abbildung 23. Read-through-Caching

Szenario mit Write-through-Caching

In einem Write-through-Cache (Durchschreibcache) erfolgt bei jedem Schreibvorgang in den Cache ein synchroner Schreibvorgang über den Loader in die Datenbank. Diese Methode gewährleistet die Konsistenz mit dem Back-End, verringert aber die Schreibleistung, weil die Datenbankoperation synchron erfolgt. Da der Cache und die Datenbank beide aktualisiert werden, werden bei nachfolgenden Leseoperationen dieselben Daten im Cache gefunden und Datenbankaufrufe vermieden. Ein Write-through-Cache wird häufig in Kombination mit einem Read-through-Cache verwendet.

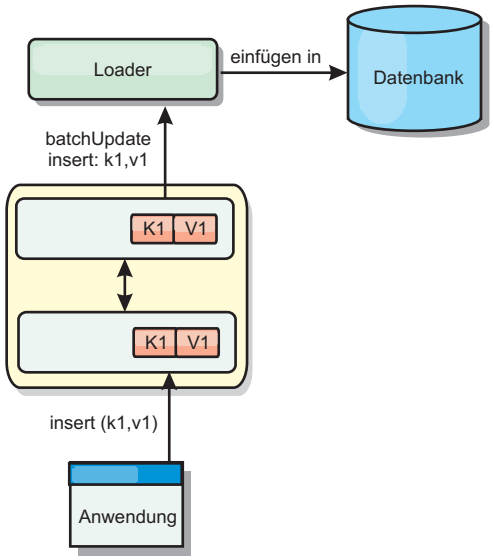


Abbildung 24. Write-through-Caching

Szenario mit Write-behind-Caching

Die Datenbanksynchronisation kann verbessert werden, indem Änderungen asynchron geschrieben werden. Dies wird als Write-behind- oder Write-back-Cache

(Rückschreibcache) bezeichnet. Änderungen, die normalerweise synchron in den Loader geschrieben werden, werden stattdessen in eXtreme Scale gepuffert und über einen Hintergrund-Thread in die Datenbank geschrieben. Die Schreibleistung wird erheblich verbessert, weil die Datenbankoperation aus der Clienttransaktion entfernt wird und die Schreibvorgänge in die Datenbank komprimiert werden können.

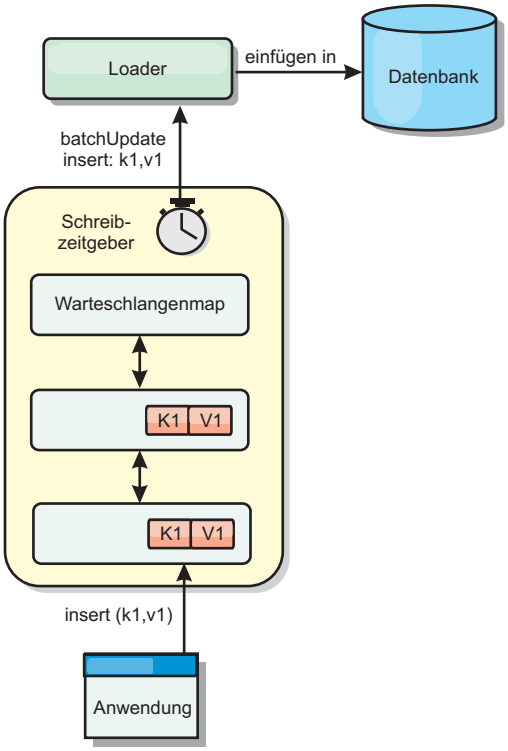


Abbildung 25. Write-behind-Caching

Write-behind-Caching

Java

Sie können Write-behind-Caching verwenden, um die Kosten für die Aktualisierung einer Datenbank, die Sie als Back-End verwenden, zu reduzieren.

Übersicht über das Write-behind-Caching

Beim Write-behind-Caching werden Aktualisierungen für das Loader-Plug-in asynchron in die Warteschlange eingereiht. Sie können die Leistung von Aktualisierungs-, Einfüge- und Entfernungsoperationen für die Map verbessern, indem Sie die eXtreme-Scale-Transaktion von der Datenbanktransaktion entkoppeln. Die asynchrone Aktualisierung wird nach einer zeitbasierten Verzögerung (z. B. fünf Minuten) oder einer eintragsbasierten Verzögerung (z. B. 1000 Einträge) durchgeführt.

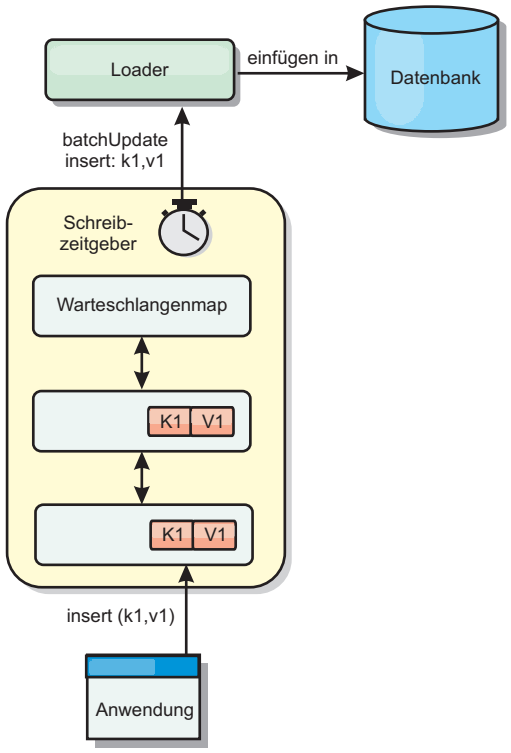


Abbildung 26. Write-behind-Caching

Bei der Write-behind-Konfiguration in einer BackingMap wird ein Thread zwischen dem Loader (Ladeprogramm) und der Map erstellt. Anschließend delegiert der Loader Datenanforderungen über den Thread gemäß den Konfigurationseinstellungen in der Methode "BackingMap.setWriteBehind". Wenn eine eXtreme-Scale-Transaktion einen Eintrag in einer Map einfügt, aktualisiert oder entfernt, wird ein LogElement-Objekt für jeden dieser Datensätze erstellt. Diese Elemente werden an den Write-behind-Loader gesendet und in eine spezielle ObjectMap, eine so genannte Warteschlangenmap, eingereiht. Jede BackingMap mit aktivierter Write-behind-Einstellung hat ihre eigenen Warteschlangenmaps. Ein Write-behind-Thread entfernt die in die Warteschlange eingereihten Daten aus den Warteschlangenmaps und überträgt sie mit Push in den echten Back-End-Loader.

Der Write-behind-Loader sendet nur LogElement-Objekte der Typen "insert" (Einfügen), "update" (Aktualisieren) und "delete" (Löschen) an den echten Loader. Alle anderen Typen von LogElement-Objekten, wie z. B. EVICT, werden ignoriert.

Die Write-behind-Unterstützung ist eine Erweiterung des Loader-Plug-ins, das Sie verwenden, um eXtreme Scale mit der Datenbank zu integrieren. Sehen Sie sich beispielsweise die Informationen zur Konfiguration eines JPA-Loaders im Abschnitt JPA-Loader konfigurieren an.

Vorteile

Das Aktivieren der Write-behind-Unterstützung hat die folgenden Vorteile:

- **Isolation von Back-End-Fehlern:** Durch das Write-behind-Caching können Back-End-Fehler isoliert werden. Wenn die Back-End-Datenbank ausfällt, werden Aktualisierungen in die Warteschlangenmap eingereiht. Die Anwendungen können

weiterhin Transaktionen an eXtreme Scale senden. Nach der Wiederherstellung des Back-Ends werden die Daten in der Warteschlangenmap mit Push an das Back-End übertragen.

- **Geringere Back-End-Last:** Der Write-behind-Loader fasst die Aktualisierungen auf Schlüsselbasis so zusammen, dass nur eine einzige zusammengefasste Aktualisierung pro Schlüssel in der Warteschlangenmap vorhanden ist. Bei dieser Zusammenfassung verringert sich die Anzahl der Aktualisierungen für die Back-End-Datenbank.
- **Verbesserte Transaktionsleistung:** Die Zeiten einzelner eXtreme-Scale-Transaktionen verringern sich, weil sie nicht auf die Synchronisation der Daten mit dem Back-End warten müssen.

Loader

Java

Mit einem Loader-Plug-in kann sich eine Daten-Grid-Map wie ein Speichercache für Daten verhalten, die gewöhnlich in einem persistenten Speicher auf demselben System oder einem anderen System gespeichert werden. Gewöhnlich wird eine Datenbank oder ein Dateisystem als persistenter Speicher verwendet. Es kann auch eine ferne Java Virtual Machine (JVM) als Datenquelle verwendet werden, was die Erstellung hubbasierter Caches mit eXtreme Scale ermöglicht. Ein Loader enthält die Logik für das Lesen aus einem und das Schreiben in einem persistenten Speicher.

Übersicht

Loader (Ladeprogramme) sind BackingMap-Plug-ins, die aufgerufen werden, wenn Änderungen an der BackingMap vorgenommen werden oder wenn die BackingMap eine Datenanforderung nicht bedienen kann (Cachefehler). Der Loader wird aufgerufen, wenn der Cache eine Anforderung für einen Schlüssel nicht bedienen kann. Er unterstützt Read-through-Funktionen und eine verzögerte Füllung des Caches. Ein Loader lässt außerdem Aktualisierungen in der Datenbank zu, wenn sich Cachewerte ändern. Alle Änderungen in einer Transaktion werden gruppiert, um die Anzahl der Datenbankinteraktionen zu minimieren. Zusammen mit dem Loader wird ein TransactionCallback-Plug-in verwendet, um die Abgrenzung der Back-End-Transaktion auszulösen. Die Verwendung dieses Plug-ins ist wichtig, wenn mehrere Maps an einer einzelnen Transaktion beteiligt sind oder wenn Transaktionsdaten ohne Festschreibung mit Flush in den Cache übertragen werden.

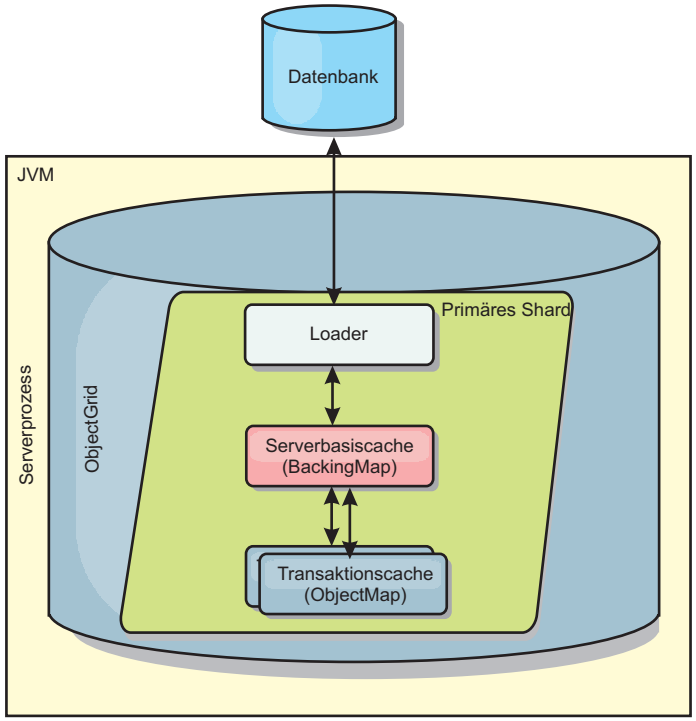


Abbildung 27. Loader

Der Loader kann auch überqualifizierte Aktualisierungen verwenden, um keine Datenbanksperren halten zu müssen. Anhand eines im Cachewert gespeicherten Versionsattributs kann der Loader das Vorher- und Nachher-Abbild des Werts erkennen, wenn dieser im Cache aktualisiert wird. Dieser Wert kann anschließend bei der Aktualisierung der Datenbank bzw. des Back-Ends verwendet werden, um sicherzustellen, dass die Daten nicht aktualisiert wurden. Ein Loader kann auch so konfiguriert werden, dass das Datengrid beim Start vorher geladen wird. Wenn mit Partitionierung gearbeitet wird, wird jeder Partition eine Loader-Instanz zugeordnet. Hat die Map "Company" beispielsweise zehn Partitionen, gibt es zehn Loader-Instanzen, eine für jede primäre Partition. Bei der Aktivierung des primären Shards für die Map wird die Methode "preloadMap" für den Loader synchron oder asynchron aufgerufen. Dies ermöglicht das automatische Laden von Daten aus dem Back-End in die Mappartition. Wenn die Methode synchron aufgerufen wird, werden alle Clienttransaktionen blockiert, um einen inkonsistenten Zugriff auf das Datengrid zu verhindern. Alternativ kann ein Client-Preloader zum Laden des vollständigen Datengrids verwendet werden.

Es gibt zwei integrierte Loader, die die Integration mit relationalen Datenbank-Back-Ends erheblich vereinfachen. Die JPA-Loader nutzen die ORM-Funktionen (Object-Relational Mapping, objektrelationale Abbildung) der OpenJPA- und Hibernate-Implementierungen der Spezifikation Java Persistence API (JPA). Weitere Informationen finden Sie unter „JPA-Loader” auf Seite 75.

Wenn Sie Loader in einer Konfiguration mit mehreren Rechenzentren verwenden, müssen Sie berücksichtigen, wie Revisionsinformationen und Cachekonsistenz zwischen den Datengrids verwaltet werden. Weitere Informationen finden Sie im Abschnitt „Hinweise zu Ladeprogrammen in einer Multimastertopologie” auf Seite 183.

Loader-Konfiguration

Wenn Sie der BackingMap-Konfiguration einen Loader hinzufügen möchten, können Sie die programmgesteuerte Konfiguration oder die XML-Konfiguration verwenden. Ein Loader steht mit einer BackingMap in folgender Beziehung.

- Eine BackingMap kann nur einen einzigen Loader haben.
- Eine Client-BackingMap (naher Cache) kann keinen Loader haben.
- Eine Loader-Definition kann auf mehrere BackingMaps angewendet werden, aber jede BackingMap hat eine eigene Loader-Instanz.

Vorheriges Laden von Daten und Vorbereitung

In vielen Szenarien, die die Verwendung eines Loaders (Ladeprogramms) beinhalten, können Sie Ihr Datengrid durch vorheriges Laden von Daten (Preload) vorbereiten.

Wenn das Grid als vollständiger Cache verwendet wird, muss das Datengrid alle Daten aufnehmen und geladen werden, bevor Clients eine Verbindung zum Grid herstellen können. Wenn Sie einen Teilcache verwenden, müssen Sie den Cache mit Daten vorbereiten (Aufwärmphase), sodass Clients sofortigen Zugriff auf die Daten haben, wenn sie eine Verbindung zum Grid herstellen.

Es gibt zwei Methoden für das vorherige Laden von Daten in das Datengrid: Verwendung eines Loader-Plug-ins (Ladeprogramm) oder Verwendung eines Client-Loaders. Diese beiden Methoden werden in den folgenden Abschnitten beschrieben.

Loader-Plug-in

Das Loader-Plug-in wird jeder Map zugeordnet und ist für die Synchronisation eines einzelnen primären Partitions-Shards mit der Datenbank zuständig. Die Methode "preloadMap" des Loader-Plug-ins wird automatisch aufgerufen, wenn ein Shard aktiviert wird. Wenn Sie beispielsweise 100 Partitionen haben, sind 100 Loader-Instanzen vorhanden, die jeweils die Daten für ihre Partition laden. Wenn die Loader-Instanzen synchron ausgeführt werden, werden alle Clients blockiert, bis das vorherige Laden der Daten (der so genannte Preload-Prozess) abgeschlossen ist.

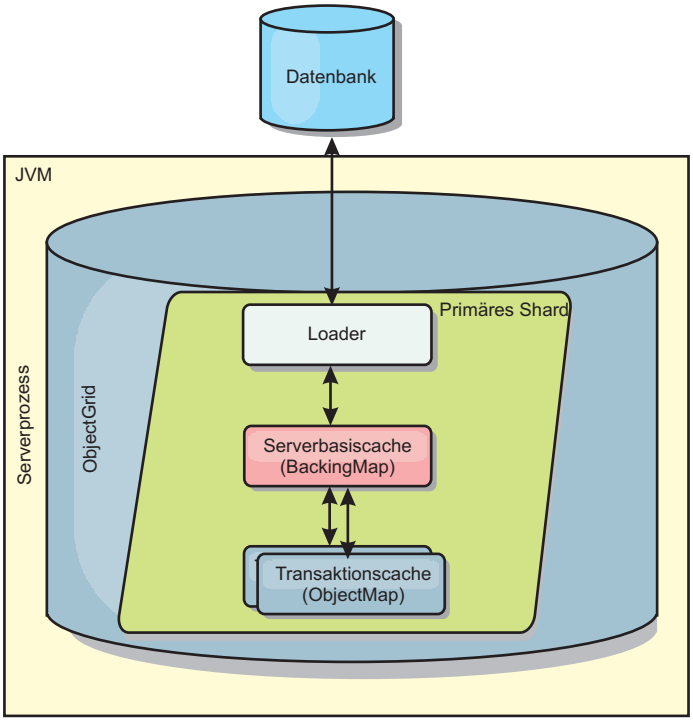


Abbildung 28. Loader-Plug-in

Client-Loader

Ein Client-Loader ist ein Muster für die Verwendung eines oder mehrerer Clients, um Daten in das Grid zu laden. Die Verwendung mehrerer Clients zum Laden von Griddaten kann effektiv sein, wenn das Partitionsschema nicht in der Datenbank gespeichert ist. Sie können Client-Loader manuell oder automatisch aufrufen, wenn das Datengrid gestartet wird. Client-Loader können optional die Schnittstelle "StateManager" verwenden, um den Status des Datengrids auf den Preload-Modus zu setzen, sodass Clients nicht auf das Grid zugreifen können, wenn das vorherige Laden der Daten in das Grid durchgeführt wird. WebSphere eXtreme Scale enthält einen JPA-basierten (Java Persistence API) Loader, den Sie verwenden können, um das Datengrid automatisch über die OpenJPA- oder Hibernate-JPA-Provider zu laden. Weitere Informationen zu Cache-Providern finden Sie unter „JPA-L2-Cache-Plug-in” auf Seite 42.

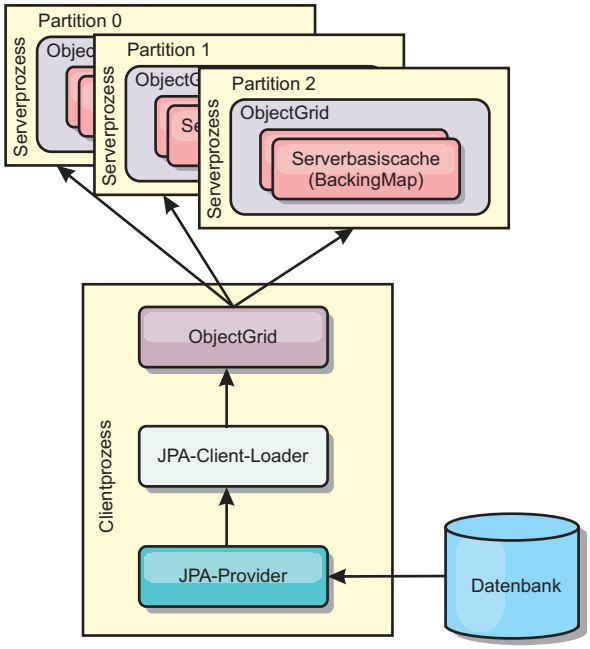


Abbildung 29. Client-Loader

Verfahren für die Datenbanksynchronisation

Wenn WebSphere eXtreme Scale als Cache verwendet wird, müssen Anwendungen so geschrieben werden, dass veraltete Daten toleriert werden, wenn die Datenbank unabhängig von einer eXtreme-Scale-Transaktion aktualisiert werden kann. Für den Einsatz als Verarbeitungsbereich für die synchronisierte speicherinterne Datenbank stellt eXtreme Scale mehrere Methoden für die konstante Aktualisierung des Caches bereit.

Verfahren für die Datenbanksynchronisation

Regelmäßige Aktualisierung

Der Cache kann mit Hilfe der zeitbasierten JPA-Datenbankaktualisierungskomponente (Java Persistence API) automatisch ungültig gemacht oder regelmäßig aktualisiert werden. Die Aktualisierungskomponente fragt die Datenbank in regelmäßigen Abständen über einen JPA-Provider nach Aktualisierungen oder Einfügungen ab, die seit der vorherigen Aktualisierung vorgenommen wurden. Alle gefundenen Änderungen werden automatisch ungültig gemacht oder aktualisiert, wenn ein Teilcache verwendet wird. Wenn ein vollständiger Cache verwendet wird, können die Einträge erkannt und in den Cache eingefügt werden. Es werden keine Einträge aus dem Cache entfernt.

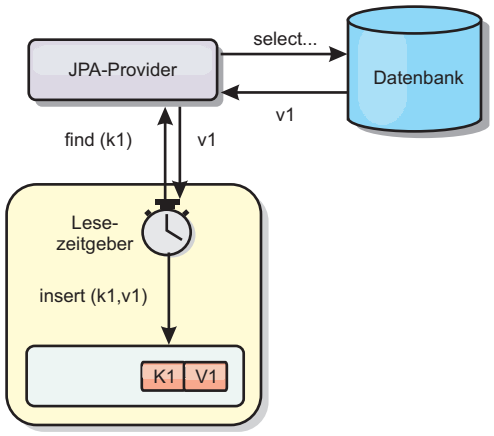


Abbildung 30. Regelmäßige Aktualisierung

Bereinigung

Teilcaches können Bereinigungsrichtlinien verwenden, um Daten ohne Beeinträchtigung der Datenbank automatisch aus dem Cache zu entfernen. Mit eXtreme Scale werden drei integrierte Richtlinien bereitgestellt: Lebensdauer (TTL, Time-to-Live), LRU (least recently used) und LFU (last frequently used). Alle drei Richtlinien können Daten aggressiver entfernen, wenn Speicherengpässe auftreten, indem die Option für speicherbasierte Bereinigung aktiviert wird.

Ereignisbasierte Invalidierung

Teilcaches und vollständige Caches können mit Hilfe eines Ereignisgenerators wie Java Message Service (JMS) ungültig gemacht oder aktualisiert werden. Die Invalidierung mit JMS kann manuell an jeden Prozess gebunden werden, der das Back-End über einen Datenbankauslöser aktualisiert. Es wird ein JMS-ObjectGridEvent-Listener-Plug-in in eXtreme Scale bereitgestellt, das Clients benachrichtigen kann, wenn Änderungen im Server-Cache vorgenommen wurden. Auf diese Weise kann das Zeitfenster, in dem der Client veraltete Daten sieht, verringert werden.

Programmgesteuerte Invalidierung

Die APIs von eXtreme Scale unterstützen die manuelle Interaktion zwischen nahem Cache und Server-Cache über die API-Methoden "Session.beginNoWriteThrough()", "ObjectMap.invalidate()" und "EntityManager.invalidate()". Wenn ein Client- oder Serverprozess einen Teil der Daten nicht mehr benötigt, können Sie mit den Invalidierungsmethoden Daten aus dem nahen Cache bzw. Server-Cache entfernen. Die Methode "beginNoWriteThrough" gilt für alle ObjectMap- und EntityManager-Operationen im lokalen Cache ohne Aufruf des Loaders. Wenn die Methode von einem Client aufgerufen wird, gilt die Operation nur für den nahen Cache (der ferne Loader wird nicht aufgerufen). Wird die Methode im Server aufgerufen, gilt die Operation nur für den Serverbasiscache ohne Aufruf des Loaders.

Dateninvalidierung

Sie können Invalidierungsmechanismen verwenden, um veraltete Cachedaten zu entfernen.

Administrative Invalidierung

Sie können die Webkonsole oder das Dienstprogramm **xscmd** verwenden, um Daten basierend auf dem Schlüssel ungültig zu machen. Sie können die Cachedaten mit einem regulären Ausdruck filtern und die Daten dann auf der Basis des regulären Ausdrucks ungültig machen.

Ereignisgesteuerte Invalidierung

Teilcaches und vollständige Caches können mit Hilfe eines Ereignisgenerators wie Java Message Service (JMS) ungültig gemacht oder aktualisiert werden. Die Invalidierung mit JMS kann manuell an jeden Prozess gebunden werden, der das Back-End über einen Datenbankauslöser aktualisiert. Es wird ein JMS-ObjectGridEvent-Listener-Plug-in in eXtreme Scale bereitgestellt, das Clients benachrichtigen kann, wenn Änderungen im Server-Cache vorgenommen wurden. Dieser Typ von Benachrichtigung verkleinert das Zeitfenster, in dem der Client veraltete Daten sieht.

Der ereignisgesteuerte Invalidierungsmechanismus setzt sich gewöhnlich aus den folgenden drei Komponenten zusammen:

- **Ereigniswarteschlange:** In einer Ereigniswarteschlange werden die Datenänderungsereignisse gespeichert. Die Ereigniswarteschlange kann eine JMS-Warteschlange, eine Datenbank, eine speicherinterne FIFO-Warteschlange oder ein beliebiges Manifest sein, das Datenänderungsereignisse verwalten kann.
- **Ereignis-Publisher:** Ein Ereignis-Publisher veröffentlicht die Datenänderungsereignisse in der Ereigniswarteschlange. Ein Ereignis-Publisher ist gewöhnlich eine Anwendung, die Sie erstellen, oder eine Implementierung eines eXtreme-Scale-Plug-ins. Der Ereignis-Publisher weiß, wann die Daten geändert werden, oder ändert die Daten selbst. Wenn eine Transaktion festgeschrieben wird, werden Ereignisse für die geänderten Daten generiert, und der Ereignis-Publisher veröffentlicht diese Ereignisse in der Ereigniswarteschlange.
- **Ereigniskonsument:** Ein Ereigniskonsument konsumiert Datenänderungsereignisse. Der Ereigniskonsument ist gewöhnlich eine Anwendung, die sicherstellt, dass die Daten im Zielgrid mit den neuesten Änderungen aus anderen Grids aktualisiert werden. Dieser Ereigniskonsument interagiert mit der Ereigniswarteschlange, um die neuesten Datenänderungen abzurufen, und wendet die Datenänderungen auf das Zielgrid an. Die Ereigniskonsumenten können APIs von eXtreme Scale verwenden, um veraltete Daten ungültig zu machen oder um das Grid mit den neuesten Daten zu aktualisieren.

JMSObjectGridEventListener hat beispielsweise eine Option für ein Client/Server-Modell, bei der die Ereigniswarteschlange eine festgelegte JMS-Destination ist. Alle Serverprozesse sind Ereignis-Publisher. Wenn eine Transaktion festgeschrieben wird, ruft der Server die Datenänderungen ab und veröffentlicht sie in der festgelegten JMS-Destination. Alle Clientprozesse sind Ereigniskonsumenten. Sie empfangen Datenänderungen von der festgelegten JMS-Destination und wenden die Änderungen auf den nahen Cache des Clients an.

Weitere Informationen finden Sie unter JMS-basierte Clientsynchronisation konfigurieren.

Programmgesteuerte Invalidierung

Die APIs von WebSphere eXtreme Scale unterstützen die manuelle Interaktion zwischen nahem Cache und Server-Cache über die API-Methoden "Session.beginNoWriteThrough()", "ObjectMap.invalidate()" und "EntityManager.invalidate()".

Wenn ein Client- oder Serverprozess einen Teil der Daten nicht mehr benötigt, können Sie mit den Invalidierungsmethoden Daten aus dem nahen Cache bzw. Server-Cache entfernen. Die Methode "beginNoWriteThrough" gilt für alle ObjectMap- und EntityManager-Operationen im lokalen Cache ohne Aufruf des Loaders. Wenn die Methode von einem Client aufgerufen wird, gilt die Operation nur für den nahen Cache (der ferne Loader wird nicht aufgerufen). Wird die Methode im Server aufgerufen, gilt die Operation nur für den Serverbasiscache ohne Aufruf des Loaders.

Sie können die programmgesteuerte Invalidierung zusammen mit anderen Techniken verwenden, um festzustellen, wann die Daten ungültig gemacht werden müssen. Diese Invalidierungsmethode verwendet beispielseise ereignisgesteuerte Invalidierungsmechanismen, um die Datenänderungsereignisse zu empfangen, und anschließend APIs, um die veralteten Daten ungültig zu machen.

8.6+ Invalidierung des nahen Caches

Wenn Sie einen nahen Cache verwenden, können Sie eine asynchrone Invalidierung konfigurieren, die jedes Mal ausgelöst wird, wenn eine Aktualisierungs-, Lösch- oder Invalidierungsoperation im Datengrid ausgeführt wird. Da die Operation asynchron ist, können Sie trotzdem veraltete Daten im Datengrid sehen.

Zum Aktivieren der Invalidierung des nahen Caches setzen Sie das Attribut **`nearCacheInvalidationEnabled`** in der BackingMap in der ObjectGrid-XML-Deskriptordatei.

Indexierung

Java

Verwenden Sie das Plug-in "MapIndexPlugin", um einen Index oder mehrere Indizes in einer BackingMap für die Unterstützung von Datenzugriffen ohne Schlüssel zu erstellen.

Indextypen und Konfiguration

Das Indexierungsfeature wird durch das Plug-in "MapIndexPlugin" oder kurz "Index" dargestellt. Index ist ein BackingMap-Plug-in. Für eine BackingMap können mehrere Index-Plug-ins konfiguriert werden, solange jedes Plug-in den Index-Konfigurationsregeln entspricht.

Sie können das Indexierungsfeature verwenden, um einen oder mehrere Indizes in einer BackingMap zu erstellen. Ein Index wird aus einem Attribut oder einer Liste von Attributen eines Objekts in der BackingMap erstellt. Das Feature bietet Anwendungen eine Möglichkeit, bestimmte Objekte schneller zu finden. Mit dem Indexierungsfeature können Anwendungen mit einem bestimmten Wert oder innerhalb eines bestimmten Wertebereichs indexierter Attribute finden.

Es gibt zwei Typen von Indexierung: statische Indexierung und dynamische Indexierung. Bei der statischen Indexierung müssen Sie das Index-Plug-in in der BackingMap konfigurieren, bevor Sie die ObjectGrid-Instanz initialisieren. Sie können diese Konfiguration durch XML- oder programmgesteuerte Konfiguration der BackingMap vornehmen. Die statische Indexierung beginnt mit der Erstellung eines Index während der ObjectGrid-Initialisierung. Der Index ist immer mit der BackingMap synchronisiert und zur Verwendung bereit. Nach dem Start des stati-

schen Indexierungsprozesses erfolgt die Verwaltung des Index im Rahmen des Transaktionsverwaltungsprozesses von eXtreme Scale. Wenn Transaktionen Änderungen festschreiben, werden diese Änderungen auch im statischen Index durchgeführt, und Indexänderungen werden rückgängig gemacht, wenn die Transaktion rückgängig gemacht wird.

Bei der dynamischen Indexierung können Sie einen Index in einer BackingMap vor oder nach der Initialisierung der übergeordneten ObjectGrid-Instanz erstellen. Anwendungen haben eine Lebenszykluskontrolle über den dynamischen Indexierungsprozess, d. h., Sie können einen dynamischen Index entfernen, wenn er nicht mehr benötigt wird. Wenn eine Anwendung einen dynamischen Index erstellt, ist der Index möglicherweise nicht zur sofortigen Verwendung bereit, weil die Erstellung des Index eine gewisse Zeit dauert. Da die Erstellungsdauer vom Volumen der zu indexierenden Daten abhängig ist, wird die Schnittstelle "DynamicIndexCallback" für Anwendungen bereitgestellt, die Benachrichtigungen empfangen möchten, wenn bestimmte Indexierungsereignisse eintreten. Zu diesen Ereignissen gehören die Bereitschaft des Index (ready), Fehler (error) und das Löschen des Index (destroy). Anwendungen können diese Callback-Schnittstelle implementieren und sich beim dynamischen Indexierungsprozess registrieren.

8.6+ Wenn eine BackingMap ein konfiguriertes Index-Plug-in hat, können Sie das Proxy-Objekt für den Anwendungsindex von der entsprechenden ObjectMap abrufen. Wenn Sie die Methode getIndex in der Schnittstelle "ObjectMap" aufrufen und den Namen des Index-Plug-ins übergeben, wird das Index-Proxy-Objekt zurückgegeben. Sie müssen das Index-Proxy-Objekt in die entsprechende Anwendungsindexschnittstelle, z. B. MapIndex, MapRangeIndex, MapGlobalIndex oder eine angepasste Indexschnittstelle, umsetzen. Nach dem Abrufen des Index-Proxy-Objekts können Sie in der Anwendungsindexschnittstelle definierte Methoden verwenden, um zwischengespeicherte Objekte zu suchen.

Die Schritte zur Verwendung der Indexierung sind in der folgenden Liste zusammengefasst:

- Fügen Sie statische oder dynamische Index-Plug-ins in der BackingMap hinzu.
- Rufen Sie mit der Methode "getIndex" von ObjectMap ein Proxy-Objekt für den Anwendungsindex ab.
- Setzen Sie das Proxy-Objekt für den Index in eine entsprechende Anwendungsindexschnittstelle um, wie z. B. MapIndex, MapRangeIndex oder eine angepasste Indexschnittstelle.
- Verwenden Sie die in der Anwendungsindexschnittstelle definierten Methoden, um zwischengespeicherte Objekte zu suchen.

8.6+ Die Klasse "HashIndex" ist die integrierte Index-Plug-in-Implementierung, die die folgenden integrierten Anwendungsindexschnittstellen unterstützen kann:

- MapIndex
- MapRangeIndex
- MapGlobalIndex

Sie können auch eigene Indizes erstellen. Sie können HashIndex als statischen oder dynamischen Index in der BackingMap hinzufügen, ein MapIndex-, MapRangeIndex- oder MapGlobalIndex-Index-Proxy-Objekt abrufen und das Index-Proxy-Objekt zum Suchen zwischengespeicherter Objekte verwenden.

8.6+

Globaler Index

Der globale Index ist eine Erweiterung der integrierten Klasse "HashIndex", die auf Shards in verteilten, partitionierten Datengridumgebungen ausgeführt wird. Der globale Index überwacht die Position indexierter Attribute und bietet effiziente Möglichkeiten, um Partitionen, Schlüssel, Werte oder Einträge anhand von Attributen in großen, partitionierten Datengridumgebungen zu finden.

Wenn der globale Index im integrierten HashIndex-Plug-in aktiviert ist, können Anwendungen ein Index-Proxy-Objekt in den Typ "MapGlobalIndex" umsetzen und diesen zum Suchen von Daten verwenden.

Standardindex

Wenn Sie durch die Schlüssel in einer lokalen Map iterieren möchten, können Sie den Standardindex verwenden. Dieser Index erfordert keine Konfiguration, aber er muss für das Shard über einen Agenten oder eine ObjectGrid-Instanz, die mit der Methode ShardEvents.shardActivated(ObjectGrid shard) abgerufen wird, verwendet werden.

Hinweis zur Datenqualität

Die Ergebnisse der Indexabfragemethoden stellen nur eine Momentaufnahme der Daten zu einem bestimmten Zeitpunkt dar. Es werden keine Sperren für Dateneinträge angefordert, nachdem die Ergebnisse an die Anwendung zurückgegeben wurden. Die Anwendung muss sich darüber im Klaren sein, dass Datenaktualisierungen für eine zurückgegebene Datengruppe vorgenommen werden können. Beispiel: Die Anwendung ruft den Schlüssel eines zwischengespeicherten Objekts mit der Methode findAll von MapIndex ab. Dieses zurückgegebene Schlüsselobjekt ist einem Dateneintrag im Cache zugeordnet. Die Anwendung muss in der Lage sein, die Methode "get" in ObjectMap auszuführen, um ein Objekt durch Übergabe des Schüsselobjekts zu suchen. Wenn eine andere Transaktion das Datenobjekt aus dem Cache entfernt, kurz bevor die Methode "get" aufgerufen wird, ist das zurückgegebene Ergebnis null.

Hinweise zur Leistung der Indexierung

Eine der Hauptzielsetzungen des Indexierungsfeatures ist die Verbesserung der Gesamtleistung der BackingMap. Wenn die Indexierung nicht ordnungsgemäß verwendet wird, kann dies die Leistung der Anwendung beeinträchtigen. Berücksichtigen Sie vor der Verwendung dieses Features die folgenden Faktoren.

- **Anzahl gleichzeitiger Transaktionen mit Schreibzugriff:** Die Indexverarbeitung kann jedesmal stattfinden, wenn eine Transaktion Daten in eine BackingMap schreibt. Schreiben viele Transaktionen gleichzeitig Daten in die Map, kann es zu Leistungseinbußen kommen, wenn eine Anwendung versucht, Indexabfrageoperationen durchzuführen.
- **Größe der von einer Abfrageoperation zurückgegebenen Ergebnismenge:** Je größer die Ergebnismenge wird, desto mehr nimmt die Abfrageleistung ab. Ab einer Ergebnismengengröße von 15 % der Gesamtgröße der BackingMap beginnen sich Leistungseinbußen abzuzeichnen.
- **Anzahl der für dieselbe BackingMap erstellten Indizes:** Jeder Index belegt Systemressourcen. Mit steigender Indexanzahl für die BackingMap nimmt die Leistung ab.

Die Indexierungsfunktion kann die Leistung einer BackingMap erheblich verbessern. Die besten Ergebnisse lassen sich erzielen, wenn hauptsächlich Leseoperationen für die BackingMap durchgeführt werden, wenn die Abfrageergebnismenge nur einen kleinen Prozentsatz der BackingMap-Einträge enthält und wenn nur einige wenige Indizes für die BackingMap erstellt werden.

JPA-Loader

Java

Java Persistence API (JPA) ist eine Spezifikation, die die Zuordnung von Java-Objekten zu relationalen Datenbank ermöglicht. JPA enthält eine vollständige ORM-Spezifikation (Object-Relational Mapping, objektrelationale Abbildung) mit Metadatenannotationen für die Sprache Java und XML-Deskriptoren für die Definition der Zuordnung von Java-Objekten zu einer relationalen Datenbank und umgekehrt. Es gibt eine Reihe von Open-Source- und kostenpflichtigen Implementierungen.

Sie können eine JPA-Loader-Plug-in-Implementierung mit eXtreme Scale verwenden, um mit jeder vom ausgewählten Loader unterstützten Datenbank zu interagieren. Zur Verwendung von JPA müssen Sie einen unterstützten JPA-Provider wie OpenJPA oder Hibernate, JAR-Dateien und eine Datei `META-INF/persistence.xml` in Ihrem Klassenpfad haben.

Das JPALoader-Plug-in "com.ibm.websphere.objectgrid.jpa.JPALoader" und das JPAEntityLoader-Plug-in "com.ibm.websphere.objectgrid.jpa.JPAEntityLoader" sind zwei integrierte JPA-Loader-Plug-ins, die verwendet werden, um die ObjectGrid-Maps mit einer Datenbank zu synchronisieren. Sie müssen eine JPA-Implementierung wie Hibernate oder OpenJPA haben, um dieses Feature verwenden zu können. Als Datenbank kann jedes Back-End verwendet werden, das vom ausgewählten JPA-Provider unterstützt wird.

Sie können das JPALoader-Plug-in verwenden, wenn Sie Daten über die API "ObjectMap" speichern. Verwenden Sie das JPAEntityLoader-Plug-in, wenn Sie Daten über die API "EntityManager" speichern.

Architektur der JPA-Loader

Der JPA-Loader wird für eXtreme-Scale-Maps verwendet, in denen POJOs (Plain Old Java Objects) gespeichert werden.

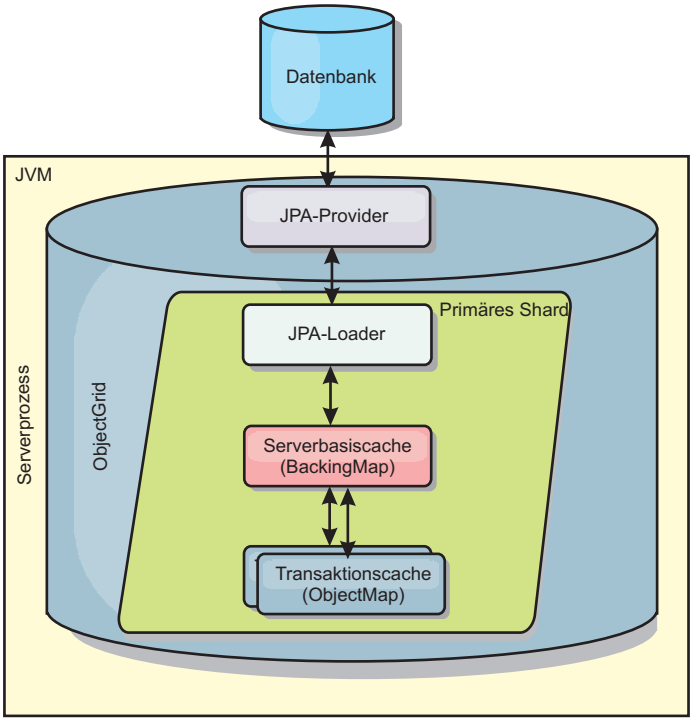


Abbildung 31. Architektur der JPA-Loader

Wenn eine Methode ObjectMap.get(Object key) aufgerufen wird, prüft die Laufzeitumgebung von eXtreme Scale zunächst, ob der Eintrag auf ObjectMap-Ebene vorhanden ist. Wenn nicht, delegiert die Laufzeitumgebung die Anforderung an den JPA-Loader. Auf die Anforderung hin, den Schlüssel zu laden, ruft der JPALoader die JPA-Methode EntityManager.find(Object key) auf, um die Daten auf JPA-Ebene zu suchen. Sind die Daten im JPA-EntityManager vorhanden, werden sie zurückgegeben, wenn nicht, interagiert der JPA-Provider mit der Datenbank, um den Wert abzurufen.

Bei einer Aktualisierung der ObjectMap, z. B. über die Methode "ObjectMap.update(Object key, Object value)", erstellt die Laufzeitumgebung von eXtreme Scale ein LogElement-Objekt für diese Aktualisierung und sendet es an den JPALoader. Der JPALoader ruft die JPA-Methode "EntityManager.merge(Object value)" auf, um den Wert in der Datenbank zu aktualisieren.

Beim JPAEntityLoader sind dieselben vier Ebenen beteiligt. Da das JPAEntityLoader-Plug-in jedoch für Maps verwendet wird, in denen eXtreme-Scale-Entitäten gespeichert werden, können Relationen zwischen den einzelnen Entitäten das Einsatzszenario komplizierter machen. Eine eXtreme-Scale-Entität unterscheidet sich von einer JPA-Entität. Weitere Einzelheiten finden Sie in den Informationen zum JPAEntityLoader-Plug-in in der Veröffentlichung *Programmierung*.

Methoden

Loader (Ladeprogramme) stellen drei Hauptmethoden bereit:

1. get: Gibt eine Liste mit Werten zurück, die der Liste der Schlüssel entspricht, die durch Abruf der Daten über JPA übergeben werden. Die Methode verwendet JPA, um die Entitäten in der Datenbank zu suchen. Für das JPALoader-Plug-in enthält die zurückgegebene Liste eine Liste der JPA-Entitäten, die direkt von der Suchoperation zurückgegeben werden. Für das JPAEntityLoader-Plug-

in enthält die zurückgegebene Liste Tupel für die eXtreme-Scale-Entitätswerte, die aus den JPA-Entitäten konvertiert wurden.

2. batchUpdate: Schreibt die Daten aus den ObjectGrid-Maps in die Datenbank. Ja nach Operationstyp (Einfügen, Aktualisieren oder Löschen) verwendet der Loader die JPA-Operationen "persist", "merge" und "remove", um die Daten in der Datenbank zu aktualisieren. Für JPALoader werden die Objekte in der Map direkt als JPA-Entitäten verwendet. Für JPAEntityLoader werden die Entitätstupel in der Map in Objekte konvertiert, die als JPA-Entitäten verwendet werden.
3. preloadMap: Lädt die Daten über die Methode "ClientLoader.load" des Clientladeprogramms vorab in die Map. Für partitionierte Maps wird die Methode "preloadMap" nur in einer einzigen Partition aufgerufen. Die Partition wird mit der Eigenschaft "preloadPartition" der Klasse "JPALoader" bzw. "JPAEntityLoader" angegeben. Wenn die Eigenschaft "preloadPartition" auf einen Wert kleiner als null oder größer als (*Gesamtanzahl_der_Partitionen* - 1) gesetzt wird, wird das vorherige Laden inaktiviert.

JPALoader- und JPAEntityLoader-Plug-ins arbeiten mit JPATxCallback, um die eXtreme-Scale-Transaktionen und JPA-Transaktionen zu koordinieren. JPATxCallback muss in der ObjectGrid-Instanz für die Verwendung dieser beiden Loader konfiguriert werden.

Konfiguration und Programmierung

Wenn Sie JPA-Loader in einer Multimasterumgebung verwenden, lesen Sie den Abschnitt „Hinweise zu Ladeprogrammen in einer Multimastertopologie" auf Seite 183. Weitere Einzelheiten zum Konfigurieren von JPA-Loadern finden Sie in den Informationen zu JPA-Loadern in der Veröffentlichung *Verwaltung*. Weitere Informationen zur Programmierung von JPA-Loadern finden Sie in der Veröffentlichung *Programmierung*.

Übersicht über die Serialisierung

Java

Daten werden im Datengrid immer in Form von Java-Objekten ausgedrückt, aber nicht unbedingt in dieser Form gespeichert. WebSphere eXtreme Scale verwendet mehrere Java-Prozesse für die Serialisierung der Daten, indem die Java-Objektinstanzen bei Bedarf in Bytes und dann wieder in Objekte konvertiert werden, um die Daten zwischen Client- und Serverprozessen verschoben werden.

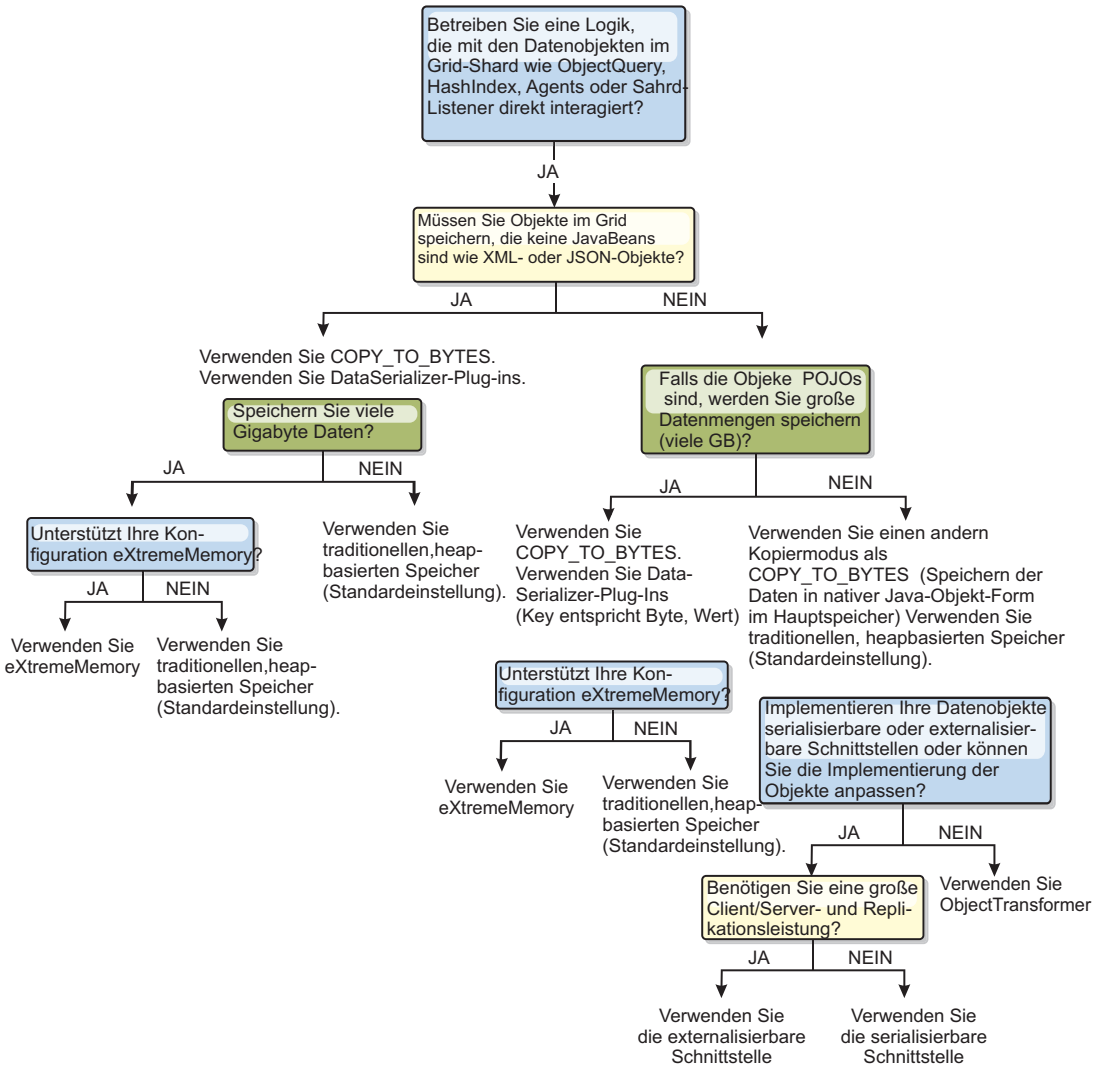
Bei der Serialisierung von Daten werden die Daten in den folgenden Situationen für die Übertragung über ein Netz in einen Datenstrom konvertiert:

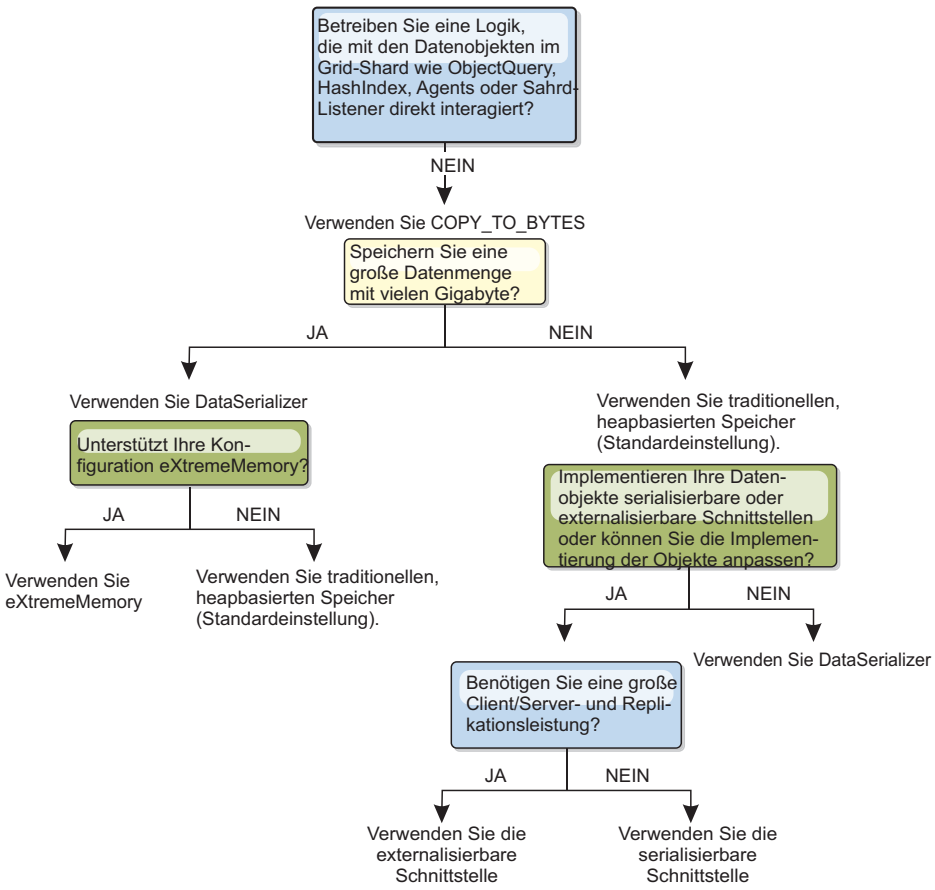
- Clients kommunizieren mit Servern, und diese Server senden Informationen an den Client zurück.
- Server replizieren Daten von einem Server auf einem anderen.

Alternativ können Sie auf den Serialisierungsprozess über WebSphere eXtreme Scale verzichten und Rohdaten als Byte-Arrays speichern. Byte-Arrays können wesentlich kostengünstiger im Hauptspeicher gespeichert werden, da die JVM (Java Virtual Machine) für die Garbage-Collection weniger Objekte durchsuchen muss und die Objekte ausschließlich bei Bedarf dekomprimiert werden können. Verwenden Sie Byte-Arrays nur, wenn Sie keinen Zugriff auf die Objekte über Abfragen oder Indizes benötigen. Da die Daten als Bytes gespeichert werden, hat eXtreme Scale keine Metadaten für die Beschreibung der abzufragenden Attribute.

Zum Serialisieren der Daten in eXtreme Scale können Sie Java-Serialisierung, das ObjectTransformer-Plug-in oder die DataSerializer-Plug-ins verwenden. Zum Optimieren der Serialisierung mit jeder dieser Optionen können Sie den Modus COPY_TO_BYTES verwenden, um die Leistung um bis 70 Prozent zu verbessern, weil die Daten serialisiert werden, wenn Transaktionen festgeschrieben werden, d. h., die Serialisierung findet nur ein einziges Mal statt. Die serialisierten Daten werden unverändert vom Client an den Server bzw. vom Server an den replizierten Server gesendet. Durch die Verwendung des Modus COPY_TO_BYTES können Sie den Speicherbedarf großer Objektgraphen reduzieren.

Verwenden Sie die folgenden Abbildungen, um festzustellen, welcher Typ von Serialisierungsmethode für Ihre Entwicklungsanforderungen am besten geeignet ist. Die erste Abbildung beschreibt die Serialisierungsmethoden, die verfügbar sind, wenn Sie Logik ausführen, die direkt mit Datenobjekten im Daten-Shard interagieren. Die letzte Abbildung zeigt die verfügbaren Optionen, wenn Sie nicht direkt mit dem Grid-Shard interagieren.





Weitere Informationen zu den unterstützten Formen der Serialisierung im Produkt eXtreme Scale finden Sie in den folgenden Abschnitten:

Serialisierung mit Java

Java-Serialisierung bezieht entweder auf die Standardserialisierung, bei der die Schnittstelle "Serializable" verwendet wird, oder auf die angepasste Serialisierung, bei der die Schnittstellen "Serializable" und "Externalizable" verwendet werden.

Standardserialisierung

Wenn Sie die Standardserialisierung verwenden möchten, implementieren Sie die Schnittstelle "java.io.Serializable", die die API enthält, die Objekte in Bytes konvertiert, die später deserialisiert werden. Verwenden Sie die Klasse java.io.ObjectOutputStream, um das Objekt persistent zu speichern. Rufen Sie anschließend die Methode ObjectOutputStream.writeObject() auf, um die Serialisierung einzuleiten und das Java-Objekt zu vereinfachen.

Angepasste Serialisierung

Es gibt einige Fälle, in denen Objekte für die Verwendung einer angepassten Serialisierung geändert werden müssen, z. B. durch Implementierung der Schnittstelle "java.io.Externalizable" oder durch Implementierung der Methoden "writeObject" und "readObject" für Klassen, die die Schnittstelle "java.io.Serializable" implementieren. Sie müssen angepasste Serialisierungstechniken verwenden, wenn die Objekte mit anderen Mechanismen als den Methoden der API "ObjectGrid" oder "EntityManager" serialisiert werden.

Wenn Objekte oder Entitäten beispielsweise als Instanzdaten in einem DataGrid-API-Agenten gespeichert werden oder wenn der Agent Objekte oder Entitäten zurückgibt, werden diese Objekte nicht mit einem ObjectTransformer umgesetzt. Der Agent verwendet jedoch automatisch den ObjectTransformer, wenn Sie die Schnittstelle "`EntityMixin`" verwenden. Weitere Einzelheiten finden Sie in der Dokumentation zu DataGrid-Agenten und entitätsbasierten Maps.

ObjectTransformer-Plug-in

Java

Mit dem ObjectTransformer-Plug-in können Sie Objekte im Cache serialisieren, entserialisieren und kopieren, um eine höhere Leistung zu erzielen.

 Die Schnittstelle "ObjectTransformer" wurde durch die DataSerializer-Plug-ins ersetzt, die Sie verwenden können, um beliebige Daten effizient in WebSphere eXtreme Scale zu speichern, damit vorhandene Produkt-APIs effizient mit Ihren Daten interagieren können.

Wenn Sie Leistungsprobleme in Bezug auf die Prozessorbelegung lesen, fügen Sie jeder Map ein ObjectTransformer-Plug-in hinzu. Wenn Sie kein ObjectTransformer-Plug-in bereitstellen, werden bis zu 60-70 % der gesamten Prozessorzeit mit der Serialisierung und das Kopieren von Einträgen verbracht.

Zweck

Wenn Sie das ObjectTransformer-Plug-in verwenden, können Ihre Anwendung angepasste Methoden für die folgenden Operationen bereitstellen:

- Serialisierung und Entserialisierung des Schlüssels für einen Eintrag,
- Serialisierung und Entserialisierung des Werts für einen Eintrag,
- Kopieren eines Schlüssels oder eines Werts für einen Eintrag.

Wenn kein ObjectTransformer-Plug-in bereitgestellt wird, müssen Sie in der Lage sein, die Schlüssel und Werte zu serialisieren, weil das ObjectGrid eine Serialisierungs- und Entserialisierungsfolge verwendet, um die Objekte zu kopieren. Diese Methode ist kostenintensiv, und deshalb sollten Sie ein ObjectTransformer-Plug-in verwenden, wenn die Leistung kritisch ist. Der Kopiervorgang findet statt, wenn eine Anwendung ein Objekt in einer Transaktion zu ersten Mal sucht. Sie können diesen Kopiervorgang vermeiden, indem Sie den Kopiermodus der Map auf NO_COPY setzen. Mit der Kopiermoduseinstellung COPY_ON_READ können Sie die Anzahl der Kopiervorgänge reduzieren. Optimieren Sie die Kopieroperation, wenn diese von der Anwendung benötigt wird, indem Sie eine angepasste Kopiermethode in diesem Plug-in bereitstellen. Ein solches Plug-in kann den Kopieraufwand von 65–70 % auf 2/3 % der gesamten Prozessorzeit reduzieren.

Die Standardimplementierungen der Methoden "copyKey" und "copyValue" versuchen zuerst, die Methode "clone" zu verwenden, sofern diese verfügbar ist. Wenn keine Implementierung der Methode "clone" bereitgestellt wird, wird standardmäßig mit Serialisierung gearbeitet.

Die Objektserialisierung wird auch direkt verwendet, wenn eXtreme Scale im verteilten Modus ausgeführt wird. LogSequence verwendet das ObjectTransformer-Plug-in für die Serialisierung von Schlüsseln und Werten, bevor die Änderungen an die Peers im ObjectGrid übertragen werden. Sie müssen sorgfältig vorgehen, wenn Sie eine angepasste Serialisierungsmethode als Ersatz für die integrierte JDK-

Serialisierung bereitstellen. Die Versionssteuerung von Objekten ist eine komplexes Thema, und es können Probleme mit der Versionskompatibilität auftreten, wenn Sie nicht sicherstellen, dass Ihre angepassten Methoden für die Versionssteuerung konzipiert sind.

Die folgende Liste enthält Details zur Serialisierung von Schlüsseln und Werten durch eXtreme Scale:

- Wenn ein angepasstes ObjectTransformer-Plug-in geschrieben und integriert wird, ruft eXtreme Scale Methoden in der Schnittstelle "ObjectTransformer" auf, um Schlüssel und Werte zu serialisieren und Kopien von Objektschlüsseln und -werten abzurufen.
- Wenn kein angepasstes ObjectTransformer-Plug-in verwendet wird, führt eXtreme Scale die Serialisierung und Entserialisierung von Werten nach dem Standardverfahren durch. Wenn das Standard-Plug-in verwendet wird, wird jedes Objekt als extern bereitstellbares (Externalizable) oder als serialisierbares (Serializable) Objekt implementiert.
  - Wenn das Objekt die Schnittstelle "Externalizable" unterstützt, wird die Methode "writeExternal" aufgerufen. Objekte, die als extern bereitstellbare Objekte implementiert werden, tragen zu einer besseren Leistung bei.
  - Wenn das Objekt die Schnittstelle "Externalizable" nicht unterstützt und die Schnittstelle "Serializable" implementiert, wird das Objekt mit der Methode "ObjectOutputStream" gespeichert.

Schnittstelle "ObjectTransformer" verwenden

Ein ObjectTransformer-Objekt muss die Schnittstelle "ObjectTransformer" implementieren und die folgenden allgemeinen Konventionen für ObjectGrid-Plug-ins einhalten.

Es gibt zwei Ansätze (programmgesteuerte Konfiguration und XML-Konfiguration), mit denen ein ObjectTransformer-Objekt im folgt der BackingMap-Konfiguration hinzugefügt werden kann.

ObjectTransformer-Objekt programmgesteuert integrieren

Das folgende Code-Snippet erstellt das angepasste ObjectTransformer-Objekt und fügt es einer BackingMap hinzu:

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid myGrid = objectGridManager.createObjectGrid("myGrid", false);
BackingMap backingMap = myGrid.getMap("myMap");
MyObjectTransformer myObjectTransformer = new MyObjectTransformer();
backingMap.setObjectTransformer(myObjectTransformer);
```

ObjectTransformer durch XML-Konfiguration integrieren

Angenommen, der Klassenname der ObjectTransformer-Implementierung ist "com-.company.org.MyObjectTransformer". Diese Klasse implementiert die Schnittstelle "ObjectTransformer". Eine ObjectTransformer-Implementierung kann mit der folgenden XML konfiguriert werden:

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
    xmlns="http://ibm.com/ws/objectgrid/config">
    <objectGrids>
        <objectGrid name="myGrid">
     <backingMap name="myMap" pluginCollectionRef="myMap" />
        </objectGrid>
    </objectGrids>
```

```
    <backingMapPluginCollections>
 <backingMapPluginCollection id="myMap">
      <bean id="ObjectTransformer" className="com.company.org.MyObjectTransformer" />
 </backingMapPluginCollection>
    </backingMapPluginCollections>
</objectGridConfig>
```

Einsatzszenarien für ObjectTransformer

Sie können das ObjectTransformer-Plug-in in den folgenden Situationen verwenden:

- Nicht serialierbares Objekt
- Serialisierbares Objekt, aber Serialisierungsleistung muss verbessert werden
- Schlüssel- oder Wertkopie

Im folgenden Beispiel wird ObjectGrid verwendet, um die Klasse "Stock" zu speichern:

```
/**
* Stock-Objekt für ObjectGrid-Demonstration
*
*
*/
public class Stock implements Cloneable {
    String ticket;
    double price;
    String company;
    String description;
    int serialNumber;
    long lastTransactionTime;
    /**
    * @return Gibt die Beschreibung zurück.
    */
    public String getDescription() {
        return description;
    }
    /**
    * @param description Die festzulegende Beschreibung.
    */
    public void setDescription(String description) {
        this.description = description;
    }
    /**
    * @return Gibt den lastTransactionTime-Wert zurück.
    */
    public long getLastTransactionTime() {
        return lastTransactionTime;
    }
    /**
    * @param lastTransactionTime Der festzulegende lastTransactionTime-Wert.
    */
    public void setLastTransactionTime(long lastTransactionTime) {
        this.lastTransactionTime = lastTransactionTime;
    }
    /**
    * @return Gibt den Preis zurück.
    */
    public double getPrice() {
        return price;
    }
    /**
    * @param price Der festzulegende Preis.
    */
    public void setPrice(double price) {
        this.price = price;
    }
    /**
    * @return Gibt den serialNumber-Wert zurück.
    */
    public int getSerialNumber() {
        return serialNumber;
    }
    /**
    * @param serialNumber Der festzulegende serialNumber-Wert.
    */
```

```
    public void setSerialNumber(int serialNumber) {
        this.serialNumber = serialNumber;
    }
    /**
    * @return Gibt das Ticket zurück.
    */
    public String getTicket() {
        return ticket;
    }
    /**
    * @param ticket Das festzulegende Ticket.
    */
    public void setTicket(String ticket) {
        this.ticket = ticket;
    }
    /**
    * @return Gibt die Firma zurück.
    */
    public String getCompany() {
        return company;
    }
    /**
    * @param company Die festzulegende Firma.
    */
    public void setCompany(String company) {
        this.company = company;
    }
    //clone
    public Object clone() throws CloneNotSupportedException
    {
        return super.clone();
    }
}
```

Sie können eine angepasste ObjectTransformer-Klasse für die Klasse "Stock" schreiben:

```
/**
* Angepasste Implementierung des ObjectGrid-ObjectTransformer-Plug-ins für Stock-Objekt
*
*/
public class MyStockObjectTransformer implements ObjectTransformer {
/* (keine Javadoc)
* @see
* com.ibm.websphere.objectgrid.plugins.ObjectTransformer#serializeKey
* (java.lang.Object,
* java.io.ObjectOutputStream)
*/
public void serializeKey(Object key, ObjectOutputStream stream) throws IOException {
    String ticket= (String) key;
    stream.writeUTF(ticket);
}

/* (keine Javadoc)
* @see com.ibm.websphere.objectgrid.plugins.
ObjectTransformer#serializeValue(java.lang.Object,
java.io.ObjectOutputStream)
*/
public void serializeValue(Object value, ObjectOutputStream stream) throws IOException {
    Stock stock= (Stock) value;
    stream.writeUTF(stock.getTicket());
    stream.writeUTF(stock.getCompany());
    stream.writeUTF(stock.getDescription());
    stream.writeDouble(stock.getPrice());
    stream.writeLong(stock.getLastTransactionTime());
    stream.writeInt(stock.getSerialNumber());
}

/* (keine Javadoc)
* @see com.ibm.websphere.objectgrid.plugins.
ObjectTransformer#inflateKey(java.io.ObjectInputStream)
*/
public Object inflateKey(ObjectInputStream stream) throws IOException, ClassNotFoundException {
    String ticket=stream.readUTF();
    return ticket;
}

/* (keine Javadoc)
* @see com.ibm.websphere.objectgrid.plugins.
ObjectTransformer#inflateValue(java.io.ObjectInputStream)
*/

public Object inflateValue(ObjectInputStream stream) throws IOException, ClassNotFoundException {
    Stock stock=new Stock();
    stock.setTicket(stream.readUTF());
```

```
    stock.setCompany(stream.readUTF());
    stock.setDescription(stream.readUTF());
    stock.setPrice(stream.readDouble());
    stock.setLastTransactionTime(stream.readLong());
    stock.setSerialNumber(stream.readInt());
    return stock;
}

/* (keine Javadoc)
* @see com.ibm.websphere.objectgrid.plugins.
ObjectTransformer#copyValue(java.lang.Object)
*/
public Object copyValue(Object value) {
    Stock stock = (Stock) value;
    try {
        return stock.clone();
    }
    catch (CloneNotSupportedException e)
    {
        // Ausnahmenachricht anzeigen    }
}

/* (keine Javadoc)
* @see com.ibm.websphere.objectgrid.plugins.
ObjectTransformer#copyKey(java.lang.Object)
*/
public Object copyKey(Object key) {
    String ticket=(String) key;
    String ticketCopy= new String (ticket);
    return ticketCopy;
}
}
```

Integrieren Sie anschließend diese angepasste Klasse "MyStockObjectTransformer" in die BackingMap:

```
ObjectGridManager ogf=ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid og = ogf.getObjectGrid("NYSE");
BackingMap bm = og.defineMap("NYSEStocks");
MyStockObjectTransformer ot = new MyStockObjectTransformer();
bm.setObjectTransformer(ot);
```

Serialisierung mit den DataSerializer-Plug-ins

Verwenden Sie die DataSerializer-Plug-ins, um beliebige Daten in WebSphere eXtreme Scale effizient zu speichern, sodass vorhandene Produkt-APIs effizient mit Ihren Daten interagieren können.

Serialisierungsmethoden wie Java-Serialisierung und das Plug-in "ObjectTransformer" ermöglichen das Marshalling von Daten über das Netz. Wenn Sie diese Serialisierungsoptionen mit dem Kopiermodus COPY_TO_BYTES verwenden, wird das Verschieben von Daten zwischen Clients und Servern außerdem kostengünstiger, und die Leistung verbessert sich. Diese Optionen löschen jedoch die folgenden potenziell auftretenden Probleme nicht:

- Schlüssel werden nicht in Bytes gespeichert, sie bleiben Java-Objekte.
- Der serverseitige Code muss das Objekt weiterhin deserialisieren. Abfrage und Index können beispielsweise weiterhin Reflexion verwenden und müssen das Objekt deserialisieren. Außerdem benötigen Agenten, Listener und Plug-ins weiterhin das Objektformat.
- Klassen müssen weiterhin im Serverklassenpfad enthalten sein.
- Die Daten behalten das Java-Serialisierungsformat (ObjectOutputStream).

Die DataSerializer-Plug-ins bieten eine effiziente Methode für die Lösung dieser Probleme. Das DataSerializer-Plug-in bietet Ihnen eine Möglichkeit, Ihr Serialisierungformat oder Byte-Array für WebSphere eXtreme Scale zu beschreiben, sodass das Produkt das Byte-Array abfragen kann, ohne dass ein bestimmtes Objektformat vorausgesetzt wird. Die öffentlichen Klassen und Schnittstellen der DataSerializer-Plug-is sind im Paket "com.ibm.websphere.objectgrid.plugins.io" enthalten. Weitere Informationen finden Sie in der .

Wichtig: Java-Entitätsobjekte werden nicht direkt in den BackingMaps gespeichert, wenn Sie die API "EntityManager" verwenden. Die API "EntityManager" konvertiert die Entitätsobjekte in Tupelobjekte. Entitätsmaps werden automatisch einem hoch optimierten ObjectTransformer zugeordnet. Wenn die API "ObjectMap" oder die API" EntityManager" für die Interaktion mit Entitätsmaps verwendet wird, wird die ObjectTransformer-Entität aufgerufen. Deshalb fällt bei der Verwendung von Entitäten keine Serialisierungsarbeit an, weil das Produkt diesen Prozess automatisch ausführt.

Übersicht über die Skalierbarkeit

WebSphere eXtreme Scale ist über die Verwendung partitionierter Daten skalierbar und kann bei Bedarf auf Tausende von Containern skaliert werden, weil jeder Container von den anderen Containern unabhängig ist.

WebSphere eXtreme Scale teilt Datenbestände in verschiedene Partitionen auf, die zur Laufzeit zwischen Prozessen oder sogar zwischen physischen Servern verschoben werden können. Sie können beispielsweise mit einer Implementierung von vier Servern beginnen und diese dann auf eine Implementierung mit zehn Servern erweitern, wenn der Cachebedarf steigt. So, wie Sie weitere physische Server und Verarbeitungseinheiten für die vertikale Skalierbarkeit hinzufügen können, können Sie die elastischen Horizontalskalierungsfähigkeiten durch Partitionierung erweitern. Die horizontale Skalierung ist einer der Hauptvorteile von WebSphere eXtreme Scale gegenüber einer speicherinternen Datenbank. Speicherinterne Datenbanken können nur vertikal skaliert werden.

Mit WebSphere eXtreme Scale können Sie außerdem eine Gruppe von APIs verwenden, um transaktionsorientiert auf diese partitionierten und verteilten Daten zuzugreifen. In Bezug auf die Leistung sind die Optionen, die Sie für die Interaktion mit dem Cache auswählen, genauso signifikant wie die Funktionen für die Verwaltung des Caches für Verfügbarkeit.

Anmerkung: Skalierbarkeit ist nicht verfügbar, wenn Container miteinander kommunizieren. Das Verfügbarkeitsmanagement- oder Stammgruppenprotokoll ist ein $O(N^2)$-Wartungsalgorithmus für Überwachungssignale und Sichten, wird aber abgeschwächt, indem die Anzahl der Stammgruppen-Member unter 20 gehalten wird. Replikation findet nur im Peer-to-Peer-Modus zwischen Shards statt.

Verteilte Clients

Das Clientprotokoll von WebSphere eXtreme Scale unterstützt eine hohe Anzahl an Clients. Die Partitionierungsstrategie bietet Unterstützung, indem davon ausgegangen wird, dass nicht immer alle Clients an allen Partitionen interessiert sind, weil Verbindungen auf mehrere Container verteilt sein können. Clients werden direkt mit den Partitionen verbunden, sodass die Latenzzeit auf eine übertragene Verbindung beschränkt ist.

Datengrids, Partitionen und Shards

Ein Datengrid ist in Partitionen unterteilt. Eine Partition enthält einen exklusiven Teil der Daten. Eine Partition enthält ein oder mehrere Shards: ein primäres Shard und Replikat-Shards. Replikat-Shards sind in einer Partitin nicht erofrderlich, aber Sie können Replikat-Shards für die Unterstützung der hohen Verfügbarkeit verwenden. Unabhängig davon, ob Ihre Implementierung ein unabhängiges speicher-

internes Datengrid oder ein speicherinterner Datenbankverarbeitungsbereich ist, ist der Datenzugriff in eXtreme Scale im Wesentlichen von den Shard-Konzepten abhängig.

Die Daten für eine Partition werden zur Laufzeit in einer Gruppe von Shards gespeichert. Diese Gruppe von Shards setzt sich aus einem primären Shard und unter Umständen einem oder mehreren Replikat-Shards zusammen. Ein Shard ist die kleinste Einheit, die eXtreme Scale in einer Java Virtual Machine hinzufügen oder entfernen kann.

Es gibt zwei Verteilungsstrategien: feste Partitionsverteilung (Standardeinstellung) und containerbezogene Verteilung. Die folgende Beschreibung konzentriert sich auf die Verwendung der Strategie der festen Partitionsverteilung.

Gesamtanzahl der Shards

Wenn Ihre Umgebung beispielsweise zehn Partitionen mit einer Million Objekten ohne Replikate enthält, sind zehn Shards vorhanden, in denen jeweils 100.000 Objekte gespeichert sind. Wenn Sie diesem Szenario ein Replikat hinzufügen, ist in jeder Partition ein zusätzliches Shard vorhanden. In diesem Fall sind dann 20 Shards vorhanden, zehn primäre Shards und zehn Replikat-Shards. In jedem dieser Shards werden 100.000 Objekte gespeichert. Jede Partition setzt sich aus einem primären Shard und einem oder mehreren (N) Replikat-Shards zusammen. Die Festlegung der optimalen Shard-Anzahl ist ein kritischer Faktor. Wenn Sie eine geringe Anzahl an Shards konfigurieren, werden die Daten nicht gleichmäßig auf die Shards verteilt, was zu Fehlern wegen Speicherengpässen und Problemen durch Prozessüberlastung führt. Sie müssen bei der Skalierung mindestens zehn Shards für jede JVM festlegen. Wenn Sie das Grid implementieren, verwenden Sie möglicherweise eine höhere Anzahl an Partitionen.

Anzahl der Shards pro JVM

Szenario mit einer geringen Anzahl an Shards für jede JVM

Daten werden in einer JVM in Shard-Einheiten hinzugefügt und entfernt. Shards werden nie unterteilt. Wenn 10 GB Daten und 20 Shards zum Speichern dieser Daten vorhanden sind, enthält jedes Shard durchschnittlich 500 MB Daten. Wenn das Datengrid auf neun JVMs verteilt ist, hat jede JVM durchschnittlich zwei Shards. Da 20 nicht durch 9 teilbar ist, haben einige JVMs drei Shards. Die Verteilung ist wie folgt:

- Sieben Java Virtual Machines mit zwei Shards
- Zwei Java Virtual Machines mit drei Shards

Da jedes Shard 500 MB Daten enthält, ist die Verteilung der Daten unsymmetrisch. Die sieben JVMs mit zwei Shards enthalten jeweils 1 GB Daten. Die beiden JVMs mit drei Shards enthalten jeweils 50 % mehr Daten (oder 1,5 GB), was eine sehr viel höhere Speicherlast darstellt. Da die beiden JVMs drei Shards haben, empfangen sie auch 50 % mehr Anforderungen für ihre Daten. Deshalb führt eine geringe Anzahl an Shards für jede JVM zu einem Ungleichgewicht. Zur Verbesserung der Leistung erhöhen Sie die Anzahl der Shards für jede JVM.

Szenario mit einer höheren Anzahl an Shards pro JVM

In diesem Szenario wird eine sehr viel höhere Anzahl an Shards verwendet. Es gibt 101 Shards mit neun JVMs für 10 GB Daten. In diesem Fall werden in jedem Shard 99 MB Daten gespeichert. Die Shards sind wie folgt auf die JVMs verteilt:

- Sieben JVMs mit 11 Shards
- Zwei JVMs mit 12 Shards

Die beiden JVMs mit jeweils 12 Shards enthalten jetzt nur 99 MB mehr Daten als die anderen Shards. Dies ergibt eine Differenz von 9 %. In diesem Szenario ist die Last gleichmäßiger verteilt als in dem Szenario mit einer geringeren Anzahl an Shards, wo die Differenz bei 50 % liegt. Vom Standpunkt der Prozessorauslastung betrachtet, fällt auf die beiden JVMs mit den jeweils 12 Shards im Vergleich mit den sieben JVMs mit jeweils 11 Shards nur 9 % mehr Arbeit. Durch die Erhöhung der Shard-Anzahl in jeder JVM wird die Daten- und Prozessorauslastung fair und gleichmäßig verteilt.

Verwenden Sie 10 Shards für jede JVM, wenn Sie Ihr System in maximaler Größe bzw. die Ausführung der maximalen Anzahl an JVMs in Ihrem System planen.

Zusätzliche Verteilungsfaktoren

Die Anzahl der Partitionen, die Verteilungsstrategie und der Typ der Replikate wird in der Implementierungsrichtlinie definiert. Die Anzahl der verteilten Shards richtet sich nach der definierten Implementierungsrichtlinie. Die Attribute "minSyncReplicas", "developmentMode", "maxSyncReplicas" und "maxAsyncReplicas" wirken sich auf die Verteilung von Partitionen und Replikaten aus.

Die folgenden Faktoren haben Auswirkungen auf den Zeitpunkt der Shard-Verteilung:

- Befehle **xscmd -c suspendBalancing** und **xscmd -c resumeBalancing**
- Servereigenschaftendatei, die die Eigenschaft **placementDeferralInterval** enthält, die die Anzahl der Millisekunden definiert, bevor Shards auf die Container-Server verteilt werden
- Attribut **numInitialContainers** in der Implementierungsrichtlinie

Wenn beim Erststart nicht die maximale Anzahl an Replikaten verteilt wird, können weitere Replikate verteilt werden, wenn Sie später weitere Server starten. Bedenken Sie bei der Planung der Shard-Anzahl pro JVM darauf, dass die maximale Anzahl an primären Shards und Replikat-Shards davon abhängig ist, dass genügend JVMs fr die Unterstützung der maximalen Anzahl an Replikaten verfügbar sind. Ein Replikat wird niemals in demselben Prozess wie das primäre Shard ausgeführt. Wenn ein Prozess verloren geht, würden sonst das primäre Shard und das Replikat-Shard verloren gehen. Wenn das Attrbut "developmentMode" auf `false` gesetzt ist, werden die primären Shards und die Replikat-Shards nicht an denselben physischen Server verteilt.

Partitionierung

Partitionierung für das Scale-out einer Anwendung verwenden. Sie können die Anzahl der Partitionen in Ihrer Implementierungsrichtlinie definieren.

Informationen zur Partitionierung

Partitionierung ist nicht dasselbe wie RAID-Striping (Redundant Array of Independent Disks), bei dem Teile jeder Instanz auf alle Stripes verteilt werden. Jede Partition enthält die vollständigen Daten für einzelne Einträge. Partitionierung ist ein

effektives Mittel für Skalierung, aber nicht auf alle Anwendungen anwendbar. Anwendungen, die Transaktionsgarantien über große Datengruppen hinweg erfordern, können nicht skaliert und nicht effizient partitioniert werden. Deshalb unterstützt WebSphere eXtreme Scale derzeit keine partitionsübergreifende zweiphasige Festschreibung.

Wichtig: Wählen Sie die Anzahl der Partitionen sorgfältig aus. Die in der Implementierungsrichtlinie definierte Partitionsanzahl wirkt sich direkt auf die Anzahl der Container-Server aus, auf die eine Anwendung skaliert werden kann. Jede Partition setzt sich aus einem primären Shard und der konfigurierten Anzahl an Replikat-Shards zusammen. Mit der Formel `(Anzahl_Partitionen*(1 + Anzahl_Replikate))` wird die Anzahl der Container berechnet, die verwendet werden kann, um eine einzelne Anwendung horizontal zu skalieren.

Partitionen verwenden

Ein Datengrid kann Tausende von Partitionen haben. Ein Datengrid kann vertikal bis auf das Produkt aus Partitionsanzahl und Shard-Anzahl pro Partition skaliert werden. Wenn Sie beispielsweise 16 Partitionen haben und jede Partition ein einziges primäres Shard oder zwei Shards hat, können Sie potenziell eine Skalierung auf bis 32 Java Virtual Machines erreichen. In diesem Fall wird für jede JVM ein einziges Shard definiert. Sie müssen, basierend auf der erwarteten Anzahl an Java Virtual Machines, die wahrscheinlich verwendet werden, eine angemessene Anzahl an Partitionen auswählen. Mit jedem Shard erhöht sich die Prozessor- und Speicherbelegung für das System. Das System ist so konzipiert, dass es horizontal skaliert werden kann, um diesen Aufwand entsprechend der Anzahl verfügbarer Java Virtual Machines des Servers handhaben zu können.

Anwendungen sollten nicht Tausende von Partitionen verwenden, wenn die Anwendung in einem Datengrid mit vier Container-Server-JVMs ausgeführt wird. In der Konfiguration der Anwendung sollte eine angemessene Anzahl an Shards für jede Container-Server-JVM angegeben werden. Eine unverhältnismäßige Konfiguration sind beispielsweise 2000 Partitionen mit zwei Shards, die in vier Container-JVMs ausgeführt werden. Diese Konfiguration würde bedeuten, dass 4000 Shards auf vier Container-JVMs bzw. 1000 Shards pro Container-JVM verteilt werden.

Eine bessere Konfiguration wären 10 Shards für jede erwartetet Container-JVM. Diese Konfiguration bietet Ihnen trotzdem die Möglichkeit einer elastischen Skalierung bis hin zum Zehnfachen der Erstkonfiguration bei einer gleichzeitig angemessenen Anzahl an Shards pro Container-JVM.

Stellen Sie sich das folgende Skalierungsbeispiel vor: Sie haben derzeit sechs physische Server mit jeweils zwei Container-JVMs. Sie erwarten ein Wachstum auf 20 physische Server in einem Zeitraum von drei Jahren. Mit 20 physischen Servern haben Sie 40 Container-Server-JVMs, und Sie wählen 60 aus, um pessimistisch zu sein. Sie möchten vier Shards pro Container-JVM haben. Sie haben 60 potenzielle Container bzw. insgesamt 240 Shards. Wenn Sie pro Partition ein primäres Shard und ein Replikat-Shard haben, möchten Sie 120 Partitionen haben. Diese Beispielrechnung ergibt 240, geteilt durch 12 Container-JVMs, bzw. 20 Shards pro Container-JVM für die Erstimplementierung mit der Möglichkeit einer späteren Skalierung auf 20 Computer.

ObjectMap und Partitionierung

Bei der Verteilungsstrategie FIXED_PARTITION werden Maps auf Partitionen und die Hash-Werte von Schlüsseln auf verschiedene Partitionen verteilt. Der Client

muss nicht wissen, zu welcher Partition die Schlüssel gehören. Wenn ein MapSet mehrere Maps hat, müssen die Maps in separaten Transaktionen festgeschrieben werden.

Entitäten und Partitionierung

EntityManager-Entitäten besitzen eine Optimierung, die Clients hilft, die mit Entitäten in einem Server arbeiten. Das Entitätsschema im Server für das MapSet kann eine einzige Stammentität spezifizieren. Der Client muss auf alle Entitäten über diese Stammentität zugreifen. Der EntityManager findet dann die zugehörigen Entitäten über diese Stammentität, ohne dass die zugehörigen Maps einen gemeinsamen Schlüssel haben müssen. Die Stammentität stellt die Affinität zu einer einzelnen Partition her. Diese Partition wird nach der Herstellung der Affinität für all Entitätsabrufe innerhalb der Transaktion verwendet. Diese Affinität kann zu Speichereinsparungen führen, weil die zugehörigen Maps keinen gemeinsamen Schlüssel benötigen. Die Stammentität muss mit einer modifizierten Entitätsannotation angegeben werden, wie im folgenden Beispiel gezeigt wird:

```
@Entity(schemaRoot=true)
```

Verwenden Sie die Entität, um das Stammelement des Objektgraphen zu ermitteln. Der Objektgraph definiert die Beziehungen zwischen Entitäten. Jede verbundene Entität muss in dieselbe Partition aufgelöst werden. Es wird davon ausgegangen, dass sich alle untergeordneten Elemente in derselben Partition wie das Stammelement befinden. Die untergeordneten Entitäten im Objektgraphen sind von einem Client aus nur über die Stammentität zugänglich. Stammentitäten sind in partitionierten Umgebungen immer erforderlich, wenn ein Client von eXtreme Scale für die Kommunikation mit dem Server verwendet wird. Es kann nur ein einziger Stammentitätstyp pro Client definiert werden. Stammentitäten sind nicht erforderlich, wenn XTP-ObjectGrids (Extreme Transaction Processing) verwendet werden, da die gesamte Kommunikation mit der Partition über direkten lokalen Zugriff und nicht über den Client/Server-Mechanismus erfolgt.

Verteilung und Partitionen

Sie können zwischen zwei Verteilungsstrategien für WebSphere eXtreme Scale wählen: feste Partitionsverteilung und containerbezogene Verteilung. Die Auswahl der Verteilungsstrategie hat Einfluss darauf, wie die Implementierungskonfiguration Partitionen im fernen Datengrid verteilt.

Feste Partitionsverteilung

Sie können die Verteilungsstrategie in der XML-Datei für Implementierungsrichtlinien festlegen. Die Standardverteilungsstrategie ist die feste Partitionsverteilung, die mit der Einstellung `FIXED_PARTITION` aktiviert wird. Die Anzahl primärer Shards, die auf die verfügbaren Container verteilt werden, entspricht der Anzahl der Partitionen, die Sie mit dem Attribut "numberOfPartitions" konfiguriert haben. Wenn Sie Replikate konfiguriert haben, wird die minimale Gesamtanzahl der verteilten Shards mit der folgenden Formel definiert: `((1 primäres Shard + Mindestanzahl synchroner Shards) * Anzahl definierter Partitionen)`. Die maximale Gesamtanzahl verteilter Shards wird mit der folgenden Formel definiert: `((1 primäres Shard + maximale Anzahl synchroner Shards + maximale Anzahl asynchroner Shards) * Partitionen)`. Ihre Implementierung von WebSphere eXtreme Scale verteilt die Shards auf die verfügbaren Container. Die Schlüssel jeder Map werden, basierend auf der definierten Gesamtanzahl der Partitionen, in den zugeordneten Partitionen verschlüsselt. Die Schlüssel werden auch dann in dersel-

ben Partition verschlüsselt, wenn die Partition aufgrund eines Failovers oder aufgrund von Serveränderungen verschoben wird.

Wenn numberPartitions beispielsweise den Wert 6 und minSync den Wert 1 für MapSet1 hat, ist die Gesamtanzahl der Shards für dieses MapSet 12, weil jede der 6 Partitionen ein synchrones Replikat erfordert. Wenn drei Container gestartet sind, verteilt WebSphere eXtreme Scale vier Shards pro Container für MapSet1.

Containerbezogene Verteilung

Die alternative Verteilungsstrategie ist die containerbezogene Verteilung, die mit der Einstellung `PER_CONTAINER` für das Attribut "placementStrategy" im MapSet-Element in der XML-Implementierungsdatei aktiviert wird. Bei dieser Strategie entspricht die Anzahl primärer Shards, die an jeden neuen Container verteilt wird, der Anzahl der Partitionen, *P*, die Sie konfiguriert haben. Die Implementierungsumgebung von WebSphere eXtreme Scale verteilt *P* Replikate jeder Partition für jeden verbleibenden Container. Die Einstellung von numInitialContainers wird ignoriert, wenn Sie die containerbezogene Verteilung verwenden. Die Partitionen werden größer, wenn die Container zunehmen. Die Schlüssel für Maps sind bei dieser Strategie nicht auf eine bestimmte Partition festgelegt. Der Client leitet Anforderungen an eine Partition weiter und verwendet eine wahlfreie primäre Partition. Wenn in Client die Verbindung zu derselben Sitzung wiederherstellen möchte, die er für die erneute Suche eines Schlüssels verwendet hat, müssen Sie ein Sitzungs-Handle verwenden.

Weitere Informationen finden Sie im Abschnitt zur Verwendung eines Session-Handle für die Weiterleitung in der Veröffentlichung *Programmierung*.

Wenn ein Failover stattfindet oder Server gestoppt werden, verschiebt die Umgebung von WebSphere eXtreme Scale die primären Shards in der containerbezogenen Verteilungsstrategie, wenn diese noch Daten enthalten. Wenn die Shards leer sind, werden sie verworfen. Bei der containerbezogenen Strategie werden alte primäre Shards nicht beibehalten, weil neue primäre Shards für jeden Container verteilt werden.

WebSphere eXtreme Scale lässt die containerbezogene Verteilung als Alternative zur so genannten "typischen" Verteilung zu, einer Lösung mit festgelegten Partitionen, bei der der Schlüssel einer Map verschlüsselt auf einer dieser Partitionen gespeichert wird. Im containerbezogenen Fall (den Sie mit PER_CONTAINER festlegen) verteilt die Implementierung die Partitionen auf die Gruppe der online befindlichen Container-Server und skaliert diese automatisch, wenn Container dem Datengrid des Servers hinzugefügt bzw. aus diesem entfernt werden. Ein Datengrid mit festen Partitionen eignet sich für schlüsselbasierte Grids, bei denen die Anwendung ein Schlüsselobjekt verwendet, um die Daten im Grid zu suchen. Die Alternative wird im Folgenden beschrieben.

Beispiel für ein containerbezogenes Datengrid

Datengrids mit der Verteilungsstrategie PER_CONTAINER sind anders. Sie legen die Verteilungsstrategie PER_CONTAINER für das Datengrid mit dem Attribut "placementStrategy" in der XML-Implementierungsdatei fest. Anstatt die gewünschte Gesamtpartitionsanzahl im Datengrid zu konfigurieren, geben Sie an, wie viele Partitionen Sie pro gestartetem Container wünschen.

Wenn Sie beispielsweise fünf Partitionen pro Container definieren, werden fünf neue anonyme primäre Partitions-Shards erstellt, wenn Sie diesen Container-Server starten, und die erforderlichen Replikate werden in den anderen implementierten Container-Servern erstellt.

Im Folgenden sehen Sie eine mögliche Folge in einer containerbezogenen Umgebung beim Anwachsen des Datengrids.

1. Start von Container C0 mit 5 primären Shards (P0-P4)
   - C0 enthält P0, P1, P2, P3, P4
2. Start von Container C1 mit 5 weiteren primären Shards (P5-P9). Die Replikat-Shards werden gleichmäßig auf die Container verteilt.
   - C0 enthält: P0, P1, P2, P3, P4, R5, R6, R7, R8, R9
   - C1 enthält: P5, P6, P7, P8, P9, R0, R1, R2, R3, R4
3. Start von Container C2 mit 5 weiteren primären Shards (P10-P14). Die Replikat-Shards werden gleichmäßig neu verteilt.
   - C0 enthält: P0, P1, P2, P3, P4, R7, R8, R9, R10, R11, R12
   - C1 enthält: P5, P6, P7, P8, P9, R2, R3, R4, R13, R14
   - C2 enthält: P10, P11, P12, P13, P14, R5, R6, R0, R1

Das Muster wird fortgesetzt, wenn weitere Container gestartet werden. Bei jedem Start eines weiteren Containers werden fünf neue primäre Partitionen erstellt, und die Replkate werden gleichmäßig neu auf die verfügbaren Container im Datengrid verteilt.

Anmerkung: WebSphere eXtreme Scale verschiebt primäre Shards bei der Strategie PER_CONTAINER nicht, es werden nur Replikate verschoben.

Denken Sie daran, dass die Partitionsnummern beliebig sind, d. h. keinen Bezug zu Schlüsseln haben, und deshalb schlüsselbasiertes Routing nicht verwendet werden kann. Wenn ein Container gestoppt wird, werden die erstellten Partitions-IDs für diesen Container nicht mehr verwendet, und es entsteht eine Lücke in den Partitions-IDs. Wenn Container C2 in dem Beispiel ausfällt, sind die Partitionen P5-P9 nicht mehr verfügbar. Es bleiben nur die Partitionen P0-P4 und P10-P14 übrig. Somit ist ein schlüsselbasiertes Hashing nicht möglich.

Die Verwendung von Zahlen wie fünf oder 10 (was wahrscheinlicher ist) für die Anzahl der Partitionen pro Container empfiehlt sich, wenn man an die Auswirkungen eines Containerausfalls denkt. Damit die Last der Host-Shards gleichmäßig im Datengrid verteilt wird, benötigen Sie mehr als nur eine Partition pro Container. Wenn Sie nur eine einzige Partition pro Container verwenden und ein Container ausfällt, muss ein einziger Container (der mit dem entsprechenden Replikat-Shard) die volle Last des ausgefallenen primären Shards tragen. In diesem Fall verdoppelt sich sofort die Last für den Container. Wenn Sie jedoch fünf Partitionen pro Container haben, übernehmen fünf Container die Last des ausgefallenen Containers, was die Auswirkungen auf jeden Container um 80 Prozent reduziert. Die Verwendung mehrerer Partitionen pro Container verringert die potenziellen Auswirkungen auf jeden Container im Allgemeinen erheblich. Stellen Sie sich einen Fall vor, in dem die Last eines Containers unerwartet stark ansteigt. Die Replikationslast dieses Containers wird auf 5 Container und nicht nur auf einen einzigen verteilt.

Containerbezogene Richtlinie verwenden

Es gibt verschiedene Szenarien, in denen die containerbezogene Strategie eine ideale Konfiguration ist, z. B. bei der Replikation von HTTP-Sitzungen oder beim Sta-

tus von Anwendungssitzungen. In einem solchen Fall ordnet ein HTTP-Router eine Sitzung einem Servlet-Container zu. Der Servlet-Container muss eine HTTP-Sitzung erstellen und wählt eines der fünf lokalen primären Partitions-Shards für die Sitzung auswählen. Die "ID" der ausgewählten Partition wird anschließend in einem Cookie gespeichert. Der Servlet-Container hat jetzt lokalen Zugriff auf den Sitzungsstatus, was einen Zugriff ohne Latenzzeit auf die Daten für diese Anforderung bedeutet, solange die Sitzungsaffinität aufrecht erhalten bleibt. eXtreme Scale repliziert alle Änderungen an der Partition.

Stellen sich die Auswirkungen eines Falls in der Praxis vor, in dem Sie mehrere Partitionen pro Container (sagen wir erneut 5) haben. Natürlich haben Sie mit jedem neuen Container, der gestartet wird, 5 weitere primäre Partitions-Shards und 5 weitere Replikat-Shards. Mit der Zeit müssen weitere Partitionen erstellt werden, und diese dürfen weder verschoben noch entfernt werden. So verhalten sich die Container aber in Wirklichkeit nicht. Wenn ein Container gestartet wird, enthält er 5 primäre Shards, die so genannten primären "Ausgangs-Shards". Wenn der Container ausfällt, werden die Replikate zu primären Shards,und eXtreme Scale erstellt 5 weitere Replikate, um die hohe Verfügbarkeit aufrecht zu erhalten (sofern Sie die automatische Reparatur nicht inaktivieren). Die neuen primären Shards befinden sich in einem anderen Container als dem, von dem sie erstellt wurden, und werden deshalb als "fremde" primäre Shards bezeichnet. Die Anwendung sollte neue Statusinformationen oder Sitzungen nie in einem fremden primären Shards speichern. Das fremde primäre Shard hat irgendwann keine Einträge mehr, und eXtreme Scale löscht dann dieses fremde Shard und die zugehörigen Replikate. Die fremden primären Shards unterstützen die weitere Verfügbarkeit vorhandener Sitzungen (aber keine neuen Sitzungen).

Ein Client kann weiterhin mit einem Datengrid interagieren, das nicht schlüsselbasiert ist. Der Client startet eine Transaktion und speichert Daten im Datengrid unabhängig von Schlüsseln. Er fordert bei der Sitzung ein SessionHandle-Objekt an, ein serialisierbares Handle, das dem Client bei Bedarf die Interaktion mit derselben Partition ermöglicht. Weitere Informationen finden Sie im Abschnitt zur Verwendung eines SessionHandle für das Routing im *Programmierhandbuch*. WebSphere eXtreme Scale wählt eine Partition für den Client aus der Liste der primären Ausgangspartitions-Shards aus. Er gibt keine fremde primäre Partition zurück. Das SessionHandle kann beispielsweise in ein HTTP-Cookie serialisiert werden, das später wieder in ein Cookie konvertiert wird. Anschließend können die APIs von WebSphere eXtreme Scale eine Sitzung abrufen, die über das SessionHandle wieder an dieselbe Partition gebunden wird.

Anmerkung: Für die Interaktion mit einem PER_CONTAINER-Datengrid können keine Agenten verwendet werden.

Vorteile

Die vorherige Beschreibung weicht von einem normalen FIXED_PARTITION- oder Hash-Datengrid ab, weil der containerbezogene Client Daten an einer Stelle im Grid speichert, ein Handle für die Daten abruft und das Handle verwendet, um erneut auf die Daten zuzugreifen. Es gibt keinen von der Anwendung bereitgestellten Schlüssel wie im Fall mit festgelegten Partitionen.

Ihre Implementierung erstellt keine neue Partition für jede Sitzung. In einer containerbezogenen Implementierung müssen die Schlüssel, die zum Speichern von Daten in der Partition verwendet werden, deshalb innerhalb dieser Partition eindeutig sein. Sie lassen von Ihrem Client beispielsweise eine eindeutige Sitzungs-ID erstellen und verwenden diese als Schlüssel, um Informationen in den Maps dieser Par-

tition zu suchen. Anschließend interagieren mehrere Clientsitzungen mit derselben Partition, sodass die Anwendung eindeutige Schlüssel verwenden muss, um Sitzungsdaten in jeder angegebenen Partition zu speichern.

In den vorherigen Beispielen wurden 5 Partitionen verwendet, aber der Parameter "numberOfPartitions" in der ObjectGrid-XML-Datei kann verwendet werden, um die gewünschte Anzahl an Partitionen anzugeben. Die Einstellung gilt nicht pro Datengrid, sondern pro Container. (Die Anzahl der Replikate wird wie bei der Richtlinie für festgelegte Partitionen angegeben.)

Die containerbezogene Richtlinie kann auch für mehrere Zonen verwendet werden. Wenn möglich, gibt eXtreme Scale ein SessionHandle für eine Partition zurück, deren primäres Shard sich in derselben Zone wie dieser Client befindet. Der Client kann die Zone als Parameter für den Container oder über eine API angeben. Die Clientzonen-ID kann mit `serverproperties` oder `clientproperties` festgelegt werden.

Die PER_CONTAINER-Strategie für ein Datengrid eignet sich für Anwendungen, die dialogbasierte Statusinformationen anstelle von datenbankorientierten Daten speichern. Der Schlüssel für den Zugriff auf die Daten ist eine Dialog-ID und bezieht sich nicht auf einen bestimmten Datenbanksatz. Diese Strategie bietet eine höhere Leistung (weil die primären Partitions-Shards beispielsweise mit den Servlets zusammengefasst werden können) und eine einfachere Konfiguration (ohne Partitionen und Container berechnen zu müssen).

Einzelpartitionstransaktionen und datengridübergreifende Partitionstransaktionen

Java

Der Hauptunterschied zwischen WebSphere eXtreme Scale und traditionellen Datenspeicherlösungen wie relationalen oder speicherinternen Datenbanken ist die Verwendung der Partitionierung, die eine lineare Skalierung des Caches ermöglicht. Die wichtigen Transaktionstypen, die berücksichtigt werden müssen, sind Einzelpartitionstransaktionen und datengridübergreifende Partitionstransaktionen.

Im Allgemeinen können Interaktionen mit dem Cache, wie im folgenden Abschnitt beschrieben, in die Kategorien "Einzelpartitionstransaktionen" und "Datengridübergreifende Partitionstransaktionen" eingeteilt werden.

Einzelpartitionstransaktionen

Einzelpartitionstransaktionen sind die vorzuziehende Methode für die Interaktion mit Caches in WebSphere eXtreme Scale. Wenn eine Transaktion auf eine Einzelpartition beschränkt ist, ist sie standardmäßig auf eine einzelne Java Virtual Machine und damit auf einen einzelnen Servercomputer beschränkt. Ein Server kann *M* dieser Transaktionen pro Sekunde ausführen, und wenn Sie *N* Computer haben, sind `M*N` Transaktionen pro Sekunde möglich. Wenn sich Ihr Geschäft erweitert und Sie doppelt so viele dieser Transaktionen pro Sekunde ausführen müssen, können Sie *N* verdoppeln, indem Sie weitere Computer kaufen. Auf diese Weise können Sie Kapazitätsanforderungen erfüllen, ohne die Anwendung zu ändern, Hardware zu aktualisieren oder die Anwendung außer Betrieb zu nehmen.

Zusätzlich zu der Möglichkeit, den Cache so signifikant skalieren zu können, maximieren Einzelpartitionstransaktionen auch die Verfügbarkeit des Caches. Jede Transaktion ist nur von einem einzigen Computer abhängig. Jeder der anderen

(N-1) Computer kann ausfallen, ohne den Erfolg oder die Antwortzeit der Transaktion zu beeinflussen. Wenn Sie also mit 100 Computern arbeiten und einer dieser Computer ausfällt, wird nur 1 Prozent der Transaktionen, die zum Zeitpunkt des Ausfalls dieses Servers unvollständig sind, rückgängig gemacht. Nach dem Ausfall des Servers verlagert WebSphere eXtreme Scale die Partitionen des ausgefallenen Servers auf die anderen 99 Computer. In diesem kurzen Zeitraum vor der Durchführung der Operation können die anderen 99 Computer weiterhin Transaktionen ausführen. Nur die Transaktionen, an denen die Partitionen beteiligt sind, die umgelagert werden, sind blockiert. Nach Abschluss des Failover-Prozesses ist der Cache mit 99 Prozent seiner ursprünglichen Durchsatzkapazität wieder vollständig betriebsbereit. Nachdem der ausgefallene Server ersetzt und der Ersatzserver dem Datengrid hinzugefügt wurde, kehrt der Cache zu einer Durchsatzkapazität von 100 Prozent zurück.

Datengridübergreifende Transaktionen

Was Leistung, Verfügbarkeit und Skalierbarkeit betrifft, sind datengridübergreifende Transaktionen das Gegenteil von Einzelpartitionstransaktionen. Datengridübergreifende Transaktionen greifen auf jede Partition und damit auf jeden Computer in der Konfiguration zu. Jeder Computer im Datengrid wird aufgefordert, einige Daten zu suchen und anschließend das Ergebnis zurückzugeben. Die Transaktion kann erst abgeschlossen werden, nachdem jeder Computer geantwortet hat, und deshalb wird der Durchsatz des gesamten Datengrids durch den langsamsten Computer beschränkt. Das Hinzufügen von Computern macht den langsamsten Computer nicht schneller und verbessert damit auch nicht den Durchsatz des Caches.

Datengridübergreifende Transaktionen haben einen ähnlichen Effekt auf die Verfügbarkeit. Wenn Sie mit 100 Servern arbeiten und ein Server ausfällt, werden 100 Prozent der Transaktionen, die zum Zeitpunkt des Serverausfalls in Bearbeitung sind, rückgängig gemacht. Nach dem Ausfall des Servers beginnt WebSphere eXtreme Scale mit der Verlagerung der Partitionen des ausgefallenen Servers auf die anderen 99 Computer. In dieser Zeit, d. h. bis zum Abschluss des Failover-Prozesses, kann das Datengrid keine dieser Transaktionen verarbeiten. Nach Abschluss des Failover-Prozesses ist der Cache wieder betriebsbereit, aber mit verringerter Kapazität. Wenn jeder Computer im Datengrid 10 Partitionen bereitstellt, erhalten 10 der verbleibenden 99 Computer während des Failover-Prozesses mindestens eine zusätzliche Partition. Eine zusätzliche Partition erhöht die Arbeitslast dieses Computers um mindestens 10 Prozent. Da der Durchsatz des Datengrids in einer datengridübergreifenden Transaktion auf den Durchsatz des langsamsten Computers beschränkt ist, reduziert sich der Durchsatz durchschnittlich um 10 Prozent.

Einzelpartitionstransaktionen sind im Hinblick auf die horizontale Vorwärtsskalierung (Scale-out) mit einem verteilten, hoch verfügbaren Objektcache wie WebSphere eXtreme Scale den datengridübergreifenden Transaktionen vorzuziehen. Die Maximierung der Leistung solcher Systemtypen erfordert die Verwendung von Techniken, die sich von den traditionellen relationalen Verfahren unterscheiden, aber Sie datengridübergreifende Transaktionen in skalierbare Einzelpartitionstransaktionen konvertieren.

Bewährte Verfahren für die Erstellung skalierbarer Datenmodelle

Die bewährten Verfahren für die Erstellung skalierbarer Anwendungen mit Produkten wie WebSphere eXtreme Scale sind in zwei Kategorien einteilbar: Grundsätze und Implementierungstipps. Grundsätze sind Kernideen, die im Design der Daten selbst erfasst werden müssen. Es ist sehr unwahrscheinlich, dass sich eine

Anwendung, die diese Grundsätze nicht einhält, problemlos skalieren lässt, selbst für ihre Haupttransaktionen. Implementierungstipps werden für problematische Transaktionen in einer ansonsten gut entworfenen Anwendung angewendet, die sich an die allgemeinen Grundsätze für skalierbare Datenmodelle hält.

Grundsätze

Einige wichtige Hilfsmittel für die Optimierung der Skalierbarkeit sind Basiskonzepte oder Grundsätze, die beachtet werden müssen.

Duplizieren an Stelle von Normalisieren

Der wichtigste Punkt, der bei Produkten wie WebSphere eXtreme Scale zu beachten ist, ist der, dass sie für die Verteilung von Daten auf sehr viele Computer konzipiert sind. Wenn das Ziel darin besteht, die meisten oder sogar alle Transaktionen auf einer einzelnen Partition auszuführen, muss das Datenmodelldesign sicherstellen, dass sich alle Daten, die die Transaktion unter Umständen benötigt, auf der Partition befinden. In den meisten Fällen kann dies nur durch Duplizierung der Daten erreicht werden.

Stellen Sie sich beispielsweise eine Anwendung wie ein Nachrichtenbrett vor. Zwei sehr wichtige Transaktionen für ein Nachrichtenbrett zeigen alle Veröffentlichungen eines bestimmten Benutzers und alle Veröffentlichungen unter einem bestimmten Topic an. Stellen Sie sich zunächst vor, wie diese Transaktionen mit einem normalisierten Datenmodell arbeiten, das einen Benutzerdatensatz, einen Topic-Datensatz und einen Veröffentlichungsdatensatz mit dem eigentlichen Text enthält. Wenn Veröffentlichungen mit Benutzerdatensätzen partitioniert werden, wird aus der Anzeige des Topics eine gridübergreifende Transaktion und umgekehrt. Topics und Benutzer können nicht gemeinsam partitioniert werden, da sie eine Viele-zu-viele-Beziehung haben.

Die beste Methode für die Skalierung dieses Nachrichtenbretts ist die Duplizierung der Veröffentlichungen, wobei eine Kopie mit dem Topic-Datensatz und eine Kopie mit dem Benutzerdatensatz gespeichert wird. Die anschließende Anzeige der Veröffentlichungen eines Benutzers ist eine Einzelpartitionstransaktion, die Anzeige der Veröffentlichungen unter einem Topic ist eine Einzelpartitionstransaktion, und die Aktualisierung oder das Löschen einer Veröffentlichung ist eine Transaktion, an der zwei Partitionen beteiligt sind. Alle drei Transaktionen können linear skaliert werden, wenn die Anzahl der Computer im Datengrid zunimmt.

Skalierbarkeit an Stelle von Ressourcen

Die größte Hindernis, das beim Einsatz denormalisierter Datenmodell überwunden werden muss, sind die Auswirkungen, die diese Modell auf Ressourcen haben. Die Verwaltung von zwei, drei oder mehr Kopien derselben Daten kann den Anschein erwecken, dass zu viele Ressourcen benötigt werden, als dass dieser Ansatz praktikabel ist. Wenn Sie mit diesem Szenario konfrontiert werden, berücksichtigen Sie die folgenden Fakten: Hardwareressourcen werden von Jahr zu Jahr billiger. Zweitens, und noch wichtiger, mit WebSphere eXtreme Scale fallen die meisten verborgenen Kosten weg, die bei der Implementierung weiterer Ressourcen anfallen.

Messen Sie Ressourcen anhand der Kosten und nicht anhand von Computerbegriffen wie Megabyte oder Prozessoren. Datenspeicher, die mit normalisierten relationalen Daten abreiten, müssen sich im Allgemeinen auf demselben Computer befinden. Diese erforderliche Co-Location bedeutet, dass ein einzelner größerer Unternehmenscomputer an Stelle mehrerer kleinerer

Computer erworben werden muss. Bei Unternehmenshardware ist es nicht unüblich, dass ein einziger Computer, der in der Lage ist, eine Million Transaktionen pro Sekunde zu verarbeiten, mehr kostet als 10 Computer zusammen, die in der Lage sind, jeweils 100.000 Transaktionen pro Sekunden auszuführen.

Außerdem fallen Geschäftskosten für die Implementierung der Ressourcen an. Irgendwann reicht die Kapazität in einem expandierenden Unternehmen einfach nicht mehr aus. In diesem Fall setzen Sie den Betrieb entweder aus, während Sie die Umstellung auf einen größeren und schnelleren Computer durchführen, oder Sie erstellen eine zweite Produktionsumgebung, auf die Sie den Betrieb dann umstellen können. In beiden Fällen fallen zusätzliche Kosten durch das ausgefallene Geschäft oder durch die Verwaltung der doppelten Kapazität in der Übergangsphase an.

Mit WebSphere eXtreme Scale muss die Anwendung nicht heruntergefahren werden, um Kapazität hinzuzufügen. Wenn die Prognose für Ihr Geschäft lautet, dass Sie 10 Prozent mehr Kapazität für das kommende Jahr benötigen, erhöhen Sie die Anzahl der Computer im Datengrid um 10 Prozent. Sie können diese Erweiterung ohne Anwendungsausfallzeit und ohne den Einkauf von Kapazitäten durchführen, die Sie hinterher nicht mehr benötigen.

Datenkonvertierungen vermeiden

Wenn Sie WebSphere eXtreme Scale verwenden, müssen Daten in einem Format gespeichert werden, das von der Geschäftslogik direkt konsumiert werden kann. Die Aufteilung der Daten in ein primitiveres Format ist kostenintensiv. Die Konvertierung muss durchgeführt werden, wenn die Daten geschrieben und wenn die Daten gelesen werden. Mit relationalen Datenbanken ist diese Konvertierung unumgänglich, weil die Daten letztendlich relativ häufig auf der Platte gespeichert werden, aber mit WebSphere eXtreme Scale fallen diese Konvertierungen weg. Der größte Teil der Daten wird im Hauptspeicher gespeichert und kann deshalb in genau dem Format gespeichert werden, das die Anwendung erfordert.

Durch die Einhaltung dieser einfachen Regel können Sie Ihre Daten dem ersten Grundsatz entsprechend denormalisieren. Der gängigste Konvertierungstyp für Geschäftsdaten ist die JOIN-Operation, die erforderlich ist, um normalisierte Daten in eine Ergebnismenge zu konvertieren, die den Anforderungen der Anwendung entspricht. Durch die implizite Speicherung der Daten im richtigen Format werden diese JOIN-Operationen vermieden, und es entsteht ein denormalisiertes Datenmodell.

Unbegrenzte Abfragen vermeiden

Unbegrenzte Abfragen lassen sich nicht gut skalieren, egal, wie gut Sie Ihre Daten auch strukturieren. Verwenden Sie beispielsweise keine Transaktion, die eine Liste aller Einträge nach Wert sortiert abfragt. Diese Transaktion funktioniert möglicherweise, wenn die Gesamtanzahl der Einträge bei 1000 liegt, aber wenn die die Gesamtanzahl der Einträge 10 Million erreicht, gibt die Transaktion alle 10 Millionen Einträge zurück. Wenn Sie diese Transaktion ausführen, sind zwei Ergebnisse am wahrscheinlichsten: Die Transaktion überschreitet das zulässige Zeitlimit, oder im Client tritt eine abnormale Speicherbedingung auf.

Die beste Option ist, die Geschäftslogik so zu ändern, dass nur die Top 10 oder 20 Einträge zurückgegeben werden können. Durch diese Änderung der Logik bleibt die Größe der Transaktion verwaltbar, unabhängig davon, wie viele Einträge im Cache enthalten sind.

Schema definieren

Der Hauptvorteil der Normalisierung von Daten ist der, dass sich das Datenbanksystem im Hintergrund um die Datenkonsistenz kümmern kann. Wenn Daten für Skalierbarkeit denormalisiert werden, ist diese automatische Verwaltung der Datenkonsistenz nicht mehr möglich. Sie müssen ein Datenmodell implementieren, das auf der Anwendungsebene oder als Plug-in für das verteilte Datengrid arbeiten kann, um die Datenkonsistenz zu gewährleisten.

Stellen Sie sich das Beispiel mit dem Nachrichtenbrett vor. Wenn eine Transaktion eine Veröffentlichung aus einem Topic entfernt, muss das Veröffentlichungsduplikat im Benutzerdatensatz entfernt werden. Ohne ein Datenmodell ist es möglich, dass ein Entwickler den Anwendungscode zum Entfernen der Veröffentlichung aus dem Topic schreibt und vergisst, die Veröffentlichung aus dem Benutzerdatensatz zu entfernen. Wenn der Entwickler jedoch ein Datenmodell verwendet, anstatt direkt mit dem Cache zu interagieren, kann die Methode "removePost" im Datenmodell die Benutzer-ID aus der Veröffentlichung extrahieren, den Benutzerdatensatz suchen und das Veröffentlichungsduplikat im Hintergrund entfernen.

Alternativ können Sie einen Listener implementieren, der auf der tatsächlichen Partition ausgeführt wird, das Topic überwacht und bei einer Änderung des Topics Benutzerdatensatz automatisch anpasst. Ein Listener kann von Vorteil sein, weil die Anpassung am Benutzerdatensatz lokal vorgenommen werden kann, wenn die Partition den Benutzerdatensatz enthält. Selbst wenn sich der Benutzerdatensatz auf einer anderen Partition befindet, findet die Transaktion zwischen Servern und nicht zwischen dem Client und dem Server statt. Die Netzverbindung zwischen Servern ist wahrscheinlich schneller als die Netzverbindung zwischen dem Client und dem Server.

Konkurrenzsituationen vermeiden

Szenarien wie die Verwendung eines globalen Zählers vermeiden. Das Datengrid kann nicht skaliert werden, wenn ein einzelner Datensatz im Vergleich mit den restlichen Datensätzen unverhältnismäßig oft verwendet wird. Die Leistung des Datengrids wird durch die Leistung des Computers beschränkt, der diesen Datensatz enthält.

Versuchen Sie in solchen Situationen, den Datensatz aufzuteilen, sodass er pro Partition verwaltet wird. Stellen Sie sich beispielsweise eine Transaktion vor, die die Gesamtanzahl der Einträge im verteilten Cache zurückgibt. Anstatt jede Einfüge- und Entfernungsoperation auf einen einzelnen Datensatz zugreifen zu lassen, dessen Zähler sich erhöht, können Sie einen Listener auf jeder Partition einsetzen, der die Einfüge- und Entfernungsoperation verfolgt. Mit dieser Listenerverfolgung können aus Einfüge- und Entfernungsoperationen Einzelpartitionsoperationen werden.

Das Lesen des Zählers wird zu einer datengridübergreifenden Operation, aber die Leseoperation war bereits vorher genauso ineffizient wie eine datengridübergreifende Operation, weil ihre Leistung an die Leistung des Computers gebunden war, auf dem sich der Datensatz befindet.

Implementierungstipps

Zum Erreichen der besten Skalierbarkeit können Sie außerdem die folgenden Tipps beachten.

Umgekehrte Suchindizes verwenden

Stellen Sie sich ein ordnungsgemäß denormalisiertes Datenmodell vor, in dem Kundendatensätze auf der Basis der Kunden-ID partitioniert werden. Diese Partitionierungsmethode ist die logische Option, weil nahezu jede Geschäftsoperation, die mit dem Kundendatensatz ausgeführt wird, die Kunden-ID verwendet. Eine wichtige Transaktion, in der die Kunden-ID jedoch nicht verwendet wird, ist die Anmeldetransaktion. Es ist üblich, dass Benutzernamen oder E-Mail-Adressen für die Anmeldung verwendet werden, und keine Kunden-IDs.

Der einfache Ansatz für das Anmeldeszenario ist die Verwendung einer datengridübergreifenden Transaktion, um den Kundendatensatz zu suchen. Wie zuvor erläutert, ist dieser Ansatz nicht skalierbar.

Die nächste Option ist die Partitionierung nach Benutzernamen oder E-Mail-Adressen. Diese Option ist nicht praktikabel, da alle Operationen, die auf der Kunden-ID basieren, zu datengridübergreifenden Transaktionen werden. Außerdem möchten die Kunden auf Ihrer Site möglicherweise ihren Benutzernamen oder ihre E-Mail-Adresse ändern. Produkte wie WebSphere eXtreme Scale benötigen den Wert, der für die Partitionierung der Daten verwendet wird, um konstant zu bleiben.

Die richtige Lösung ist die Verwendung eines umgekehrten Suchindex. Mit WebSphere eXtreme Scale kann ein Cache in demselben verteilten Grid wie der Cache erstellt werden, der alle Benutzerdatensätze enthält. Dieser Cache ist hoch verfügbar, partitioniert und skalierbar. Dieser Cache kann verwendet werden, um einen Benutzernamen oder eine E-Mail-Adresse einer Kunden-ID zuzuordnen. Dieser Cache verwandelt die Anmeldung in eine Operation, an der zwei Partitionen beteiligt sind, und nicht in eine gridübergreifende Transaktion. Dieses Szenario ist zwar nicht so effektiv wie eine Einzelpartitionstransaktion, aber der Durchsatz nimmt linear mit steigender Anzahl an Computern zu.

Berechnung beim Schreiben

Die Generierung häufig berechneter Werte wie Durchschnittswerte oder Summen kann kostenintensiv sein, weil bei diesen Operationen gewöhnlich sehr viele Einträge gelesen werden müssen. Da in den meisten Anwendungen mehr Leseoperationen als Schreiboperationen ausgeführt werden, ist es effizient, diese Werte beim Schreiben zu berechnen und das Ergebnis anschließend im Cache zu speichern. Durch dieses Verfahren werden Leseoperationen schneller und skalierbarer.

Optionale Felder

Stellen Sie sich einen Benutzerdatensatz vor, der eine geschäftliche Telefonnummer, eine private Telefonnummer und eine Handy-Nummer enthält. Ein Benutzer kann alle, keine oder eine beliebige Kombination dieser Nummern haben. Wenn die Daten normalisiert sind, sind eine Benutzertabelle und eine Telefonnummerntabelle vorhanden. Die Telefonnummern für einen bestimmten Benutzer können über eine JOIN-Operation zwischen den beiden Tabellen ermittelt werden.

Die Denormalisierung dieses Datensatzes erfordert keine Datenduplizierung, weil die meisten Benutzer nicht dieselben Telefonnummern haben. Stattdessen müssen freie Bereiche im Benutzerdatensatz zulässig sein. Anstatt eine Telefonnummerntabelle zu verwenden, können Sie jedem Benutzerdatensatz drei Attribute hinzufügen, eines für jeden Telefonnummerntyp. Durch das Hinzufügen dieser Attribut wird die JOIN-Operation vermieden, und die Suche der Telefonnummern für einen Benutzer wird zu einer Einzelpartitionsoperation.

Verteilung von Viele-zu-viele-Beziehungen

Stellen Sie sich eine Anwendung, die Produkte und die Länden verfolgt, in denen die Produkte verkauft werden. Ein Produkt wird in vielen Läden verkauft, und ein Laden verkauft viele Produkte. Angenommen, diese Anwendung verfolgt 50 große Einzelhändler. Jedes Produkt wird in maximal 50 Länden verkauft, wobei jeder Laden Tausende von Produkten verkauft.

Verwalten Sie eine Liste der Läden in der Produktentität (Anordnung A), anstatt eine Liste von Produkten in jeder Ladenentität zu verwalten (Anordnung B). Wenn Sie sich einige der Transaktionen ansehen, die diese Anwendung ausführen muss, ist leicht zu erkennen, warum Anordnung A skalierbarer ist.

Sehen Sie sich zuerst die Aktualisierungen an. Wenn bei Anordnung A ein Produkt aus dem Bestand eines Ladens entfernt wird, wird die Produktentität gesperrt. Enthält das Datengrid 10000 Produkte, muss nur 1/10000 des Grids gesperrt werden, um die Aktualisierung durchzuführen. Bei Anordnung B enthält das Datengrid nur 50 Länden, sodass 1/50 des Grids gesperrt werden muss, um die Aktualisierung durchzuführen. Obwohl beide Fälle als Einzelpartitionsoperationen eingestuft werden können, lässt sich Anordnung A effizienter skalieren.

Sehen Sie sich jetzt die Leseoperationen für Anordnung A an. Das Durchsuchen eines Ladens, in dem ein Produkt verkauft wird, ist eine Einzelpartitionsoperation, die skalierbar und schnell ist, weil die Transaktion nur einen kleine Datenmenge überträgt. Bei Anordnung B wird aus dieser Transaktion eine datengridübergreifende Transaktion, weil auf jede Ladenentität zugegriffen werden muss, um festzustellen, ob das Produkt in diesem Laden verkauft wird. Daraus ergibt sich ein enormer Leistungsvorteil für Anordnung A.

Skalierung mit normalisierten Daten

Eine zulässige Verwendung von datengridübergreifenden Transaktionen ist die Skalierung der Datenverarbeitung. Wenn ein Datengrid 5 Computer enthält und eine datengridübergreifende Transaktion zugeteilt wird, die 100.000 Datensätze auf jedem Computer durchsucht, durchsucht diese Transaktion insgesamt 500.000 Datensätze. Wenn der langsamste Computer im Datengrid 10 dieser Transaktionen pro Sekunde ausführen kann, ist das Datengrid in der Lage, 5.000.000 Datensätze pro Sekunde zu durchsuchen. Wenn sich die Daten im Grid verdoppeln, muss jeder Computer 200.000 Datensätze durchsuchen, und jede Transaktion durchsucht insgesamt 1.000.000 Datensätze. Diese Datenzunahme verringert den Durchsatz des langsamsten Computers auf 5 Transaktionen pro Sekunde und damit den Durchsatz des Datengrids auf 5 Transaktionen pro Sekunde. Das Datengrid durchsucht weiterhin 5.000.000 Datensätze pro Sekunde.

In diesem Szenario kann jeder Computer durch die Verdopplung der Computeranzahl zu seiner vorherigen Last von 100.000 Datensätzen zurückkehren, und der langsamste Computer kann wieder 10 dieser Transaktionen pro Sekunde verarbeiten. Der Durchsatz des Datengrids bleibt bei 10 Anforderungen pro Sekunde, aber jetzt verarbeitet jede Transaktion 1.000.000 Datensätze, sodass das Grid seine Kapazität in Bezug auf die Verarbeitung von Datensätzen auf 10.000.000 pro Sekunde verdoppelt hat.

Für Anwendungen wie Suchmaschinen, die sowohl in Bezug auf die Datenverarbeitung (angesichts der zunehmenden Größe des Internets) als auch in Bezug auf den Durchsatz (angesichts der zunehmenden Anzahl an Benutzern) skalierbar sein müssen, müssen Sie mehrere Grids mit einem

Umlaufverfahren für die Anforderungen zwischen den Datengrids erstellen. Wenn Sie den Durchsatz erhöhen müssen, fügen Sie Computer und ein weiteres Datengrid für die Bearbeitung der Anforderungen hinzu. Wenn die Datenverarbeitung erhöht werden muss, fügen Sie weitere Computer hinzu, und halten Sie die Anzahl der Datengrids konstant.

Skalierung in Einheiten oder Pods

Obwohl Sie ein Datengrid in Tausenden von Java Virtual Machines implementieren können, sollten Sie darüber nachdenken, das Datengrid möglicherweise in Einheiten oder Pods einzuteilen, um die Zuverlässigkeit zu erhöhen und das Testen der Konfiguration zu vereinfachen. Ein Pod ist eine Gruppe von Servern, in denen dieselben Anwendungen ausgeführt werden.

Implementierungen eines einzigen großen Datengrids

Tests haben ergeben, dass eXtreme Scale auf über 1000 JVMs skaliert werden kann. Diese Tests animieren zum Erstellen von Anwendungen für die Implementierung eines einzigen Datengrids auf sehr vielen Maschinen. Obwohl diese Vorgehensweise möglich ist, wird sie aus verschiedenen Gründen, die im Folgenden beschrieben werden, nicht empfohlen.

1. **Budgetbedenken:** Realistisch betrachtet ist Ihre Umgebung nicht in der Lage, ein Datengrid mit 1000 Servern zu testen. Tests mit einem sehr viel kleineren Datengrid sind unter Berücksichtigung von Budgetgründen jedoch möglich, sodass Hardware nicht doppelt gekauft werden muss, insbesondere bei einer solch hohen Anzahl an Servern.
2. **Verschiedene Anwendungsversionen:** Für jeden einzelnen Test sehr viele Maschinen zu benötigen, ist nicht sehr praktisch. Es besteht das Risiko, dass nicht dieselben Faktoren getestet werden, wie es in einer Produktionsumgebung der Fall ist.
3. **Datenverlust:** Die Ausführung einer Datenbank auf einem einzigen Festplattenlaufwerk ist unzuverlässig. Jedes Problem mit dem Festplattenlaufwerk führt zum Verlust von Daten. Die Ausführung einer ausbaufähigen Anwendung in einem einzigen Datengrid ist ähnlich. Es ist wahrscheinlich, dass Sie Programmfehler in Ihrer Umgebung und in Ihren Anwendungen haben. Die Speicherung aller Daten auf einem einzigen großen System führt deshalb häufig zum Verlust großer Datenmengen.

Aufteilung des Datengrids

Die Aufteilung des Anwendungsdatengrids in Pods (Einheiten) ist zuverlässiger. Ein Pod ist eine Gruppe von Servern, auf denen ein homogener Anwendungs-Stack ausgeführt wird. Pods können beliebig groß sein, sollten sich idealerweise aber aus ca. 20 physischen Servern zusammensetzen. Anstatt 500 physische Server in einem einzigen Datengrid zu verwenden, können Sie 25 Pods mit jeweils 20 physischen Servern verwenden. Es sollte jeweils nur eine einzige Version eines Anwendungs-Stacks in einem bestimmten Pod ausgeführt werden, aber die verschiedenen Pods können jeweils eigene Versionen eines Anwendungs-Stacks haben.

Im Allgemeinen werden in einem Anwendungs-Stack die Versionsstände der folgenden Komponenten berücksichtigt:

- Betriebssystem
- Hardware
- JVM
- Version von WebSphere eXtreme Scale

- Anwendung
- Weitere erforderliche Komponenten

Ein Pod ist eine Verteilungseinheit mit einer für Tests geeigneten Größe. Anstatt Hunderte von Servern für Tests zu verwenden, empfiehlt es sich, nur 20 Server zu verwenden. In diesem Fall testen Sie dieselbe Konfiguration wie in einer Produktionsumgebung. In Produktionsumgebungen werden Grids mit einer Größe von maximal 20 Servern, die einen Pod bilden, verwendet. Sie können Belastungstests für einen einzigen Pod ausführen und dabei die Kapazität, die Anzahl der Benutzer, das Datenvolumen und den Transaktionsdurchsatz dieses Pods bestimmen. Dies vereinfacht die Planung und entspricht dem Standard einer vorhersehbaren Skalierung zu vorhersehbaren Kosten.

Einrichtung einer Pod-basierten Umgebung

In manchen Fällen muss der Pod nicht unbedingt 20 Server haben. Die Pod-Größe dient ausschließlich dem Zweck praxistauglicher Tests. Ein Pod sollte klein genug sein, dass der Teil der beim Auftreten von Pod-Problemen in einer Produktionsumgebung betroffenen Transaktionen tolerierbar bleibt.

Im Idealfall wirkt sich ein Programmfehler auf einen einzigen Pod aus. Ein Programmfehler würde sich nur auf vier Prozent der Anwendungstransaktionen und nicht auf 100 Prozent auswirken. Außerdem sind Upgrades einfacher, weil sie nacheinander in jeweils einem Pod implementiert werden können. Wenn aufgrund eines Upgrades in einem Pod Probleme auftreten, kann der Benutzer somit den Pod auf die vorherige Version zurücksetzen. Upgrades umfassen alle Änderungen, die an der Anwendung bzw. am Anwendungs-Stack vorgenommen werden, sowie Systemaktualisierungen. Bei Upgrades sollte möglichst jeweils nur ein einziges Element des Stacks aktualisiert werden, um die Problemdiagnose zu vereinfachen.

Zum Implementieren einer Umgebung mit Pods benötigen Sie eine Routing-Schicht oberhalb der Pods, die auf- und abwärtskompatibel ist, wenn Software-Upgrades auf Pods angewendet werden. Außerdem sollten Sie ein Verzeichnis erstellen, das Informationen dazu enthält, welcher Pod welche Daten enthält. (Hierfür können Sie ein weiteres datenbankgestütztes eXtreme-Scale-Datengrid verwenden, vorzugsweise mit Verwendung des Write-behind-Szenarios.) Dies ergibt eine 2-Schichten-Lösung. Schicht 1 ist das Verzeichnis und wird verwendet, um den Pod zu ermitteln, der eine bestimmte Transaktion bearbeitet. Schicht 2 setzt sich aus den Pods selbst zusammen. Wenn Schicht 1 einen Pod identifiziert, leitet das Setup jede Transaktion an den richtigen Server im Pod weiter, der gewöhnlich der Server ist, der die Partition für die von der Transaktion verwendeten Daten enthält. Optional können Sie einen nahen Cache auf Schicht 1 verwenden, um die Auswirkungen zu mindern, die mit der Ermittlung des richtigen Pods verbunden sind.

Die Verwendung von Pods ist geringfügig komplexer als die Verwendung eines einzigen Datengrids, aber aufgrund der Verbesserungen in Bezug auf den Betrieb, die Tests und die Zuverlässigkeit sind Pods ein entscheidender Faktor bei Skalierbarkeitstests.

Übersicht über Verfügbarkeit

Hohe Verfügbarkeit

Mit hoher Verfügbarkeit unterstützt WebSphere eXtreme Scale zuverlässige Datenredundanz und Fehlererkennung.

WebSphere eXtreme Scale organisiert JVM-Datengrids eigenständig in einem losen baumähnlichen Verbund. Der Katalogservice im Stammverzeichnis und die Stammgruppen, die die Container enthalten, sind die Blätter des Baums. Weitere Informationen finden Sie im Abschnitt „Caching-Architektur: Maps, Container, Clients und Kataloge" auf Seite 17.

Jede Stammgruppe wird vom Katalogservice automatisch in Gruppen von ungefähr 20 Servern unterteilt. Die Stammgruppen-Member überwachen die anderen Member der Gruppe auf ihre Vitalität. Außerdem wählt jede Stammgruppe ein Member als führendes Member aus, das für die Kommunikation der Gruppeninformation an den Katalogservice zuständig ist. Eine Begrenzung der Stammgruppengröße sorgt für eine effektive Vitalitätsüberwachung und eine hoch skalierbare Umgebung.

Anmerkung: In einer Umgebung mit WebSphere Application Server, in der die Stammgruppengröße geändert werden kann, unterstützt eXtreme Scale maximal 50 Member pro Stammgruppe.

Austausch von Überwachungssignalen

1. Sockets werden zwischen Java Virtual Machines bleiben offen, und wenn ein Socket unerwartet geschlossen wird, wird dieses unerwartete Schließen als Fehler von der Peer-JVM erkannt. Bei dieser Erkennung werden Fehlerfälle wie beispielsweise das sehr schnelle Beenden der Java Virtual Machine abgefangen. Außerdem unterstützt dieser Erkennungstyp die Wiederherstellung nach solchen Fehlern in gewöhnlich weniger als einer Sekunde.
2. Weitere Typen von Fehlern sind Betriebssystempanik, physischer Ausfall eines Servers und Netzfehler. Diese Fehler werden über den Austausch von Überwachungssignalen erkannt.

Überwachungssignale werden in regelmäßigen Abständen von zwei Prozessen ausgetauscht. Wenn eine festgelegte Anzahl an Überwachungssignalen verpasst wird, wird von einem Fehler ausgegangen. Mit diesem Ansatz werden Fehler in N*M Sekunden erkannt. N steht für die Anzahl entgangener Überwachungssignale und M für das Intervall der Überwachungssignale (auch Heartbeatintervall). Die direkte Angabe von M und N wird nicht unterstützt. Es wird ein Reglermechanismus verwendet, damit ein Bereich getesteter M/N-Kombinationen verwendet werden kann.

Fehler

Ein Prozess kann auf verschiedene Arten fehlschlagen. Ein Prozess kann fehlschlagen, weil eine Ressourcengrenze erreicht wurde, wie z. B. die maximale Größe des Heapspeichers, oder weil eine Prozesssteuerungslogik den Prozess beendet hat. Das Betriebssystem kann ausfallen, was dazu führt, dass alle auf dem System ausgeführten Prozesse verloren gehen. Die Hardware, wie z. B. die Netzschnittstellenkarte, kann (wenn auch weniger häufig) ausfallen, was zur Trennung des Betriebssystems vom Netz führen kann. Es gibt viele weitere Fehlerquellen, die die Nichtverfügbarkeit des Prozesses zur Folge haben können. In diesem Kontext können all diese Fehler in zwei Kategorien eingeteilt werden: Prozessfehler und Konnektivitätsverlust.

Prozessfehler

WebSphere eXtreme Scale reagiert schnell auf Prozessfehler. Wenn ein Prozess fehlschlägt, ist das Betriebssystem für die Bereinigung aller übrig gebliebenen Ressour-

cen verantwortlich, die vom Prozess verwendet wurden. Diese Bereinigung umfasst die Portzuordnung und die Konnektivität. Wenn ein Prozess fehlschlägt, wird ein Signal über die von diesem Prozess verwendeten Verbindungen gesendet, um jede einzelne Verbindung zu schließen. Mit Hilfe dieser Signale kann ein Prozessfehler in kürzester Zeit von einem anderen Prozess, der mit dem fehlgeschlagenen verbunden ist, erkannt werden.

Konnektivitätsverlust

Ein Konnektivitätsverlust tritt auf, wenn die Verbindung des Betriebssystems zum Netz getrennt wird. Infolgedessen kann das Betriebssystem keine Signale an andere Prozesse senden. Es gibt mehrere Gründe für einen Konnektivitätsverlust, die jedoch in zwei Kategorien eingeteilt werden können: Hostausfall und Isolierung.

Hostausfall

Wenn der Netzstecker der Maschine gezogen wird, geht die Konnektivität sofort verloren.

Isolierung

Dieses Szenario stellt die komplizierteste Fehlerbedingung für Software dar. Die Behebung einer solchen Fehlerbedingung stellt sich so schwierig dar, weil davon ausgegangen wird, dass der Prozess nicht verfügbar ist, obwohl dies gar nicht der Fall ist. Für das System scheint ein Server oder ein anderer Prozess ausgefallen zu sein, obwohl er in Wirklichkeit ordnungsgemäß ausgeführt wird.

Containerfehler

Containerausfälle werden im Allgemeinen von Peer-Containern über den Stammgruppenmechanismus erkannt. Wenn ein Container oder eine Gruppe von Containern ausfällt, migriert der Katalogservice die Shards aus den betroffenen Containern. Der Katalogservice sucht zuerst nach einem synchronen Replikat, bevor er die Migration auf ein asynchrones Replikat durchführt. Nach der Migration der primären Shards in die neuen Hostcontainer durchsucht der Katalogservice die neuen Hostcontainer nach den Replikaten, die jetzt fehlen.

Anmerkung: Containerisolierung - Der Katalogservice migriert Shards aus Containern, wenn der Container als nicht verfügbar erkannt wird. Wenn diese Container wieder verfügbar werden, berücksichtigt der Katalogservice sie bei der Verteilung wie beim normalen Startablauf.

Latenzzeit für Erkennung von Containerfehlern

Ausfälle können in die folgenden beiden Kategorien eingeteilt werden: temporäre Ausfälle und permanente Ausfälle. Temporäre Ausfälle werden gewöhnlich durch einen Prozessfehler verursacht. Solche Ausfälle werden vom Betriebssystem erkannt, das genutzte Ressourcen, wie z. B. Netz-Sockets, schnell wiederherstellen kann. Gewöhnlich werden temporäre Ausfälle in weniger als einer Sekunde erkannt. Die Erkennung permanenter Ausfälle kann unter Verwendung der Standardoptimierung für Überwachungssignale bis zu 200 Sekunden dauern. Zu solchen Ausfällen gehören physische Ausfälle von Maschinen, das Ziehen von Netzkabeln und Betriebssystemausfälle. Die Laufzeitumgebung verlässt sich darauf, dass permanente Fehler durch den Austausch von Überwachungssignalen erkannt werden, der konfiguriert werden kann.

Ausfall des Katalogservice

Da das Katalog-Service-Grid ein eXtreme-Scale-Grid ist, wird der Stammgruppenmechanismus auch hier auf dieselbe Weise wie beim Containerausfallprozess verwendet. Der Hauptunterschied besteht darin, dass die Katalogservicedomäne einen Peer-Auswahlprozess für die Definition des primären Shards an Stelle des Katalogservicealgorithmus verwendet, der für die Container verwendet wird.

Der Verteilungsservice und der Stammgruppierungsservice sind Services vom Typ "1 von N" sind, der Positionsservice und die Verwaltung hingegen überall ausgeführt werden. Ein Service vom Typ "1 von N" wird nur in einem einzigen Member der HA-Gruppe ausgeführt. Der Positionsservice und der Verwaltungsservice werden in allen Membern der HA-Gruppe ausgeführt. Der Verteilungsservice und der Stammgruppierungsservice sind Singletons, weil sie für das Layout des Systems verantwortlich sind. Der Positionsservice und die Verwaltung sind Services, die ausschließlich im Lesezugriff arbeiten und zur Unterstützung der Skalierbarkeit überall vorhanden sind.

Der Katalogservice verwendet die Replikation für seine eigene Fehlertoleranz. Wenn ein Katalogserviceprozess fehlschlägt, wird der Service erneut gestartet um das System in einem Zustand wiederherzustellen, der die gewünschte Stufe der Verfügbarkeit bietet. Schlagen alle Prozesse, in denen der Katalogservice ausgeführt wird, fehl, verliert das Datengrid kritische Daten. Dieser Fehler führt zu einem erforderlichen Neustart aller Container-Server. Da der Katalogservice in vielen Prozessen ausgeführt werden kann, ist das Auftreten dieses Fehlers eher unwahrscheinlich. Wenn Sie jedoch alle Prozesse auf einer einzigen Maschine, in einem einzigen Blade-Gehäuse oder über einen einzigen Netz-Switch ausführen, ist die Wahrscheinlich eines Fehlers hoch. Versuchen Sie, bekannte Fehlermodi auf Maschinen, auf denen der Katalogservice ausgeführt wird, zu beseitigen, um die Fehlerwahrscheinlichkeit zu reduzieren.

Mehrere Containerausfälle

Ein Replikat wird niemals in demselben Prozess wie das primäre Shard ausgeführt, da dies beim Verlust des Prozesses zu einem Verlust des primären Shards und des Replikats führen würde. In einer Entwicklungsumgebung auf einer einzigen Maschine können Sie zwei Container verwenden und die Daten des einen Containers im jeweils anderen replizieren. Sie können das Attribut für den Entwicklungsmodus in der Implementierungsrichtlinie definieren, um zu konfigurieren, dass ein Replikat an dieselbe Maschine wie das primäre Shard verteilt wird. In einer Produktionsumgebung reicht die Verwendung einer einzige Maschine jedoch nicht aus, weil ein Verlust dieses Hosts zum Verlust beider Container-Server führt. Zum Wechsel vom Entwicklungsmodus auf einer einzigen Maschine in den Produktionsmodus mit mehreren Maschinen müssen Sie den Entwicklungsmodus in der Konfigurationsdatei der Implementierungsrichtlinie inaktivieren.

Tabelle 2. Zusammenfassung der Fehlererkennung und Wiederherstellung

Verlusttyp	Erkennungsmechanismus	Wiederherstellungsmethode
Prozessverlust	Ein-/Ausgabe	Neustart
Serververlust	Überwachungssignal	Neustart
Netzausfall	Überwachungssignal	Netz und Verbindung wiederherstellen
Serverseitige Blockierung	Überwachungssignal	Server stoppen und erneut starten

Tabelle 2. Zusammenfassung der Fehlererkennung und Wiederherstellung (Forts.)

Verlusttyp	Erkennungsmechanismus	Wiederherstellungsmethode
Server ausgelastet	Überwachungssignal	Warten, bis Server wieder verfügbar ist

Replikation für Verfügbarkeit

Die Replikation unterstützt Fehlertoleranz und erhöht die Leistung für eine verteilte eXtreme-Scale-Topologie. Die Replikation wird durch die Zuordnung von BackingMaps zu einem MapSet aktiviert.

Informationen zu MapSets

Ein MapSet ist eine Sammlung von Maps, die anhand eines Partitionsschlüssels kategorisiert werden. Dieser Partitionsschlüssel wird mit der folgenden Formel aus dem Schlüssel der jeweiligen Map abgeleitet: Hash Modulo Partitionsanzahl. Wenn eine Gruppe von Maps im MapSet den Partitionsschlüssel X hat, werden diese Maps in einer entsprechenden Partition X im Datengrid gespeichert. Wenn eine andere Gruppe den Partitionsschlüssel Y hat, werden alle Maps in Partition Y gespeichert usw. Die Daten in den Maps werden basierend auf der im MapSet definierten Richtlinie repliziert. Die Replikation findet in verteilten Topologien statt.

Den MapSets werden die Anzahl der Partitionen und eine Replikationsrichtlinie zugeordnet. Die Replikationskonfiguration für das MapSet gibt die Anzahl synchroner und asynchroner Replikat-Shards an, die das MapSet zusätzlich zum primären Shard haben muss. Sind beispielsweise ein synchrones und ein asynchrones Replikat vorhanden, wird für alle BackingMaps, die dem MapSet zugeordnet sind, jeweils automatisch ein Replikat-Shard in der Gruppe verfügbarer Container-Server für das Datengrid verteilt. Die Replikationskonfiguration kann Clients auch das Lesen von Daten aus synchron replizierten Servern ermöglichen. Auf diese Weise kann die Last der Leseanforderungen auf zusätzliche Server in eXtreme Scale verteilt werden. Die Replikation hat nur dann Auswirkung auf das Programmiermodell, wenn die BackingMaps vorher geladen werden.

Vorheriges Laden von Maps

Maps können so genannte Loader (Ladeprogramme) zugeordnet werden. Ein Loader wird verwendet, um Objekte abzurufen, wenn diese nicht in der Map gefunden werden (Cachefehler), und um Änderungen in ein Back-End zu schreiben, wenn eine Transaktion festgeschrieben wird. Loader können auch für das vorherige Laden von Daten (Preload) in eine Map verwendet werden. Die Methode preloadMap der Schnittstelle Loader wird für jede Map aufgerufen, wenn die zugehörige Partition im MapSet zu einem primären Shard wird. Die Methode preloadMap wird nicht für Replikate aufgerufen. Sie versucht, alle geplanten referenzierten Daten über die bereitgestellte Sitzung aus dem Back-End in die Map zu laden. Die jeweilige Map wird mit dem Argument "BackingMap" angegeben, das an die Methode preloadMap übergeben wird.

```
void preloadMap(Session session, BackingMap backingMap) throws LoaderException;
```

Vorheriges Laden in einem partitionierten MapSet

Maps können in N Partitionen partitioniert werden. Deshalb können Maps auf mehrere Server verteilt werden, wobei jeder Eintrag mit einem Schlüssel gekennzeichnet wird, der nur in einem einzigen dieser Server gespeichert wird. Sehr große Maps können in eXtreme Scale verwaltet werden, weil die Anwendung nicht

mehr durch die Heapspeichergröße einer einzigen JVM beschränkt ist, die alle Einträge einer Map enthält. Anwendungen, die den Preload-Prozess mit der Methode "preloadMap" der Schnittstelle "Loader" ausführen möchten, müssen den Teil der Daten angeben, die vorher geladen werden sollen. Es ist immer eine feste Anzahl an Partitionen vorhanden. Sie können diese Zahl anhand des folgenden Codebeispiels bestimmen:

```
int numPartitions = backingMap.getPartitionManager().getNumOfPartitions();
int myPartition = backingMap.getPartitionId();
```

Dieses Codebeispiel zeigt, dass eine Anwendung den Teil der Daten angeben kann, der vorher aus der Datenbank geladen werden soll. Anwendungen müssen diese Methoden auch dann verwenden, wenn die Map zunächst nicht partitioniert ist. Diese Methoden bieten Flexibilität: Wenn die Map später von den Administratoren partitioniert wird, funktioniert der Loader weiterhin ordnungsgemäß.

Die Anwendung muss Abfragen absetzen, um den Teil *myPartition* aus dem Back-End abzurufen. Wenn eine Datenbank verwendet wird, kann es unter Umständen einfacher sein, eine Spalte mit der Partitionskennung für einen bestimmten Datensatz zu haben, sofern es keine natürliche Abfrage gibt, mit der die Daten in der Tabelle einfach partitioniert werden können.

Leistung

Die Preload-Implementierung kopiert Daten aus dem Back-End in die Map, indem sie mehrere Objekte in der Map in einer einzigen Transaktion speichert. Die optimale Anzahl der pro Transaktion zu speichernden Datensätze richtet sich nach mehreren Faktoren, einschließlich der Komplexität und der Größe. Wenn die Transaktion beispielsweise Blöcke mit mehr als 100 Einträgen enthält, nehmen die Leistungsgewinne ab, wenn Sie die Anzahl der Einträge erhöhen. Zur Bestimmung der optimalen Anzahl beginnen Sie mit 100 Einträgen, und erhöhen Sie dann die Anzahl, bis keine Leistungsgewinne mehr zu verzeichnen sind. Mit größeren Transaktionen kann eine bessere Replikationsleistung erzielt werden. Denken Sie daran, dass der Preload-Code nur im primären Shard ausgeführt wird. Die vorher geladenen Daten werden über das primäre Shard in allen Replikaten repliziert, die online sind.

MapSets vorher laden

Wenn die Anwendung ein MapSet mit mehreren Maps verwendet, hat jede Map einen eigenen Loader. Jeder Loader besitzt eine Preload-Methode. Alle Maps werden nacheinander von eXtreme Scale geladen. Es kann effizienter sein, den Preload-Prozess für die Maps so zu gestalten, dass eine einzige Map als Map bestimmt wird, in die die Daten vorher geladen werden. Dieser Prozess ist eine Anwendungskonvention. Beispiel: Die beiden Maps "department" (Abteilung) und "employee" (Mitarbeiter) könnten beide den Loader der Map "department" verwenden. Bei dieser Prozedur wird über Transaktionen gewährleistet, dass in dem Fall, dass eine Anwendung eine Abteilung abrufen möchte, sich die Mitarbeiter für diese Abteilung im Cache befinden. Wenn der Loader der Map "department" eine Abteilung vorher aus dem Back-End lädt, ruft er auch die Mitarbeiter für diese Abteilung ab. Das department-Objekt und die zugehörigen employee-Objekte werden dann der Map in einer einzigen Transaktion hinzugefügt.

Wiederherstellbares vorheriges Laden

Einige Kunden haben sehr große Datenmengen, die zwischengespeichert werden müssen. Das vorherige Laden dieser Daten kann sehr zeitaufwendig sein. Manch-

mal muss das vorherige Laden abgeschlossen sein, bevor die Anwendung online gehen kann. In diesem Fall können Sie von einem wiederherstellbaren vorherigen Laden profitieren. Angenommen, es gibt Millionen Datensätze, die vorher geladen werden müssen. Die Daten werden vorab in das primäre Shard geladen, und der Prozess scheitert bei Datensatz 800.000. Normalerweise löscht das als neue primäre Shard ausgewählte Replikat den Replikationsstatus und beginnt von vorne. eXtreme Scale kann eine Schnittstelle "ReplicaPreloadController" verwenden. Der Loader für die Anwendung muss auch die Schnittstelle "ReplicaPreloadController" implementieren. Das folgende Beispiel fügt dem Loader eine einzige Methode hinzu: `Status checkPreloadStatus(Session session, BackingMap bmap);`. Diese Methode wird von der Laufzeitumgebung von eXtreme Scale aufgerufen, bevor die Methode "preload" der Schnittstelle "Loader" aufgerufen wird. eXtreme Scale prüft das Ergebnis dieser Methode (Status), um das Verhalten festzulegen, wenn ein Replikat in ein primäres Shard hochgestuft werden muss.

Tabelle 3. Statuswert und Antwort

Zurückgegebener Statuswert	Antwort von eXtreme Scale
Status.PRELOADED_ALREADY	eXtreme Scale ruft die Methode "preload" gar nicht auf, weil dieser Statuswert anzeigt, dass der Preload-Prozess für die Map bereits vollständig abgeschlossen ist.
Status.FULL_PRELOAD_NEEDED	eXtreme Scale löscht die Map und ruft ganz regulär die Methode "preload" auf.
Status.PARTIAL_PRELOAD_NEEDED	eXtreme Scale belässt die Map im aktuellen Zustand und ruft die Methode "preload" auf. Diese Strategie ermöglicht dem Loader der Anwendung, den Preload-Prozess an der Stelle fortzusetzen, an der er zuvor abgebrochen wurde.

Es ist logisch, dass ein primäres Shard beim vorherigen Laden von Daten in die Map einen Status in einer Map im MapSet hinterlassen muss, das repliziert wird, sodass das Replikat den zurückzugebenden Status bestimmen kann. Sie können dazu eine zusätzliche Map verwenden, die z. B. den Namen RecoveryMap hat. Diese RecoveryMap muss zu demselben MapSet gehören, das vorher geladen wird, um sicherzustellen, dass die replizierte Map mit den vorher geladenen Daten konsistent ist. Eine empfohlene Implementierung folgt.

Während der Preload-Prozess die einzelnen Datensatzblöcke festschreibt, aktualisiert er im Rahmen dieser Transaktion auch einen Zähler oder Wert in der RecoveryMap. Die vorher geladenen Daten und die RecoveryMap-Daten werden automatisch in den Replikaten repliziert. Wenn das Replikat in ein primäres Shard hochgestuft wird, kann es anhand der RecoveryMap prüfen, was passiert ist.

Die RecoveryMap kann einen einzigen Eintrag mit dem Statusschlüssel enthalten. Wenn kein Objekt für diesen Schlüssel vorhanden ist, müssen Sie eine vollständige Operation preload (checkPreloadStatus returns FULL_PRELOAD_NEEDED) durchführen. Wenn ein Objekt für diesen Statusschlüssel vorhanden ist und der Wert COMPLETE lautet, ist der Preload-Prozess abgeschlossen, und die Methode checkPreloadStatus gibt PRELOADED_ALREADY zurück. Andernfalls zeigt das Wertobjekt an, wo der Preload-Prozess erneut gestartet werden muss, und die Methode checkPreloadStatus gibt `PARTIAL_PRELOAD_NEEDED` zurück. Der Loader kann den Wiederherstellungspunkt in einer Instanzvariablen speichern, sodass der Loader beim Aufruf der Methode "preload" diesen Ausgangspunkt kennt. Die RecoveryMap kann auch einen Eintrag pro Map enthalten, wenn jede Map gesondert vorher geladen wird.

Handhabung der Wiederherstellung im synchronen Replikationsmodus mit einem Loader

Die Laufzeitumgebung von eXtreme Scale ist so konzipiert, dass festgeschriebene Daten beim Ausfall des primären Shards nicht verloren gehen. Im folgenden Abschnitt werden der verwendeten Algorithmen beschrieben. Diese Algorithmen gelten nur, wenn eine Replikationsgruppe die synchrone Replikation verwendet. Ein Loader ist optional.

Die Laufzeitumgebung von eXtreme Scale kann so konfiguriert werden, dass alle Änderungen in einem primären Shard synchron in den Replikaten repliziert werden. Wenn ein synchrones Replikat verteilt wird, erhält es eine Kopie der vorhandenen Daten im primären Shard. In dieser Zeit empfängt das primäre Shard weiterhin Transaktionen und kopiert sie asynchron in das Replikat. Das Replikat wird in dieser Zeit nicht als online eingestuft.

Wenn das Replikat denselben Stand wie das primäre Shard hat, wechselt das Replikat in den Peer-Modus, und die synchrone Replikation beginnt. Jede im primären Shard festgeschriebene Transaktion wird an die synchronen Replikate gesendet, und das primäre Shard wartet auf eine Antwort jedes Replikats. Eine synchrone Festschreibungsfolge mit einem Loader im primären Shard setzt sich aus den folgenden Schritten zusammen:

Tabelle 4. Festschreibungsfolge im primären Shard

Schritt mit Loader	Schritt ohne Loader
Sperren für Einträge abrufen	Identisch
Änderung mit Flush an den Loader übertragen	Nulloperation
Änderungen im Cache speichern	Identisch
Änderungen an Replikate senden und auf Bestätigung warten	Identisch
Festschreibung im Loader über das TransactionCallback-Plug-in	Methode "commit" des Plug-ins wird zwar aufgerufen, führt aber keine Aktion aus
Sperren für Einträge freigeben	Identisch

Beachten Sie, dass die Änderungen an das Replikat gesendet werden, bevor sie im Loader festgeschrieben werden. Um zu bestimmen, wann die Änderungen im Replikat festgeschrieben werden, ändern Sie diese Folge. Initialisieren Sie während der Initialisierung die Transaktionslisten im primären Shard wie folgt:

```
CommitedTx = {}, RolledBackTx = {}
```

Für eine synchrone Commit-Verarbeitung verwenden Sie die folgende Folge:

Tabelle 5. Synchrone Commit-Verarbeitung

Schritt mit Loader	Schritt ohne Loader
Sperren für Einträge abrufen	Identisch
Änderung mit Flush an den Loader übertragen	Nulloperation
Änderungen im Cache speichern	Identisch
Änderungen mit einer festgeschriebenen Transaktion senden, Rollback der Transaktion an das Replikat durchführen und auf Bestätigung warten	Identisch

Tabelle 5. Synchrone Commit-Verarbeitung (Forts.)

Schritt mit Loader	Schritt ohne Loader
Liste festgeschriebener und rückgängig gemachter Transaktionen löschen	Identisch
Festschreibung im Loader über das TransactionCallBack-Plug-in	Methode des TransactionCallBack-Plug-ins wird weiterhin aufgerufen, führt aber gewöhnlich keine Aktion aus
Bei erfolgreicher Festschreibung Transaktion der Liste festgeschriebener Transaktionen hinzufügen, andernfalls der Liste rückgängig gemachter Transaktionen hinzufügen	Nulloperation
Sperren für Einträge freigeben	Identisch

Für die Replikatverarbeitung verwenden Sie die folgende Folge:

1. Änderungen empfangen
2. Alle empfangenen Transaktionen in der Liste festgeschriebener Transaktionen festschreiben
3. Alle empfangenen Transaktionen in der Liste rückgängig gemachter Transaktionen rückgängig machen
4. Transaktion oder Sitzung starten
5. Änderung auf die Transaktion oder Sitzung anwenden
6. Transaktion oder Sitzung in der Liste offener Transaktionen speichern
7. Antwort zurücksenden

Beachten Sie, dass im Replikat keine Loader-Interaktionen stattfinden, während sich das Replikat im Replikatmodus befindet. Das primäre Shard muss alle Änderungen mit Push über den Loader übertragen. Das Replikat überträgt keine Änderungen mit Push. Dieser Algorithmus hat den Nebeneffekt, dass das Replikat immer die Transaktionen hat, diese aber erst festgeschrieben werden, wenn die nächste primäre Transaktion den Festschreibungsstatus dieser Transaktionen sendet. Erst dann werden die Transaktionen im Replikat festgeschrieben oder rückgängig gemacht. Bis dahin sind die Transaktionen nicht festgeschrieben. Sie können einen Zeitgeber für das primäre Shard hinzufügen, der das Transaktionsergebnis nach kurzer Zeit (ein paar Sekunden) sendet. Dieser Zeitgeber verringert, schließt aber das Risiko veralteter Daten in diesem Zeitfenster nicht ganz aus. Veraltete Daten sind nur dann ein Problem, wenn der Lesemodus für Replikate verwendet wird. Andernfalls haben die veralteten Daten keine Auswirkungen auf die Anwendung.

Wenn das primäre Shard ausfällt, ist es wahrscheinlich, dass einige Transaktionen zwar im primären Shard festgeschrieben oder rückgängig gemacht wurden, aber die Nachricht mit diesen Ergebnissen nicht mehr an das Replikat gesendet werden konnte. Wenn ein Replikat als neues primäres Shard hochgestuft wird, ist eine der ersten Aktionen die Behandlung dieser Bedingung. Jede offene Transaktion wird erneut für die Mapgruppe des neuen primären Shards ausgeführt. Wenn ein Loader vorhanden ist, wird die erste Transaktion an den Loader übergeben. Diese Transaktionen werden in strikter First In/First Out-Reihenfolge angewendet. Wenn eine Transaktion scheitert, wird sie ignoriert. Sind drei Transaktionen, A, B und C, offen, kann A festgeschrieben, B rückgängig gemacht und C ebenfalls festgeschrieben werden. Keine der Transaktionen hat Auswirkung auf die anderen. Gehen Sie davon aus, dass sie voneinander unabhängig sind.

Ein Loader kann eine geringfügig andere Logik verwenden, wenn er sich im Modus für Fehlerbehebung durch Funktionsübernahme und nicht im normalen Modus befindet. Der Loader kann problemlos feststellen, wann er sich im Modus für Fehlerbehebung durch Funktionsübernahme befindet, indem er die Schnittstelle "ReplicaPreloadController" implementiert. Die Methode "checkPreloadStatus" wird erst aufgerufen, wenn der Loader wieder aus dem Modus für Fehlerbehebung durch Funktionsübernahme in den normalen Modus wechselt. Wenn die Methode "apply" der Schnittstelle "Loader" vor der Methode "checkPreloadStatus" aufgerufen wird, handelt es sich deshalb um eine Wiederherstellungstransaktion. Nach dem Aufruf der Methode "checkPreloadStatus" ist die Fehlerbehebung durch Funktionsübernahme abgeschlossen.

Lastausgleich mit Replikaten

eXtreme Scale sendet, sofern nicht anders konfiguriert, alle Lese- und Schreibanforderungen an den primären Server für eine bestimmte Replikationsgruppe. Der primäre Server muss alle Anforderungen von Clients bearbeiten. Sie können konfigurieren, dass Leseanforderungen auch an Replikate des primären Servers gesendet werden. Durch das Senden von Leseanforderungen an die Replikate kann die Last der Leseanforderungen auf mehrere Java Virtual Machines (JVM) verteilt werden. Die Verwendung von Replikaten für Leseanforderungen kann jedoch zu inkonsistenten Antworten führen.

Der Lastausgleich mit Replikaten wird gewöhnlich nur verwendet, wenn Clients Daten zwischenspeichern, die sich ständig ändern oder wenn die Clients pessimistisches Sperren verwenden.

Wenn sich die Daten ständig ändern und anschließend im nahen Cache des Clients ungültig gemacht werden, sollte der primäre Server daraufhin eine relativ hohe Rate von get-Anforderungen von Clients sehen. Im pessimistischen Sperrmodus ist kein lokaler Cache vorhanden, also werden alle Anforderungen an den primären Server gesendet.

Wenn die Daten relativ statisch sind oder der pessimistische Modus nicht verwendet wird, hat das Senden von Leseanforderungen an das Replikat keine erheblichen Auswirkungen auf die Leistung. Die Häufigkeit der get-Anforderungen von Clients mit Caches, die voll von Daten sind, ist nicht sehr hoch.

Wenn ein Client zum ersten Mal gestartet wird, ist sein naher Cache leer. Cacheanforderungen an den leeren Cache werden an den primären Server weitergeleitet. Nach und nach werden Daten in den Clientcache gestellt, sodass die Anforderungslast abnimmt. Wenn sehr viele Clients gleichzeitig gestartet werden, kann die Last erheblich und das Senden von Leseanforderungen an das Replikat eine angemessene Leistungsoption sein.

Clientseitige Replikation

Mit eXtreme Scale können Sie eine Server-Map in einem oder mehreren Clients durch asynchrone Replikation replizieren. Ein Client kann über die Methode ClientReplicableMap.enableClientReplication eine lokale schreibgeschützte Kopie einer serverseitigen Map anfordern.

```
void enableClientReplication(Mode mode, int[] partitions,
ReplicationMapListener listener) throws ObjectGridException;
```

Der erste Parameter ist der Replikationsmodus. Dieser Modus kann eine fortlaufende Replikation oder eine Momentaufnahmenreplikation sein. Der zweite Parameter

ist ein Bereich von Partitions-IDs, die die Partitionen darstellen, von denen Daten repliziert werden sollen. Wenn der Wert null oder ein leerer Bereich ist, werden die Daten von allen Partitionen repliziert. Der letzte Parameter ist ein Listener für den Empfang von Clientreplikationsereignissen. Weitere Einzelheiten hierzu finden Sie in den Beschreibungen der Schnittstellen "ClientReplicableMap" und "ReplicationMapListener" in der API-Dokumentation.

Nach dem Aktivieren der Replikation wird der Server gestartet, um die Map im Client zu replizieren. Irgendwann einmal ist der Client im Vergleich mit dem Server nur ein paar Transaktionen im Rückstand.

Katalogservice mit hoher Verfügbarkeit

Eine Katalogservicedomäne ist das Datengrid der verwendeten Katalogserver, die Topologieinformationen für alle Container-Server in Ihrer eXtreme-Scale-Umgebung enthalten. Der Katalogservice steuert den Lastausgleich und das Routing für alle Clients.

Weitere Informationen zu Katalogservern finden Sie unter „Katalogservice” auf Seite 18.

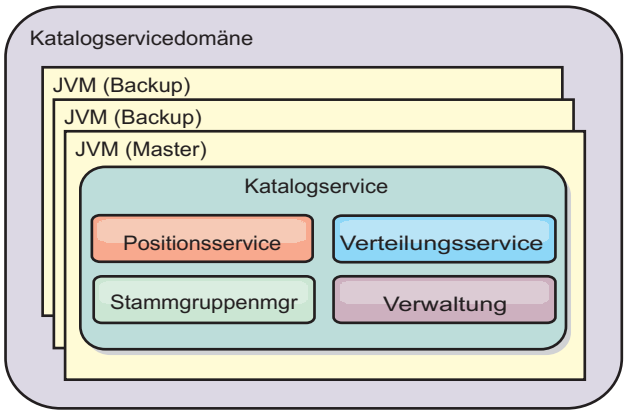


Abbildung 32. Katalogservicedomäne

Wenn mehrere Katalogserver gestartet werden, wird einer der Server als Masterkatalogserver ausgewählt. Dieser Masterserver akzeptiert Überwachungssignale und bearbeitet Systemdatenänderungen, die sich aufgrund von Änderungen im Katalogservice oder in den Containern ergeben.

Konfigurieren Sie mindestens drei Katalogserver in der Katalogservicedomäne. Katalogserver müssen auf separaten Knoten oder in anderen Installationsimages als Ihre Container-Server installiert werden, um sicherzustellen, dass Sie Ihre Server später nahtlos aktualisieren können. Wenn Ihre Konfiguration Zonen enthält, können Sie einen Katalogserver pro Zone konfigurieren.

Wenn ein Container-Server eine Verbindung zu einem der Katalogserver herstellt, wird auch die Routing-Tabelle über den CORBA-Servicekontext für die Katalogservicedomäne an den Katalogserver und den Container-Server übergeben. Ist der kontaktierte Katalogserver derzeit nicht der Masterkatalogserver, wird die Anforderung automatisch an den aktuellen Masterkatalogserver umgeleitet und die Routing-Tabelle für den Katalogserver aktualisiert.

Anmerkung: Die Datengrids einer Katalogservicedomäne und eines Container-Servers sind sehr verschieden. Die Katalogservicedomäne ist für die hohe Verfüg-

barkeit Ihrer Systemdaten verantwortlich. Das Datengrind des Container-Servers ist für die hohe Verfügbarkeit, die Skalierbarkeit und das Workload-Management Ihrer Daten verantwortlich. Deshalb sind zwei verschiedene Routing-Tabellen vorhanden: die Routing-Tabelle für die Katalogservicedomäne und die Routing-Tabelle für die Datengrid-Shards des Container-Servers.

Katalogserverquorum

Wenn der Quorummechanismus aktiviert ist, müssen alle Katalogserver im Quorum verfügbar sein, damit Verteilungsoperationen im Datengrid stattfinden können.

- „Wichtige Begriffe"
- „Überwachungssignale und Fehlererkennung"
- „Quorumverhalten" auf Seite 113
  - „Containerverhalten während des Quorumverlusts" auf Seite 116
- „Clientverhalten während des Quorumverlusts" auf Seite 116

Wichtige Begriffe

- **Überwachungssignal:** Ein Signal, das zwischen Servern ausgetauscht wird, um festzustellen, ob sie aktiv sind.
- **Quorum**: Eine Gruppe von Katalogservern, die miteinander kommunizieren und Verteilungsoperationen im Datengrid durchführen. Diese Gruppe setzt sich aus allen Katalogservern im Datengrid zusammen, sofern Sie den Quorummechanismus nicht mit Verwaltungsaktionen überschreiben.
- **Brownout**: Ein temporärer Verlust der Konnektivität zwischen Servern.
- **Blackout**: (Ausfall) Ein permanenter Verlust der Konnektivität zwischen Servern.
- **Rechenzentrum:** Eine sich an einem bestimmten Standort befindliche Gruppe von Servern, die im Allgemeinen über ein lokales Netz (LAN, Local Area Netzwork) miteinander verbunden sind.
- **Zone:** Eine Zone ist eine Konfigurationsoption, die verwendet wird, um Server zu gruppieren, die ein gemeinsames physisches Merkmal haben. Beispiele für Zonen für eine Gruppe von Servern sind ein Rechenzentrum, ein Bereichsnetz, ein Gebäude oder ein Stockwerk.

Überwachungssignale und Fehlererkennung

Container-Server und Stammgruppen

Der Katalogservice verteilt Container-Server auf Stammgruppen begrenzter Größe. Eine Stammgruppe versucht, den Ausfall ihrer Member zu erkennen. Ein einziges Member jeder Stammgruppe wird als führendes Member der Stammgruppe bestimmt. Das führende Member der Stammgruppe teilt dem Katalogservice in regelmäßigen Abständen den Aktivitätsstatus der Stammgruppe und alle Änderungen der Stammgruppenzugehörigkeiten mit. Eine Zugehörigkeitsänderung kann beispielsweise der Ausfall einer JVM oder das Hinzufügen einer neuen JVM sein, die in die Stammgruppe aufgenommen wird.

Wenn der Socket einer JVM geschlossen wird, wird diese JVM als nicht mehr verfügbar eingestuft. Außerdem sendet jedes Member der Stammgruppe in einem in der Konfiguration festgelegten Intervall Überwachungssignale über diese Sockets. Wenn eine JVM nicht innerhalb eines konfigurierten maximalen Zeitraums auf diese Überwachungssignale reagiert, wird die JVM als nicht mehr verfügbar eingestuft, woraufhin eine Fehlererkennung ausgelöst wird.

Wenn der Katalogservice eine Container-JVM als ausgefallen markiert und der Container-Server später als wieder verfügbar gemeldet wird, wird die Container-JVM angewiesen, die Container-Server von WebSphere eXtreme Scale zu beenden. Eine JVM in diesem Status ist in den Befehlsabfragen des Dienstprogramms **xscmd** nicht sichtbar. Die Nachrichten in den Protokollen der Container-JVM weisen darauf hin, dass die Container-JVM ausgefallen ist. Sie müssen diese JVM manuell erneut starten.

Wenn das führende Member der Stammgruppe zu einem Member eine Verbindung herstellen kann, setzt das führende Member seine Versuche, die Verbindung zu diesem Member herzustellen, fort.

Der vollständige Ausfall aller Member einer Stammgruppe ist auch möglich. Wenn die vollständige Stammgruppe ausfällt, muss der Katalogservice diesen Verlust erkennen.

Austausch von Überwachungssignalen in einer Katalogservicedomäne

Die Katalogservicedomäne gleicht einer privaten Stammgruppe mit einer statischen Zugehörigkeit und einem Quorummechanismus. Sie erkennt Ausfälle auf dieselbe Weise wie eine normale Stammgruppe. Das Verhalten ist jedoch anders und umfasst eine Quorum-Logik. Außerdem verwendet der Katalogservice eine weniger aggressive Konfiguration für den Austausch von Überwachungssignalen.

Ausfallerkennung

WebSphere eXtreme Scale erkennt, wenn Prozesse durch abnormale Socketschließungen beendet werden. Der Katalogservice wird sofort benachrichtigt, wenn ein Prozess beendet wird.

Weitere Einzelheiten zum Konfigurieren des Austauschs von Überwachungssignalen finden Sie in Einstellung für das Intervall der Überwachungssignale für Failover-Erkennung optimierenden Informationen zum Konfigurieren der Failovererkennung in der Veröffentlichung *Verwaltung*.

Quorumverhalten

Normalerweise haben die Member des Katalogservice vollständige Konnektivität. Die Katalogservicedomäne ist eine statische Gruppe von JVMs. WebSphere eXtreme Scale erwartet, dass alle Member des Katalogservice online sind. Wenn alle Member online sind, hat der Katalogservice das Quorum. Der Katalogservice reagiert nur auf Containerereignisse, wenn der Katalogservice das Quorum hat.

Ursache für Quorumverlust

WebSphere eXtreme Scale erwartet in den folgenden Szenarien einen Quorumverlust:

- Das Member mit der Katalogservice-JVM fällt aus.
- Es tritt ein Netz-Brownout ein.
- Es tritt ein Ausfall des Rechenzentrums ein.

WebSphere eXtreme Scale verliert das Quorum im folgenden Szenario nicht:

- Eine Katalogserverinstanz wird mit dem Befehl **stopOgServer** oder einer anderen Verwaltungsaktion gestoppt. Das System weiß, dass die Serverinstanz gestoppt wurde, was bei einem JVM-Fehler oder Brownout anders ist.

Verliert der Katalogservice ein Quorum, wartet er, bis das Quorum wiederhergestellt ist. Während der Katalogservice kein Quorum hat, ignoriert er Ereignisse von Container-Servern. Container-Server wiederholen alle Anforderungen, die vom Katalogserver in der Zeit zurückgewiesen werden. Der Austausch von Überwachungssignalen wird ausgesetzt, bis das Quorum wiederhergestellt ist.

Quorumverlust nach JVM-Ausfall

Ein Katalogserver, der ausfällt, führt zu einem Quorum-Verlust. Wenn eine JVM ausfällt, müssen Sie das Quorum so schnell wie möglich wiederherstellen. Ein ausgefallener Katalogservice kann dem Datengrid erst dann wieder beitreten, wenn das Quorum wiederhergestellt ist.

Quorumverlust aufgrund eines Netz-Brownouts

WebSphere eXtreme Scale ist auf die Möglichkeit von Brownouts vorbereitet. Ein Brownout ist ein temporärer Verlust der Konnektivität zwischen Rechenzentren. Brownouts sind gewöhnlich transient und innerhalb von Sekunden oder Minuten bereinigt. WebSphere eXtreme Scale versucht während eines Brownouts den normalen Betrieb aufrecht zu erhalten. Ein Brownout wird als einzelnes Fehlereignis betrachtet. Es wird davon ausgegangen, dass der Fehler behoben und der normale Betrieb anschließend fortgesetzt wird, ohne dass Aktionen von erforderlich sind.

Ein langer Brownout kann nur durch Benutzereingriff als Blackout klassifiziert werden. Das Quorum muss auf einer Seite des Brownouts wiederhergestellt werden, damit das Ereignis als als Blackout klassifiziert werden kann.

JVM des Katalogservice stoppen und erneut starten

Wenn ein Katalogserver mit dem Befehl **stopOgServer** gestoppt wird, sinkt das Quorum um einen Server. Die verbleibenden Server haben weiterhin das Quorum. Bei einem Neustart des Katalogservers steigt das Quorum wieder auf die vorherige Zahl.

Konsequenzen eines Quorum-Verlusts

Wenn eine Container-JVM ausfällt und das Quorum verloren gegangen ist, findet erst dann eine Wiederherstellung statt, wenn die Brownout-Bedingung behoben ist. In einem Blackout-Szenario findet die Wiederherstellung erst statt, wenn Sie den Befehl zum Überschreiben des Quorums ausführen. Der Quorumverlust und eine Containerausfall werden als doppelter Fehler gewertet, was aber nur selten vorkommt. Aufgrund des doppelten Fehlers können Anwendungen den Schreibzugriff auf Daten verlieren, die in der ausgefallenen JVM gespeichert waren. Wenn das Quorum wiederhergestellt ist, findet die normale Wiederherstellung statt.

Wenn Sie versuchen, einen Container zu starten, während das Quorum verloren ist, wird der Container nicht gestartet.

Während eines Quorum-Verlusts ist eine vollständige Clientkonnektivität zulässig. Wenn während des Quorum-Verlusts keine Containerausfälle oder Konnektivitätsprobleme auftreten, können die Clients weiterhin vollständig mit den Container-Servern interagieren.

Wenn ein Brownout auftritt, haben einige Clients möglicherweise erst dann wieder Zugriff auf die primären Kopieren oder Replikatkopien der Daten, wenn die Brownout-Bedingung behoben ist.

Neue Clients können gestartet werden, weil eine Katalogservice-JVM in jedem Rechenzentrum vorhanden sein muss. Deshalb kann selbst während eines Brownouts mindestens ein Katalogserver von einem Client erreicht werden.

Wiederherstellung des Quorums

Wenn das Quorum aus irgendeinem Grund verloren geht, wird zur Wiederherstellung des Quorums ein Wiederherstellungsprotokoll ausgeführt. Beim Verlust des Quorums wird die Aktivitätsprüfung für die Stammgruppen ausgesetzt, und alle Ausfallberichte werden ignoriert. Nach der Wiederherstellung des Quorums überprüft der Katalogservice alle Stammgruppen, um deren Zugehörigkeit unverzüglich zu bestimmen. Alle zuvor in Container-JVMs gehosteten Shards, die als ausgefallen gemeldet wurden, werden wiederhergestellt. Wenn primäre Shards verloren gegangen sind, werden die noch aktiven Replikate zu primären Shards hochgestuft. Wenn Replikat-Shards verloren gegangen sind, werden zusätzliche Replikate in den noch aktiven JVMs erstellt.

Quorum überschreiben

Das Quorum sollte überschrieben werden, wenn ein Rechenzentrum ausfällt. Ein Quorumverlust aufgrund des Ausfalls einer Container-Service-JVM oder eines Netz-Brownouts wird automatisch behoben, sobald die Katalogservice-JVM erneut gestartet bzw. das Netz-Brownout behoben ist.

Nur Administratoren haben Kenntnis von einem Ausfall eines Rechenzentrums. WebSphere eXtreme Scale behandelt Brownouts und Blackouts ähnlich. Sie müssen die Umgebung von WebSphere eXtreme Scale über solche Ausfälle mit dem Befehl **xscmd -c overrideQuorum** informieren. Auf diese Weise wird dem Katalogservice mitgeteilt, davon auszugehen, dass das Quorum mit den aktuellen Zugehörigkeiten erreicht ist und eine vollständige Wiederherstellung stattfindet. Wenn Sie einen Befehl zum Überschreiben des Quorums absetzen, bestätigen Sie, dass die JVMs im ausgefallenen Rechenzentrum wirklich ausgefallen sind und nicht wiederhergestellt werden.

In der folgenden Liste sind einige Szenarien für das Überschreiben des Quorums beschrieben. In diesem Szenario haben Sie drei Katalogserver: A, B und C.

- **Brownout:** Der Katalogserver C wird vorübergehend isoliert. Der Katalogservice verliert das Quorum und wartet auf die Behebung des Brownouts. Nach Behebung des Brownout wird der Katalogserver C wieder in die Katalogservicedomäne eingebunden, und das Quorum wird wiederhergestellt. Ihre Anwendung stellt während dieser Zeit keine Probleme fest.
- **Temporärer Ausfall:** Während eines temporären Ausfalls fällt der Katalogserver C aus, und der Katalogservice verliert das Quorum. Sie müssen das Quorum überschreiben. Nach der Wiederherstellung des Quorums können Sie den Katalogserver C erneut starten. Der Katalogserver C wird nach dem Neustart wieder in die Katalogservicedomäne eingebunden. Ihre Anwendung stellt während dieser Zeit keine Probleme fest.
- **Ausfall eines Rechenzentrums:** Sie verifizieren, dass das Rechenzentrum wirklich ausgefallen und vom Netz isoliert ist. Anschließend setzen Sie den Befehl **xscmd -c overrideQuorum** ab. Die anderen beiden noch aktiven Rechenzentren führen eine vollständige Wiederherstellung durch, indem sie Shards, die sich im ausgefallenen Rechenzentrum befinden, ersetzen. Der Katalogservice wird jetzt mit vollständigem Quorum der Katalogserver A und B ausgeführt. Die Anwendung kann Verzögerungen oder Ausnahmen während des Zeitraums zwischen

dem Start des Blackouts und Überschreiben des Quorums feststellen. Nach dem Überschreiben des Quorums wird das Datengrid wiederhergestellt, und der normale Betrieb wird fortgesetzt.

- **Wiederherstellung des Rechenzentrums:** Die noch aktiven Rechenzentren werden bereits mit überschriebenem Quorum ausgeführt. Wenn das Rechenzentrum, in dem Katalogserver C enthalten ist, erneut gestartet wird, müssen alle JVMs im Rechenzentrum erneut gestartet werden. Anschließend wird Katalogserver C wieder in die vorhandene Katalogservicedomäne eingebunden, und das Quorum wird ohne Benutzereingriff wiederhergestellt.
- **Ausfall des Rechenzentrums und Brownout:** Das Rechenzentrum, das den Katalogserver C enthält, fällt aus. Das Quorum wird überschrieben und in den verbleibenden Rechenzentren wiederhergestellt. Wenn ein Brownout zwischen den Katalogservern A und B auftritt, werden die normalen Wiederherstellungsregeln bei Brownouts angewendet. Nach der Behebung des Brownouts wird das Quorum wiederhergestellt, und die erforderliche Wiederherstellung nach einem Quorum-Verlust findet statt.

Containerverhalten während des Quorumverlusts

Container enthalten einen oder mehrere Shards. Shards sind primäre Kopien oder Replikatkopien für eine bestimmte Partition. Der Katalogservice ordnet Shards einem Container zu, und der Container berücksichtigt diese Zuordnung so lange, bis er neue Anweisungen vom Katalogservice erhält. Ein primäres Shard setzt seine Kommunikationsversuche mit seinen Replikat-Shards beispielsweise während Netz-Brownouts fort, bis der Katalogservice dem primären Shard weitere Anweisungen bereitstellt.

Verhalten synchroner Replikate

Das primäre Shard kann neue Transaktionen akzeptieren, während die Verbindung unterbrochen ist, wenn die Anzahl der Online-Replikate mindestens so hoch ist wie der Wert der Eigenschaft **minsync** für das MapSet. Wenn neue Transaktionen im primären Shard verarbeitet werden, während die Verbindung zum synchronen Replikat unterbrochen ist, wird der Replikatinhalt gelöscht und nach Wiederherstellung der Verbindung mit dem aktuellen Status des primären Shards synchronisiert.

Die synchrone Replikation zwischen Rechenzentren und die synchrone Replikation über WAN-Verbindungen wird nicht empfohlen.

Verhalten asynchroner Replikate

Wenn eine Verbindung unterbrochen ist, kann das primäre Shard neue Transaktionen akzeptieren. Das primäre Shard puffert die Änderungen bis zu einem bestimmten Grenzwert. Wenn die Verbindung mit dem Replikat wiederhergestellt wird, bevor der Grenzwert erreicht ist, wird das Replikat mit den gepufferten Änderungen aktualisiert. Beim Erreichen des Grenzwerts löscht das primäre Shard die gepufferte Liste, und bei der Wiederherstellung der Verbindung zum Replikat wird der Replikatinhalt gelöscht und neu mit dem primären Shard synchronisiert.

Clientverhalten während des Quorumverlusts

Unabhängig davon, ob die Katalogservicedomäne das Quorum hat oder nicht, können Clients für das Bootstrapping des Datengrids immer eine Verbindung zum Katalogserver herstellen. Der Client versucht, eine Verbindung zu einer Katalogser-

verinstanz herzustellen, um eine Routentabelle anzufordern und anschließend mit dem Datengrid zu interagieren. Die Netzkonnektivität kann die Interaktion des Clients mit einigen Partitionen aufgrund der Netzkonfiguration verhindern. Der Client kann für den Abruf ferner Daten eine Verbindung zu lokalen Replikaten herstellen, sofern er entsprechend konfiguriert ist. Clients können keine Daten aktualisieren, wenn die primäre Partition für diese Daten nicht verfügbar ist.

Replikate und Shards

Mit eXtreme Scale kann eine speicherinterne Datenbank oder ein Shard von einer Java Virtual Machine (JVM) in einer anderen repliziert werden. Ein Shard stellt eine Partition dar, die in einen Container gestellt wird. Mehrere Shards, die unterschiedliche Partitionen darstellen, können in einem einzigen Container enthalten sein. Jede Partition hat eine Instanz, die ein primäres Shard ist, und eine konfigurierbare Anzahl an Replikat-Shards. Die Replikat-Shards sind synchron der asynchron. Die Typen und die Verteilung der Replikat-Shards werden von eXtreme Scale über eine Implementierungsrichtlinie bestimmt, die die Mindest- und Maximalanzahl synchroner und asynchroner Shards festlegt.

Shard-Typen

Für die Replikation werden drei Typen von Shards verwendet:

- primäre Shards,
- synchrone Replikate,
- asynchrone Replikate.

Das primäre Shard empfängt alle Einfüge-, Aktualisierungs- und Entfernungsoperationen. Das primäre Shard fügt Replikate hinzu und entfernt Replikate, repliziert Daten in den Replikaten und verwaltet Commit- und Rollback-Operationen für Transaktionen.

Synchrone Replikate haben denselben Status wie das primäre Shard. Wenn ein primäres Shard Daten in einem synchronen Replikat repliziert, wird die Transaktion erst festgeschrieben, wenn sie im synchronen Replikat festgeschrieben wurde.

Asynchrone Replikate können denselben Status haben wie das primäre Shard. Wenn ein primäres Shard Daten in einem asynchronen Replikat repliziert, wartet das primäre Shard nicht auf die Festschreibung durch das asynchrone Replikat.

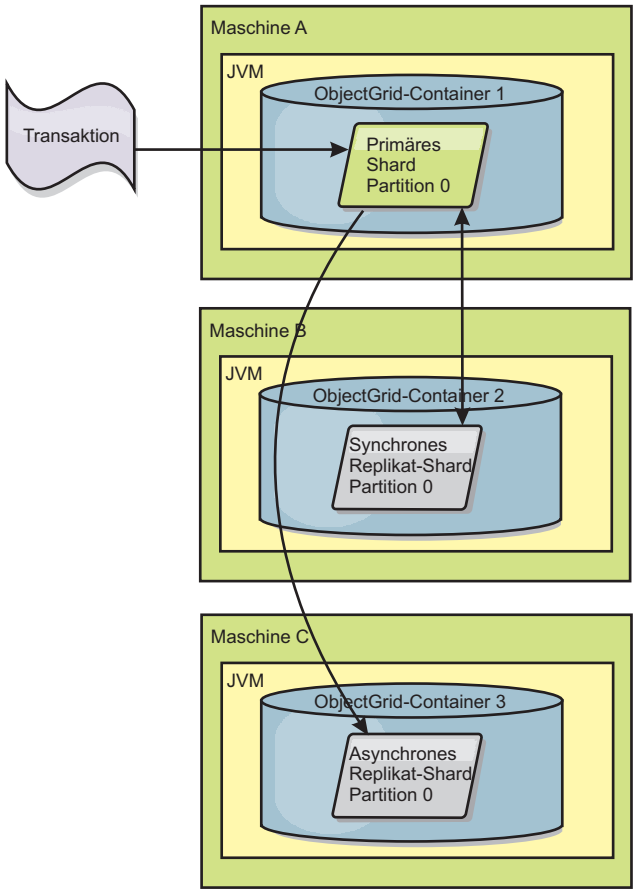


Abbildung 33. Kommunikationspfad zwischen einem primären Shard und einem Replikat-Shard

Mindestanzahl synchroner Replikat-Shards

Wenn ein primäres Shard die Festschreibung von Daten vorbereitet, prüft es, wie viele synchrone Replikat-Shards für die Festschreibung der Transaktion gestimmt haben. Wenn die Transaktion im Replikat normal verarbeitet wird, stimmt das Replikat der Festschreibung zu. Wenn im synchronen Replikat etwas schiefgelaufen ist, stimmt das Replikat der Festschreibung nicht zu. Bevor ein primäres Shard Daten festschreibt, muss die Anzahl synchroner Replikat-Shards, die für die Festschreibung gestimmt haben, der minSyncReplica-Einstellung in der Implementierungsrichtlinie entsprechen. Wenn die Anzahl synchroner Replikat-Shards, die der Festschreibung zustimmen, zu niedrig ist, schreibt das primäre Shard die Transaktion nicht fest, und es tritt ein Fehler auf. Diese Aktion stellt sicher, dass die erforderliche Anzahl synchroner Replikate mit den korrekten Daten verfügbar ist. Synchrone Replikate, in denen Fehler aufgetreten sind, nehmen ihre Registrierung zurück, um ihren Zustand zu korrigieren. Weitere Informationen zum Zurücknehmen der Registrierung finden Sie im Abschnitt Wiederherstellung von Replikat-Shards.

Das primäre Shard löst eine Fehler des Typs "ReplicationVotedToRollbackTransactionException" aus, wenn die Anzahl synchroner Replikate, die der Festschreibung zugestimmt haben, zu niedrig ist.

Replikation und Loader

Normalerweise schreibt ein primäres Shard synchron über den Loader in eine Datenbank. Der Loader und die Datenbank sind immer synchronisiert. Wenn das primäre Shard von einem Replikat-Shard übernommen wird, sind die Datenbank und der Loader möglicherweise nicht mehr synchronisiert. Beispiel:

- Das primäre Shard kann die Transaktion an das Replikat senden und dann vor der Festschreibung der Daten in der Datenbank ausfallen.
- Das primäre Shard kann die Daten in der Datenbank festschreiben und dann vor dem Senden der Daten an das Replikat ausfallen.

Beide Fälle führen dazu, dass das Replikat mit einer Transaktion gegenüber der Datenbank im Vorlauf bzw. im Rückstand ist. Diese Situation ist nicht akzeptabel. eXtreme Scale verwendet ein spezielles Protokoll und einen Vertrag mit der Loader-Implementierung, um dieses Problem ohne zweiphasige Festschreibung zu lösen. Das Protokoll folgt:

Seite des primären Shard

- Transaktion zusammen mit den vorherigen Transaktionsergebnissen senden.
- In die Datenbank schreiben und versuchen, die Transaktion festzuschreiben.
- Wenn die Datenbank festschreibt, dann in eXtreme Scale festschreiben. Wenn die Datenbank nicht festschreibt, Transaktion rückgängig machen.
- Ergebnis aufzeichnen.

Replikatseite

- Transaktion empfangen und puffern.
- Für alle Ergebnisse, die mit der Transaktion geändert werden, die gepufferten Transaktionen festschreiben und alle rückgängig gemachten Transaktionen verwerfen.

Replikatseite bei Failover

- Für alle gepufferten Transaktionen die Transaktionen dem Loader bereitstellen, und der Loader versucht, die Transaktionen festzuschreiben.
- Der Loader muss so geschrieben sein, dass jede Transaktion idempotent ist.
- Wenn die Transaktion bereits in der Datenbank vorhanden ist, führt der Loader keine Operation durch.
- Wenn die Transaktion noch nicht in der Datenbank vorhanden ist, wendet der der Loader die Transaktion an.
- Nachdem alle Transaktionen verarbeitet wurden, kann das neue primäre Shard mit der Bearbeitung der Anforderungen beginnen.

Dieses Protokoll stellt sicher, dass die Datenbank denselben Stand wie das neue primäre Shard hat.

Shardverteilung

Der Katalogservice ist für die Verteilung von Shards verantwortlich. Jedes ObjectGrid hat eine Reihe von Partitionen, die jeweils ein primäres Shard und einen optionalen Satz an Replikat-Shards haben. Der Katalogservice ordnet die Shards zu, indem er sie gleichmäßig auf die verfügbaren Container-Server verteilt. Replikat- und primäre Shards für dieselbe Partition werden nie an denselben Container-Server oder dieselbe IP-Adresse verteilt, es sei denn, die Konfiguration befindet sich im Entwicklungsmodus.

Wenn ein neuer Container-Server gestartet wird, ruft eXtreme Scale Shards aus relativ überladenen Container-Servern in den neuen leeren Container-Server ab. Diese Verschiebung von Shards ermöglicht eine horizontale Skalierung.

Horizontale Vorwärtsskalierung (Scale-out)

Horizontale Vorwärtsskalierung (Scale-out) bedeutet Folgendes: Wenn einem Datengrid zusätzliche Container-Server hinzugefügt werden, versucht eXtreme Scale, vorhandene Shards (primäre Shards oder Replikate) aus dem alten Satz der Container-Server in den neuen Satz zu verschieben. Diese Verschiebung erweitert das Datengrid, sodass Prozessor, des Netz und Hauptspeicher der neu hinzugefügten Container-Server genutzt werden können. Durch die Verschiebung wird auch das Gleichgewicht im Datengrid hergestellt und versucht sicherzustellen, dass jede JVM im Datengrid dasselbe Datenvolumen hostet. Durch die Erweiterung des Datengrids wird erreicht, dass jeder Server einen kleineren Anteil des Gesamtgrids hostet. eXtreme Scale geht davon aus, dass die Daten gleichmäßig auf die Partitionen verteilt sind. Diese Erweiterung ermöglicht eine horizontale Vorwärtsskalierung.

Horizontale Rückskalierung (Scale-In)

Horizontale Rückskalierung (Scale-in) bedeutet Folgendes: Wenn eine JVM ausfällt, versucht eXtreme Scale, den Schaden zu beheben. Besitzt die ausgefallene JVM ein Replikat, ersetzt eXtreme Scale das verloren gegangene Replikat durch Erstellung eines neuen Replikats in einer noch aktiven JVM. Besitzt die ausgefallene JVM ein primäres Shard, sucht eXtreme Scale das am besten geeignete Replikat in den noch aktiven JVMs und stuft dieses Replikat als neues primäres Shard hoch. Anschließend ersetzt eXtreme Scale das hochgestufte Replikat durch ein neues Replikat, das in den verbleibenden Servern erstellt wird. Zur Gewährleistung der Skalierbarkeit, behält eXtreme Scale die Replikatanzahl für Partitionen beim Ausfall von Servern bei.

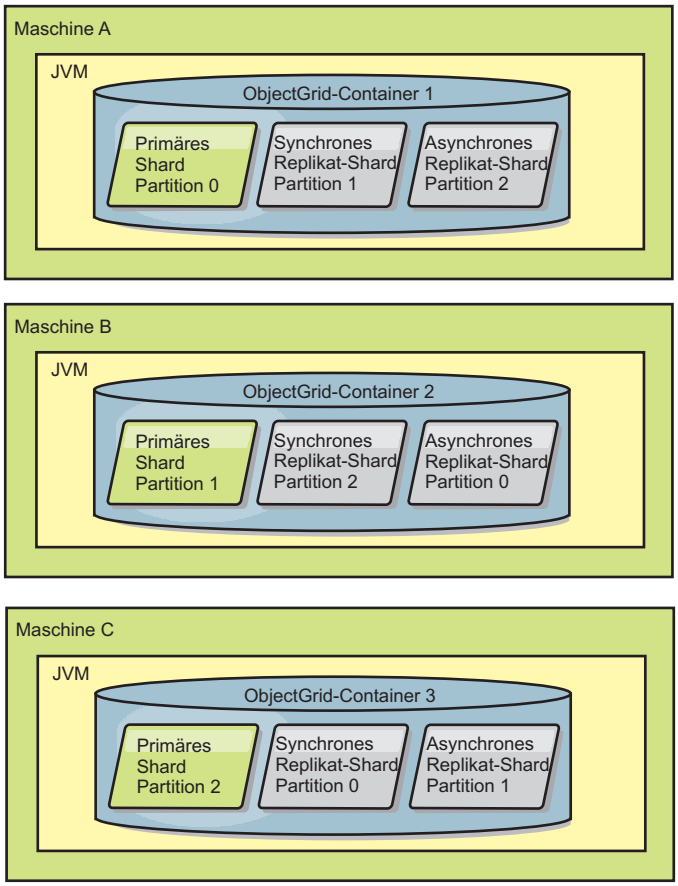


Abbildung 34. Verteilung eines ObjectGrid-MapSets über eine Implementierungsrichtlinie mit 3 Partitionen mit einem minSyncReplicas-Wert von 1, einem maxSyncReplicas-Wert von 1 und einem maxAsyncReplicas-Wert von 1

Aus Replikaten lesen

Sie können MapSets so konfigurieren, dass ein Client aus einem Replikat lesen kann und nicht nur auf die primären Shards beschränkt ist.

Häufig kann es von Vorteil sein, Replikate nicht nur einfach als potenzielle primäre Shards für Fehlerfälle einzusetzen. MapSets können beispielsweise so konfiguriert werden, dass Leseoperationen an Replikate weitergeleitet werden können, indem die Option "replicaReadEnabled" im MapSet auf "true" gesetzt wird. Die Standardeinstellung ist "false".

Weitere Informationen zum Element "MapSet" finden Sie im Abschnitt zur XML-Deskriptordatei für Implementierungsrichtlinien in der Veröffentlichung *Verwaltung*.

Durch die Aktivierung des Lesens von Replikaten kann die Leistung verbessert werden, indem Leseanforderungen an mehrere Java Virtual Machines verteilt werden. Wenn Sie diese Option nicht aktivieren, werden alle Leseanforderungen, wie z. B. die Methoden "ObjectMap.get" und "Query.getResultIterator", an das primäre Shard weitergeleitet. Wenn replicaReadEnabled auf "true" gesetzt ist, können einige Get-Anforderungen veraltete Daten zurückgeben, sodass eine Anwendung, die diese Option verwendet, in der Lage sein muss, diese Möglichkeit zu tolerieren. Cachefehler treten jedoch nicht auf. Wenn die Daten nicht im Replikat enthalten sind, wird die Get-Anforderung an das primäre Shard umgeleitet und wiederholt.

Die Option "replicaReadEnabled" kann mit synchroner und asynchroner Replikation verwendet werden.

Lastausgleich mit Replikaten

Der Lastausgleich mit Replikaten wird gewöhnlich nur verwendet, wenn Clients Daten zwischenspeichern, die sich ständig ändern oder wenn die Clients pessimistisches Sperren verwenden.

eXtreme Scale sendet, sofern nicht anders konfiguriert, alle Lese- und Schreibanforderungen an den primären Server für eine bestimmte Replikationsgruppe. Der primäre Server muss alle Anforderungen von Clients bearbeiten. Sie können konfigurieren, dass Leseanforderungen auch an Replikate des primären Servers gesendet werden. Durch das Senden von Leseanforderungen an die Replikate kann die Last der Leseanforderungen auf mehrere Java Virtual Machines (JVM) verteilt werden. Die Verwendung von Replikaten für Leseanforderungen kann jedoch zu inkonsistenten Antworten führen.

Der Lastausgleich mit Replikaten wird gewöhnlich nur verwendet, wenn Clients Daten zwischenspeichern, die sich ständig ändern oder wenn die Clients pessimistisches Sperren verwenden.

Wenn sich die Daten ständig ändern und anschließend im nahen Cache des Clients ungültig gemacht werden, sollte der primäre Server daraufhin eine relativ hohe Rate von get-Anforderungen von Clients sehen. Im pessimistischen Sperrmodus ist kein lokaler Cache vorhanden, also werden alle Anforderungen an den primären Server gesendet.

Wenn die Daten relativ statisch sind oder der pessimistische Modus nicht verwendet wird, hat das Senden von Leseanforderungen an das Replikat keine erheblichen Auswirkungen auf die Leistung. Die Häufigkeit der get-Anforderungen von Clients mit Caches, die voll von Daten sind, ist nicht sehr hoch.

Wenn ein Client zum ersten Mal gestartet wird, ist sein naher Cache leer. Cacheanforderungen an den leeren Cache werden an den primären Server weitergeleitet. Nach und nach werden Daten in den Clientcache gestellt, sodass die Anforderungslast abnimmt. Wenn sehr viele Clients gleichzeitig gestartet werden, kann die Last erheblich und das Senden von Leseanforderungen an das Replikat eine angemessene Leistungsoption sein.

Shardlebenszyklen

Für die Unterstützung der Replikation setzt sich der Lebenszyklus von Shards aus verschiedenen Stadien und Ereignissen zusammen. Der Lebenszyklus eines Shards setzt sich aus dem Onlinebringen, der Laufzeit, dem Beenden, potenziellen Failover und der Fehlerbehandlung zusammen. Shards können als Reaktion auf einen geänderten Serverstatus von einem Replikat-Shard auf ein primäres Shard hochgestuft werden.

Lebenszyklusereignisse

Wenn primäre Shards und Replikat-Shards verteilt und gestartet werden, finden verschiedene Ereignisse statt, um die Shards online zu bringen und in den Empfangsmodus zu versetzen.

Primäres Shard

Der Katalogservice verteilt ein primäres Shard für eine Partition. Außerdem verteilt der Katalogservice die Positionen der primären Shards gleichmäßig und leitet das Failover für die primären Shards ein.

Wenn ein Shard zu einem primären Shard wird, erhält es eine Liste mit Replikaten vom Katalogservice. Das neue primäre Shard erstellt eine Replikatgruppe und registriert alle Replikate.

Wenn das primäre Shard bereit ist, wird eine Nachricht in der Datei `SystemOut.log` für den Container, in dem das Shard ausgeführt wird, protokolliert, in der darauf hingewiesen wird, dass das Shard für das Geschäft bereit ist. In der Nachricht für die Geschäftsbereitschaft (oder die Nachricht CWOBJ1511I) sind der Mapname, der Name des MapSets und die Partitionsnummer des gestarteten primären Shards aufgelistet.

```
CWOBJ1511I: Mapname:MapSet-Name:Partitionsnummer (primär) ist für Business bereit.
```

Weitere Informationen zur Verteilung der Shards durch den Katalogservice finden Sie im Abschnitt „Shardverteilung” auf Seite 119.

Replikat-Shard

Replikat-Shards werden hauptsächlich vom primären Shard gesteuert, sofern das Replikat-Shard keine Probleme feststellt. Während eines normalen Lebenszyklus ist das primäre Shard für das Verteilen, Registrieren und Zurücknehmen der Registrierung eines Replikat-Shards zuständig.

Wenn ein primäres Shard ein Replikat-Shard initialisiert, wird eine Nachricht im Protokoll angezeigt, die beschreibt, wo das Replikat ausgeführt wird, um so anzuzeigen, dass das Replikat-Shard verfügbar ist. In der Nachricht für die Geschäftsbereitschaft (oder die Nachricht CWOBJ1511I) sind der Mapname, der Name des MapSets und die Partitionsnummer des Replikat-Shards aufgelistet. Diese Nachricht sehen Sie im Folgenden:

```
CWOBJ1511I: Mapname:MapSet-Name:Partitionsnummer (synchrones Replikat) ist für Business bereit.
```

oder

```
CWOBJ1511I: Mapname:MapSet-Name:Partitionsnummer (asynchrones Replikat) ist für Business bereit.
```

Asynchrones Replikat-Shard: Ein asynchrones Replikat-Shard fragt Daten vom primären Shard ab. Das Replikat passt die Abfragetaktung automatisch an, wenn es keine Daten vom primären Shard empängt, was darauf hinweist, das die Verbindung zum primären Shard blockiert ist. Die Taktung wird auch angepasst, wenn das Replikat einen Fehler empfängt, der auf einen Ausfall des primären Shards oder auf ein Netzproblem hinweist.

Wenn das asynchrone Replikat mit der Replikation beginnt, gibt es die folgende Nachricht in seiner Datei "SystemOut.log" aus. Die Nachricht kann pro CWOBJ1511-Nachricht mehrfach ausgegeben werden. Sie wird erneut ausgegeben, wenn das Replikat eine Verbindung zu einem anderen primären Shard herstellt oder wenn Schablonenmaps hinzugefügt werden.

```
CWOBJ1543I: Das asynchrone Replikat objectGrid-Name:MapSet-Name:Partitionsnummer wurde gestarter
bzw. setzt die Replikation über das primäre Shard fort. Replikation für die Maps: [Mapnamen]
```

Synchrones Replikat-Shard: Wenn das synchrone Replikat-Shard gestartet wird, ist es noch nicht im Peer-Modus. Wenn sich ein Replikat-Shard im Peer-Modus befindet, empfängt es Daten vom primären Shard, sobald Daten beim primären Shard

eintreffen. Vor dem Wechsel in den Peer-Modus benötigt das Replikat-Shard eine Kopie aller vorhandenen Daten im primären Shard.

Das synchrone Replikat kopiert ähnlich wie ein asynchrones Replikat Daten vom primären Shard, indem es Daten abfragt. Wenn es die vorhandenen Daten vom primären Shard kopiert, wechselt es in den Peer-Modus und empfängt die Daten, sobald das primäre Shard die Daten empfängt.

Nach dem Wechsel eines Replikat-Shards in den Peer-Modus gibt das Replikat eine Nachricht in seiner Datei "SystemOut.log" aus. Die angegebene Zeit bezieht sich auf die Zeit, die das Replikat-Shard benötigt hat, um seine Anfangsdaten vom primären Shard zu erhalten. Die angezeigte Zeit kann null oder sehr gering sein, wenn das primäre Shard keine zu replizierenden Daten enthält. Die Nachricht kann pro CWOBJ1511-Nachricht mehrfach ausgegeben werden. Sie wird erneut ausgegeben, wenn das Replikat eine Verbindung zu einem anderen primären Shard herstellt oder wenn Schablonenmaps hinzugefügt werden.

```
CWOBJ1526I: Replikat ObjectGrid-Name:Mapset-Name:Partitionsnummer:Mapname wechselt
in X Sekunden in den Peer-Modus.
```

Wenn das synchrone Replikat-Shards im Peer-Modus arbeitet, muss das primäre Shards Transaktionen auf alle im Peer-Modus arbeitenden synchronen Replikate kopieren. Das synchrone Replikat-Shard hat dieselbe Version der Daten wie das primäre Shard. Wenn in der Implementierungsrichtlinie eine minimale Anzahl synchroner Replikate oder minSync definiert ist, muss diese Anzahl synchroner Replikate der Festschreibung zustimmen, bevor die Transaktion erfolgreich auf dem primären Shard festgeschrieben werden kann.

Wiederherstellungsereignisse

Die Replikation ist für die Wiederherstellung nach Ausfällen und Fehlereignissen bestimmt. Wenn ein primäres Shard ausfällt, wird seine Arbeit von einem anderen Replikat übernommen. Treten Fehler bei den Replikat-Shards auf, versucht das Replikat-Shard, eine Recovery durchzuführen. Der Katalogservice steuert die Verteilung und Transaktionen der neuen primären Shards bzw. neuen Replikat-Shards.

Replikat-Shards wird zu einem primären Shard

Ein Replikat-Shard kann aus zwei Gründen zu einem primären Shard werden. Das primäre Shard wird gestoppt oder fällt aus, oder es wurde eine Entscheidung im Sinne der Lastverteilung getroffen, die ein Verschieben des vorherigen primären Shards an eine neue Position erfordert.

Der Katalogservice wählt ein neues primäres Shards unter den vorhandenen synchronen Replikat-Shards aus. Wenn ein primäres Shard verschoben werden muss und keine Replikate vorhanden sind, wird ein temporäres Replikat für den Übergang verwendet. Das neue primäre Shard registriert alle vorhandenen Replikate und akzeptiert anschließend Transaktionen als das neue primäre Shard. Wenn die vorhandenen Replikat-Shards die richtige Datenversion haben, werden die aktuellen Daten beibehalten, wenn sich die Replikat-Shards beim neuen primären Shard registrieren. Asynchrone Replikate fragen das neue primäre Shard ab.

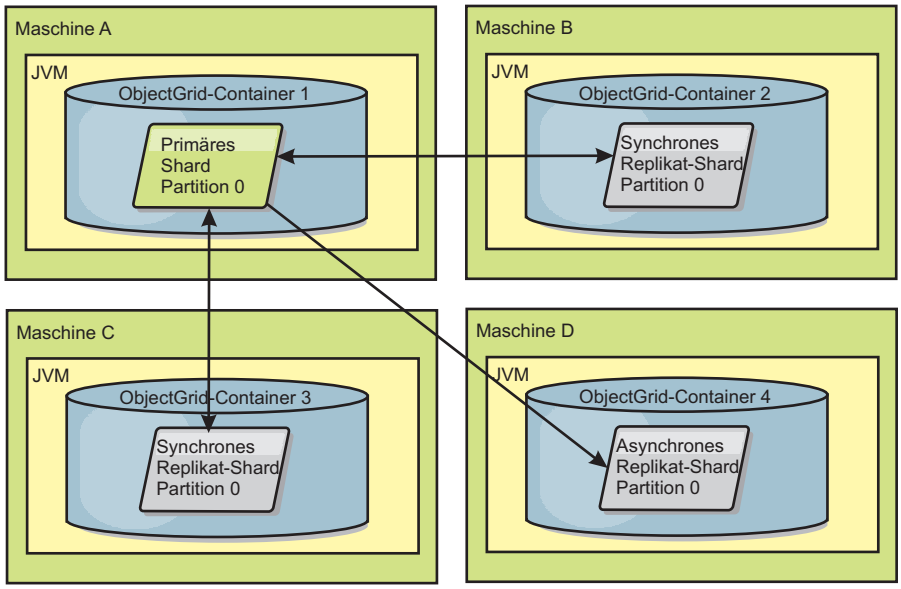


Abbildung 35. Beispielverteilung eines ObjectGrid-MapSets für die Partition "partition0". Die Implementierungsrichtlinie enthält den minSyncReplicas-Wert 1, den maxSyncReplicas-Wert 2 und den maxAsyncReplicas-Wert 1.

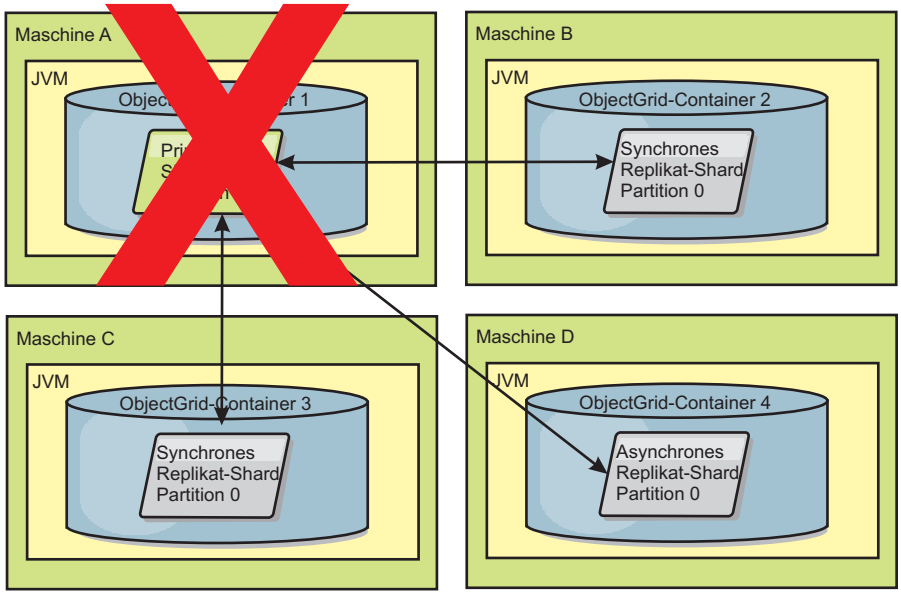


Abbildung 36. Container für das primäre Shard fällt aus

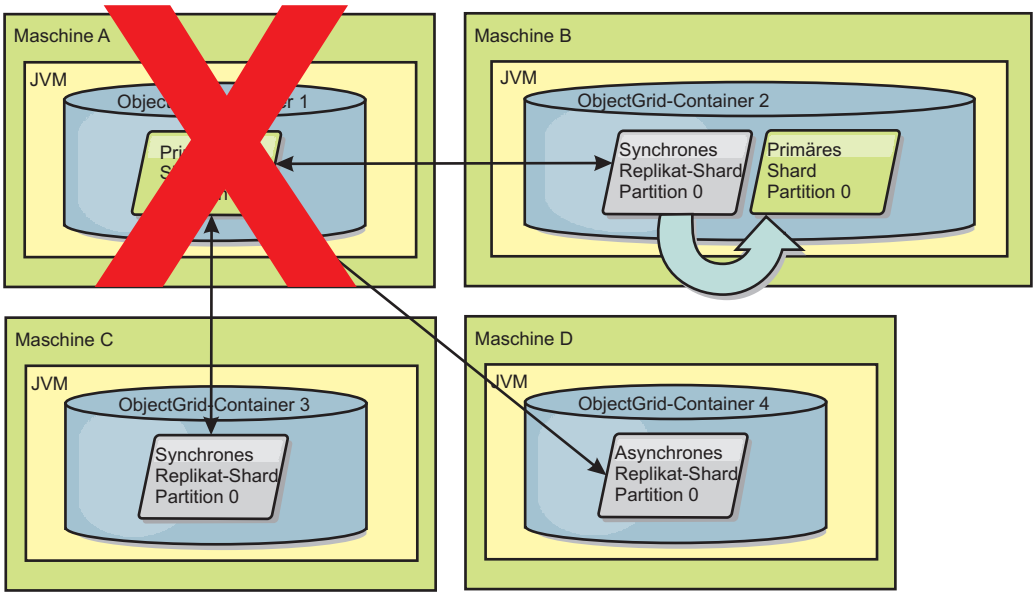


Abbildung 37. Synchrones Replikat-Shard in ObjectGrid-Container 2 wird zum primären Shard

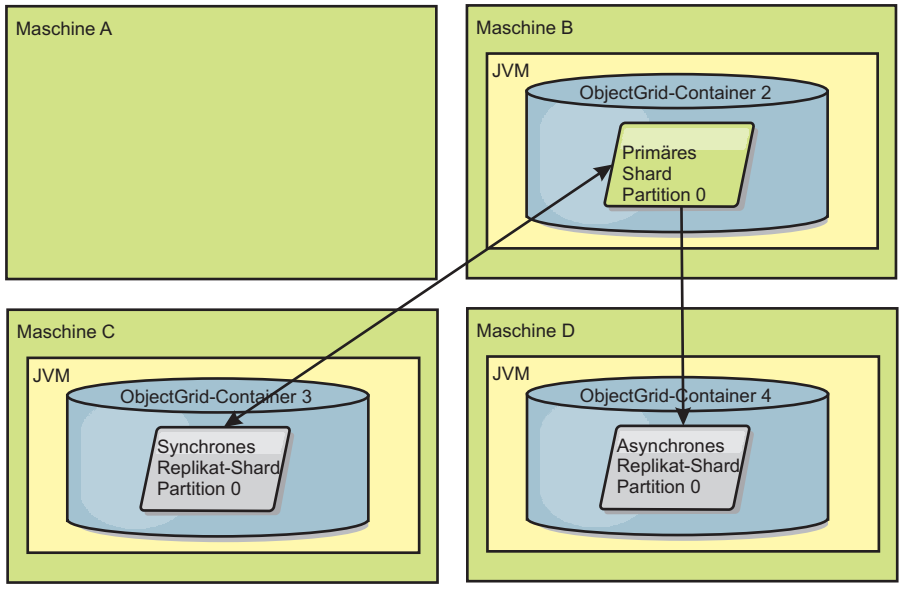


Abbildung 38. Maschine B enthält das primäre Shard. In Abhängigkeit von der Einstellung des Modus für automatische Reparatur und der Verfügbarkeit der Container wird ein neues synchrones Replikat-Shard auf einer Maschine erstellt oder nicht.

Wiederherstellung von Replikat-Shards

Ein synchrones Replikat-Shard wird vom primären Shard gesteuert. Wenn ein Replikat-Shard jedoch ein Problem feststellt, kann es ein Ereignis zur Zurücknahme der Registrierung auslösen, um den Status der Daten zu korrigieren. Das Replikat-Shard löscht die aktuellen Daten und ruft eine aktuelle Kopie vom primären Shard ab.

Wenn ein Replikat-Shard ein Ereignis zur Zurücknahme der Registrierung einleitet, gibt das Replikat eine Protokollnachricht aus:

```
CWOBJ1524I: Replikatlistener
ObjectGrid-Name:MapSet-Name:Partition muss sich beim primären Shard registrieren.
Ursache: Aufgelistete Ausnahme
```

Wenn eine Transaktion während der Verarbeitung einen Fehler in einem Replikat-Shard verursacht, hat das Replikat-Shard einen unbekannten Status. Die Transaktion wird erfolgreich im primären Shard ausgeführt, aber im Replikat ist irgendein Fehler aufgetreten. Zur Behebung des Problems leitet das Replikat ein Ereignis zum Zurücknehmen der Registrierung ein. Mit einer neuen Kopie der Daten vom primären Shard kann das Replikat-Shard seine Verarbeitung fortsetzen. Tritt dasselbe Problem erneut auf, setzt das Replikat-Shard seine Versuche, die Registrierung zurückzunehmen, nicht fort. Weitere Einzelheiten finden Sie im Abschnitt „Fehlerereignisse".

Fehlerereignisse

Ein Replikat kann die Replikation von Daten einstellen, wenn es Fehlersituationen feststellt, die nicht behoben werden können.

Zu viele Registrierungsversuche

Wenn ein Replikat sich mehrfach erneut registriert, ohne die Daten erfolgreich festzuschreiben, wird das Replikat gestoppt. Das Stoppen verhindert, dass ein Replikat in eine Endlosschleife für die Registrierung eintritt. Standardmäßig wiederholt ein Replikat-Shard seinen Registrierungsversuch dreimal nacheinander, bevor es gestoppt wird.

Wenn sich ein Replikat-Shard zu oft neu registriert, gibt es die folgende Nachricht im Protokoll aus:

```
CWOBJ1537E: ObjectGrid-Name:MapSet-Name:Partition hat die zulässige Anzahl
erneuter Registrierungen (timesAllowed) ohne erfolgreiche Transaktionen überschritten.
```

Wenn die Wiederherstellung des Replikats durch erneute Registrierung nicht möglich ist, kann ein tiefgreifendes Problem bei den Transaktionen, die sich auf das Replikat-Shard beziehen, vorliegen. Ein mögliches Problem sind fehlende Ressourcen im Klassenpfad, wenn ein Fehler bei der Dekomprimierung der Schlüssel oder Werte aus der Transaktion auftritt.

Fehler beim Wechsel in den Peer-Modus

Wenn ein Replikat versucht, in den Peer-Modus zu wechseln und ein Fehler bei der Verarbeitung des vorhandenen Datenvolumens vom primären Shard (Prüfpunktdaten) auftritt, wird das Replikat beendet. Das Beenden verhindert, dass ein Replikat mit ungültigen Anfangsdaten gestartet wird. Da das Replikat dieselben Daten vom primären Shard empfängt, wenn es sich erneut registriert, findet keine Wiederholung statt.

Wenn ein Replikat-Shard nicht in den Peer-Modus wechseln kann, gibt es die folgende Nachricht im Protokoll aus:

```
CWOBJ1527W Replikat ObjectGrid-Name:MapSet-Name:Partition:Mapname konnte nach numSeconds Sekunden nicht in den Peer-Modus wechseln.
```

Es wird eine weitere Nachricht im Protokoll angezeigt, die erläutert, warum das Replikat nicht in den Peer-Modus wechseln konnte.

Fehler bei Wiederherstellung nach Neuregistrierung oder beim Wechsel in Peer-Modus

Wenn ein Replikat die erneute Registrierung nicht durchführen oder nicht in den Peer-Modus wechseln kann, bleibt das Replikat so lange inaktiviert, bis ein neues Verteilungsereignis stattfindet. Ein neues Verteilungsereignis kann das Starten oder Stoppen eines neuen Servers sein. Sie können ein Verteilungsereignis auch mit der Methode "triggerPlacement" in der MBean "PlacementServiceMBean" einleiten.

MapSets für die Replikation

Die Replikation wird durch die Zuordnung von BackingMaps zu einem MapSet aktiviert.

Ein MapSet ist eine Sammlung von Maps, die anhand eines Partitionsschlüssels kategorisiert werden. Dieser Partitionsschlüssel wird wie folgt aus dem Schlüssel der jeweiligen Map abgeleitet: Hash-Wert Modulo Partitionsanzahl. Wenn eine Gruppe von Maps im MapSet den Partitionsschlüssel X hat, werden diese Maps in einer entsprechenden Partition X im Datengrid gespeichert. Wenn eine andere Gruppe den Partitionsschlüssel Y hat, werden alle Maps in Partition Y gespeichert usw. Außerdem werden die Daten in den Maps auf der Basis der Richtlinie repliziert, die im MapSet definiert ist. Dies wird nur für verteilte eXtreme-Scale-Topologien verwendet und ist für lokale Instanzen nicht erforderlich.

Weitere Einzelheiten finden Sie im Abschnitt „Partitionierung” auf Seite 87.

Den MapSets wird die Anzahl vorhandener Partitionen und eine Replikationsrichtlinie zugeordnet. In der Replikationskonfiguration des MapSets wird einfach die Anzahl synchroner und asynchroner Replikat-Shards angegeben, die ein MapSet zusätzlich zum primären Shard haben sollte. Wenn beispielsweise ein synchrones und ein asynchrones Replikat verwendet werden sollen, wird für alle BackingMaps, die dem MapSet zugeordnet werden, automatisch jeweils ein Replikat-Shard auf die Gruppe verfügbarer Container für eXtreme Scale verteilt. Die Replikationskonfiguration kann Clients auch das Lesen von Daten aus synchron replizierten Servern ermöglichen. Auf diese Weise kann die Last der Leseanforderungen auf zusätzliche Server in eXtreme Scale verteilt werden. Die Replikation hat nur beim vorherigen Laden der BackingMaps Auswirkungen auf das Programmiermodell.

Übersicht über die Transaktionsverarbeitung

Java

WebSphere eXtreme Scale verwendet Transaktionen als Mechanismus für die Interaktion mit Daten.

Für die Interaktion mit Daten benötigt der Thread in Ihrer Anwendung eine eigene Sitzung. Wenn die Anwendung das ObjectGrid in einem Thread verwenden möchte, rufen Sie eine der Methoden "ObjectGrid.getSession" auf, um eine Sitzung anzufordern. Über das Session-Objekt kann die Anwendung die in den ObjectGrid-Maps gespeicherten Daten bearbeiten.

Wenn eine Anwendung ein Session-Objekt verwendet, muss dieses im Kontext einer Transaktion enthalten sein. Eine Transaktion wird über die Methoden "begin", "commit" und "rollback" des Session-Objekts gestartet und festgeschrieben bzw. rückgängig gemacht. Anwendungen können auch im Modus für automatische Festschreibung arbeiten, in dem das Session-Objekt eine Transaktion automatisch startet und festschreibt, wenn eine Operation in der Map durchgeführt wird. Der Modus für automatische Festschreibung ist nicht in der Lage, mehrere Operationen zu einer einzigen Transaktion zu gruppieren, und damit die langsamere Option, wenn Sie einen Stapel mit mehreren Operationen in einer einzigen Transaktion erstellen.

Für Transaktionen, die nur eine einzige Operation enthalten, ist die automatische Festschreibung jedoch die schnellere Option.

Wenn Ihre Anwendung die Sitzung nicht mehr benötigt, verwenden Sie die optionale Methode Session.close(), um die Sitzung zu schließen. Beim Schließen der Sitzung wird die Sitzung aus dem Heapspeicher freigegeben und kann von nachfolgenden Aufrufen der Methode getSession() wiederverwendet werden, was die Leistung verbessert.

Transaktionen

Java

Transaktionen haben viele Vorteile in Bezug auf die Speicherung und Bearbeitung von Daten. Sie können Transaktionen verwenden, um das Datengrid vor gleichzeitigen Änderungen zu schützen, mehrere Änderungen als Einheit anzuwenden, Daten zu replizieren und einen Lebenszyklus für Sperren bei Änderungen zu implementieren.

Wenn eine Transaktion gestartet wird, ordnet WebSphere eXtreme Scale eine spezielle Differenzmap zu, in der die aktuellen Änderungen bzw. Kopien von Schlüssel/Wert-Paaren gespeichert werden, die von der Transaktion verwendet werden. Normalerweise wird beim Zugriff auf ein Schlüssel/Wert-Paar der Wert kopiert, bevor die Anwendung den Wert erhält. Die Differenzmap verfolgt alle Änderungen, wenn Operationen wie Einfüge-, Aktualisierungs-, Abruf-, Entfernungsoperationen usw. durchgeführt werden. Schlüssel werden nicht kopiert, weil sie als unveränderlich gelten. Wenn ein ObjectTransformer-Objekt angegeben ist, wird dieses Objekt zum Kopieren des Werts verwendet. Wenn die Transaktion optimistisches Sperren verwendet, werden auch Vorher-Kopien der Werte verfolgt, um sie als Vergleichswerte beim Festschreiben der Transaktion zu verwenden.

Wenn eine Transaktion rückgängig gemacht wird, werden die Informationen in der Differenzmap verworfen und Sperren für die Einträge freigegeben. Wenn eine Transaktion festgeschrieben wird, werden die Änderungen auf die Maps angewendet und die Sperren freigegeben. Bei der Verwendung optimistischen Sperrens vergleicht eXtreme Scale die Vorher-Versionen der Werte mit den Werten, die in der Map enthalten sind. Diese Werte müssen übereinstimmen, damit die Transaktion festgeschrieben werden kann. Dieser Vergleich ermöglicht ein Schema für das Sperren mehrerer Versionen. Hierbei entstehen jedoch zusätzliche Kosten, weil zwei Kopien erstellt werden, wenn die Transaktion auf den Eintrag zugreift. Alle Werte werden erneut kopiert, und die neue Kopie wird in der Map gespeichert. WebSphere eXtreme Scale führt diesen Kopiervorgang durch, um sich selbst davor zu schützen, dass die Anwendung die Anwendungsreferenz in den Wert nach Festschreibung ändert.

Sie können dies vermeiden, indem Sie mehrere Kopien der Informationen erstellen. Die Anwendung kann eine Kopie auch über pessimistisches Sperren anstatt über optimistisches Sperren speichern. Dies schränkt jedoch den gemeinsamen Zugriff ein. Die Kopie des Wertes zur Festschreibungszeit kann ebenfalls vermieden werden, wenn die Anwendung zustimmt, einen Wert nach der Festschreibung nicht mehr zu ändern.

Vorteile von Transaktionen

Verwenden Sie Transaktionen für die folgenden Zwecke:

Wenn Sie Transaktionen verwenden, ist Folgendes möglich:

- Änderungen können rückgängig gemacht werden, wenn eine Ausnahme eintritt oder wenn die Geschäftslogik Statusänderungen widerrufen muss.
- zum Anwenden mehrerer Änderungen als atomare Einheit beim Festschreiben,
- Sperren für Daten können gehalten und freigegeben werden, um mehrere Änderungen als atomare Einheit zur Festschreibungszeit anzuwenden.
- Threads werden vor gleichzeitigen Änderungen geschützt.
- Es kann ein Lebenszyklus für Sperren bei Änderungen implementiert werden.
- Es kann eine atomare Replikationseinheit erzeugt werden.

Transaktionsgröße

Größere Transaktionen sind effizienter, insbesondere für die Replikation. Größere Transaktionen können sich jedoch nachteilig auf den gemeinsamen Zugriff auswirken, weil die Sperren für Einträge länger gehalten werden. Wenn Sie größere Transaktionen verwenden, kann dies die Replikationsleistung steigern. Dieser Leistungsanstieg ist wichtig, wenn Sie eine Map vorher laden. Experimentieren Sie mit verschiedenen Stapelgrößen, um festzustellen, welche sich für Ihr Szenario am besten eignet.

Größere Transaktionen sind auch bei Loadern hilfreich. Wenn ein Loader verwendet wird, der SQL-Stapeloperationen durchführen kann, sind je nach Transaktion erhebliche Leistungssteigerungen und auf der Datenbankseite erheblich Lastreduktionen möglich. Diese Leistungssteigerung ist von der Loader-Implementierung abhängig.

Modus für automatische Festschreibung

Wenn keine Transaktion aktiv gestartet wird und eine Anwendung mit einem ObjectMap-Objekt interagiert, wird eine automatische Start- und Festschreibungsoperation für die Anwendung durchgeführt. Diese automatische Start- und Festschreibungsoperation funktioniert, verhindert aber, dass Rollback und Sperren effektiv funktionieren. Die Geschwindigkeit der synchronen Replikation nimmt aufgrund der sehr geringen Transaktionsgröße ab. Wenn Sie eine EntityManager-Anwendung verwenden, sollten Sie den Modus für automatische Festschreibung nicht verwenden, weil Objekte, die mit der Methode EntityManager.find gesucht werden, unmittelbar nach der Rückkehr der Methode nicht mehr verwaltet werden und somit unbrauchbar sind.

Externe Transaktionskoordinatoren

Gewöhnlich beginnt eine Transaktion mit der Methode "session.begin" und endet mit der Methode "session.commit". Wenn eXtreme Scale integriert ist, können die Transaktionen jedoch auch von externen Transaktionskoordinatoren gestartet und beendet werden. Wenn Sie einen externen Transaktionskoordinator verwenden, müssen Sie die Methoden "session.begin" und "session.commit" nicht aufrufen. Wenn Sie WebSphere Application Server verwenden, können Sie das WebSphere-TranscationCallback-Plug-in einsetzen.

Integration von Java-EE-Transaktionen

eXtreme Scale enthält einen mit Java Connector Architecture (JCA) 1.5 kompatiblen Ressourcenadapter, der Clientverbindungen zu einem fernen Datengrid und das Management lokaler Transaktionen unterstützt. Java-EE-Anwendungen (Java Platform, Enterprise Edition) wie Servlets, JSP-Dateien (JavaServer Pages) und EJB-Komponenten (Enterprise JavaBeans) können Transaktionen von eXtreme Scale mit-

hilfe der Standardschnittstelle javax.resource.cci.LocalTransaction oder mithilfe der Sitzungsschnittstelle eXtreme Scale abgrenzen.

Wenn Sie in WebSphere Application Server mit aktivierter Unterstützung des letzten Teilnehmers in der Anwendung arbeiten, können Sie die Transaktion von eXtreme Scale in einer globalen Transaktion zusammen mit anderen transaktionsorientierten Ressourcen mit zweiphasiger Festschreibung registrieren.

Transaktionsverarbeitung in Java-EE-Anwendungen:

WebSphere eXtreme Scale stellt einen eigenen Ressourcenadapter bereit, den Sie verwenden können, um Anwendungen mit dem Datengrid zu verbinden und lokale Transaktionen zu verarbeiten.

Mit der Unterstützung des Ressourcenadapters von eXtreme Scale können Java-EE-Anwendungen (Java Platform, Enterprise Edition) Clientverbindungen von eXtreme Scale suchen und lokale Transaktionen mithilfe lokaler Java-EE-Transaktionen oder mithilfe der APIs von eXtreme Scale abgrenzen. Wenn der Ressourcenadapter konfiguriert ist, können Sie die folgenden Aktionen mit Ihren Java-EE-Anwendungen ausführen:

- Verbindungsfactory des Ressourcenadapters von eXtreme Scale suchen oder in eine Java-EE-Anwendungskomponente injizieren
- Standardverbindungshandles zum Client von eXtreme Scale anfordern und unter Einhaltung der Java-EE-Konventionen in Anwendungskomponenten gemeinsam nutzen
- eXtreme-Scale-Transaktionen mithilfe der API javax.resource.cci.LocalTransaction oder der Schnittstelle com.ibm.websphere.objectgrid.Session abgrenzen
- Gesamte Client-API von eXtreme Scale verwenden, wie z. B. ObjectMap oder EntityManager

Die folgenden zusätzlichen Funktionen stehen mit WebSphere Application Server zur Verfügung:

- Registrieren von Verbindungen von eXtreme Scale bei einer globalen Transaktion als letzter Teilnehmer mit anderen Ressourcen für zweiphasige Festschreibung. Der Ressourcenadapter von eXtreme Scale bietet Unterstützung für lokale Transaktionen mit einer Ressource für einphasige Festschreibung. WebSphere Application Server ermöglicht Anwendungen, über die Unterstützung des letzten Teilnehmers eine einzige Ressource für einphasige Festschreibung bei einer globalen Transaktion zu registrieren.
- Automatische Installation des Ressourcenadapters bei Erweiterung des Profils
- Automatische Weitergabe des Sicherheitsprincipals

Zuständigkeiten des Administrators

Der Ressourcenadapter von eXtreme Scale wird im Java-EE-Anwendungsserver installiert oder mit der Anwendung eingebettet. Nach der Installation des Ressourcenadapters erstellt der Administrator eine oder mehrere Verbindungsfactory des Ressourcenadapters für jede Katalogservicedomäne und optional jede Datengridinstanz. Die Verbindungsfactory identifiziert die Eigenschaften, die erforderlich sind, um mit dem Datengrid zu kommunizieren.

Anwendungen referenzieren die Verbindungsfactory, woraufhin die Verbindung zum fernen Datengrid aufgebaut wird. Jede Verbindungsfactory hostet eine einzelne Clientverbindung von eXtreme Scale, die für alle Anwendungskomponenten wiederverwendet wird.

Wichtig: Da die Clientverbindung von eXtreme Scale einen nahen Cache enthalten kann, dürfen Anwendungen Verbindungen nicht gemeinsam nutzen. Es muss eine Verbindungsfactory für jede Anwendungsinstanz vorhanden sein, um Probleme mit der gemeinsamen Nutzung von Objekten durch mehrere Anwendungen zu vermeiden.

Die Verbindungsfactory hostet eine Clientverbindung von eXtreme Scale, die von allen referenzierenden Anwendungskomponenten gemeinsam genutzt wird. Sie können eine Managed Bean (MBean) verwenden, um auf Informationen zur Clientverbindung zuzugreifen oder um die Verbindung zurückzusetzen, wenn sie nicht mehr benötigt wird.

Zuständigkeiten des Anwendungsentwicklers

Ein Anwendungsentwickler erstellt Ressourcenreferenzen für verwaltete Verbindungsfactorys im Anwendungsimplementierungsdeskriptor oder mit Annotationen. Jede Ressourcenreferenz enthält eine lokale Referenz für die Verbindungsfactory von eXtreme Scale sowie den Geltungsbereich für die gemeinsame Nutzung von Ressourcen.

Wichtig: Die Aktivierung der gemeinsamen Nutzung von Ressourcen ist wichtig, weil sie die gemeinsame Nutzung lokaler Transaktionen durch mehrere Anwendungskomponenten ermöglicht.

Anwendungen können die Verbindungsfactory in die Java-EE-Anwendungskomponente injizieren, oder sie können sie mit JNDI suchen. Die Verbindungsfactory wird verwendet, um Verbindungshandles zur Clientverbindung von eXtreme Scale abzurufen. Die Clientverbindung von eXtreme Scale wird unabhängig von der Ressourcenadapterverbindung verwaltet und bei erstmaliger Verwendung aufgebaut und dann für alle nachfolgenden Verbindungen wiederverwendet.

Nachdem die Verbindung gefunden wurde, ruft die Anwendung eine Referenz auf die Sitzung von eXtreme Scale ab. Mit der Referenz auf die Sitzung von eXtreme Scale ist die Anwendung in der Lage, alle Client-APIs und Features von eXtreme Scale zu nutzen.

Sie können Transaktionen auf eine der folgenden Arten abgrenzen:

- Verwendung der Transaktionsdemarkationsmethoden von com.ibm.websphere-.objectgrid.Session
- Verwendung der lokalen Transaktionen von javax.resource.cci.LocalTransaction
- Verwendung einer globalen Transaktion, wenn WebSphere Application Server mit aktivierter Unterstützung des letzten Teilnehmers verwendet wird. Wenn Sie diesen Ansatz für die Demarkation wählen, müssen Sie
  - eine anwendungsverwaltete globale Transaktion mit javax.transaction.User-Transaction verwenden,
  - eine containerverwaltete Transaktion verwenden.

Zuständigkeiten des Anwendungsimplementierers

Der Anwendungsimplementierer bindet die lokale Referenz der Verbindungsfactory des Ressourcenadapters, die der Anwendungsentwickler definiert, an die Verbindungsfactorys des Ressourcenadapters, die der Administrator definiert. Der Anwendungsimplementierer muss der Anwendung den richtigen Verbindungsfactorytyp und -geltungsbereich zuweisen und sicherstellen, dass die Verbindungsfactory nicht von mehreren Anwendungen gemeinsam genutzt wird, um die gemeinsame Nutzung von Java-Objekten zu verhindern. Der Anwendungsimplementierer ist auch für die Konfiguration und Zuordnung anderer Konfigurationsinformationen zuständig, die für alle Verbindungsfactorys gleichermaßen gelten.

Attribut "CopyMode"

Java

Sie können die Anzahl der Kopien optimieren, indem Sie das Attribut "CopyMode" der BackingMap- oder ObjectMap-Objekte in der ObjectGrid-XML-Deskriptordatei definieren.

Sie können die Anzahl der Kopien optimieren, indem Sie das Attribut "CopyMode" der BackingMap- bzw. ObjectMap-Objekte definieren. Das Attribut "CopyMode" hat die folgenden gültigen Werte:

- COPY_ON_READ_AND_COMMIT
- COPY_ON_READ
- NO_COPY
- COPY_ON_WRITE
- COPY_TO_BYTES
- COPY_TO_BYTES_RAW

Der Wert COPY_ON_READ_AND_COMMIT ist der Standardwert. Wenn Sie den Wert COPY_ON_READ definieren, wird eine Kopie der ersten empfangenen Daten erstellt, aber es wird keine Kopie zur Festschreibungszeit erstellt. Dieser Modus ist sicher, wenn die Anwendung einen Wert nach dem Festschreiben einer Transaktion nicht mehr ändert. Wenn Sie den Wert NO_COPY definieren, werden die Daten nicht kopiert, was nur bei schreibgeschützten Daten sicher ist. Wenn sich die Daten nicht ändern, ist es nicht erforderlich, sie aus Isolationsgründen zu kopieren.

Gehen Sie bei der Verwendung des Attributwerts NO_COPY für Maps, die aktualisiert werden können, vorsichtig vor. WebSphere eXtreme Scale verwendet die beim ersten Berühren der Daten erstellt Kopie für das Transaktions-Rollback. Da die Änderung nur die Kopie geändert hat, verwirft eXtreme Scale die Kopie. Wenn der Attributwert NO_COPY verwendet wird und die Anwendung den festgeschriebenen Wert ändert, ist ein Rollback nicht möglich. Das Ändern der festgeschriebenen Werte führt zu Problemen bei Indizes, Replikation usw, weil die Indizes und Replikate bei der Festschreibung der Transaktion aktualisiert werden. Wenn Sie festgeschriebene Daten ändern und dann ein Rollback für die Transaktion durchführen (wobei in diesem Fall gar kein Rollback durchgeführt wird), werden die Indizes nicht aktualisiert, und es findet keine Replikation statt. Andere Threads können die nicht festgeschriebenen Änderungen sofort sehen, selbst wenn Sperren gesetzt sind. Verwenden Sie den Attributwert NO_COPY nur für schreibgeschützte Maps oder für Anwendungen, die den entsprechenden Kopiervorgang durchführen, bevor der Wert geändert wird. Wenn Sie den Attributwert NO_COPY verwenden und sich

mit einem Datenintegritätsproblem an die IBM Unterstützungsfunktion wenden, werden Sie aufgefordert, das Problem im Kopiermodus COPY_ON_READ_AND-_COMMIT zu reproduzieren.

Wenn Sie den Wert COPY_TO_BYTES verwenden, werden Werte in der Map in serialisierter Form gespeichert. Beim Lesen dekomprimiert eXtreme Scale den Wert aus der serialisierten Form und speichert beim Festschreiben den Wert in serialisierter Form. Wenn Sie diese Methode verwenden, wird beim Lesen und beim Festschreiben ein Kopiervorgang durchgeführt.

Einschränkung: 8.6+

Wenn Sie optimistisches Sperren mit COPY_TO_BYTES verwenden, können Ausnahmen des Typs "ClassNotFoundException" beim Ausführen allgemeiner Operationen, wie z. B. beim Invalidieren von Cacheeinträgen, auftreten. Diese Ausnahme treten auf, weil der Mechanismus für optimistisches Sperren die Methode "equals(...)" des Cacheobjekts aufrufen muss, damit alle Änderungen erkannt werden, bevor die Transaktion festgeschrieben wird. Zum Aufrufen der Methode "equals(...)" muss der eXtreme-Scale-Server in der Lage sein, das zwischengespeicherte Objekt zu entserialisieren, d. h., eXtreme Scale muss die Objektklasse laden.

Zum Beheben dieser Ausnahmen können Sie die zwischengespeicherten Objektklassen packen, sodass der eXtreme-Scale-Server die Klassen in eigenständigen Umgebungen laden kann. Deshalb müssen Sie die Klassen in den Klassenpfad stellen.

Falls Ihre Umgebung das OSGi-Framework enthält, packen Sie die Klassen in ein Fragment des Bundles `objectgrid.jar`. Wenn Sie eXtreme-Scale-Server im Liberty-Profil ausführen, packen Sie die Klassen als OSGi-Bundle, und exportieren Sie die Java-Pakete für diese Klassen. Installieren Sie das Bundle anschließend, indem Sie es in das Verzeichnis `grids` kopieren.

In WebSphere Application Server packen Sie die Klassen in die Anwendung oder in eine gemeinsam genutzte Bibliothek, auf die die Anwendung zugreifen kann.

Alternativ können Sie angepasste Serialisierungsprogramme verwenden, die die in eXtreme Scale gespeicherten Byte-Arrays vergleichen, damit alle Änderungen erkannt werden.

Der Standardkopiermodus für eine Map kann im BackingMap-Objekt konfiguriert werden. Mit der Methode "ObjectMap.setCopyMode" können Sie den Kopiermodus für Maps auch vor dem Starten einer Transaktion ändern.

Im Folgenden sehen Sie ein Beispiel für ein BackingMap-Snippet aus einer Datei `objectgrid.xml`, das veranschaulicht, wie der Kopiermodus für eine bestimmte BackingMap gesetzt wird. In diesem Beispiel wird davon ausgegangen, dass Sie `cc` als Namespace für `objectgrid/config` verwenden.

```
<cc:backingMap name="RuntimeLifespan" copyMode="NO_COPY"/>
```

Sperrenmanager

Java

Wenn Sie eine Sperrstrategie konfigurieren, wird zur Gewährleistung der Konsistenz von Cacheeinträgen ein Sperrenmanager für die BackingMap erstellt.

Konfiguration des Sperrenmanagers

Wenn Sie die Sperrstrategie PESSIMISTIC oder OPTIMISTIC verwenden, wird ein Sperrenmanager für die BackingMap erstellt. Der Sperrenmanager verwendet eine Hash-Tabelle, um die Einträge zu verfolgen, die von einer oder mehreren Transaktionen gesperrt werden. Wenn die Hash-Tabelle viele Mapeinträge enthält, kann durch die Verwendung weiterer Sperr-Buckets eine bessere Leistung erzielt werden. Das Risiko von Java-Synchronisationskollisionen sinkt mit zunehmender Anzahl an Buckets. Werden zusätzliche Buckets verwendet, sind auch mehr gemeinsame Zugriffe möglich. Die vorherigen Beispiele haben veranschaulicht, wie eine Anwendung die Anzahl der Sperr-Buckets für eine bestimmte BackingMap-Instanz festlegen kann.

Um zu verhindern, dass eine Ausnahme des Typs "java.lang.IllegalStateException" ausgelöst wird, muss die Methode "setNumberOfLockBuckets" vor dem Aufruf der Methode "initialize" bzw. "getSession" in der ObjectGrid-Instanz aufgerufen werden. Der Methodenparameter "setNumberOfLockBuckets" ist ein primitiver Java-Integer, der die Anzahl der zu verwendenden Sperr-Buckets angibt. Die Verwendung einer Primzahl gewährleistet eine gleichmäßige Verteilung der Mapeinträge auf die Sperr-Buckets. Ein guter Ausgangspunkt für das Erzielen der besten Leistung ist, die Anzahl der Sperr-Buckets auf ungefähr zehn Prozent der erwarteten Anzahl an BackingMap-Einträgen zu setzen.

Sperrstrategien

Java

Die folgenden Sperrstrategien sind verfügbar: PESSIMISTIC (pessimistisch), OPTIMISTIC (optimistisch) und NONE (Ohne). Bei der Auswahl einer Sperrstrategie müssen Sie Aspekte wie den Prozentsatz der einzelnen Typen von Operationen, die potenzielle Verwendung eines Loaders usw. berücksichtigen.

Sperren werden über Transaktionen gebunden. Sie können die folgenden Sperreinstellungen angeben:

- **Keine Sperren**: Die Ausführung ohne Sperren ist die schnellste. Wenn Sie schreibgeschützte Daten verwenden, benötigen Sie möglicherweise keine Sperren.
- **Pessimistisches Sperren**: Es werden Sperren für Einträge angefordert, die so lange gehalten werden, bis die Transaktion festgeschrieben wird. Diese Sperrstrategie bietet eine gute Konsistenz, geht aber zu Lasten des Durchsatzes.
- **Optimistisches Sperren**: Erstellt eine Vorher-Kopie jedes Datensatzes, den die Transaktion berührt, und vergleicht diese mit den aktuellen Eintragswerten, wenn die Transaktion festgeschrieben wird. Wenn die Vorher-Kopie und der Eintragswert nicht übereinstimmen, wird die Transaktion rückgängig gemacht. Es werden keine Sperren bis zur Festschreibungszeit gehalten. Diese Sperrstrategie unterstützt einen besseren gemeinsamen Zugriff als die pessimistische Strategie, birgt aber das Risiko von Transaktions-Rollbacks und Speicherkosten für die zusätzliche Kopie des Eintrags.

Legen Sie die Sperrstrategie in der BackingMap fest. Es ist nicht möglich, die Sperrstrategie für jede einzelne Transaktion zu ändern. Im Folgenden sehen Sie ein Beispiel-XML-Snippet, das veranschaulicht, wie der Sperrmodus in einer Map über die XML-Datei festgelegt wird, und in dem davon ausgegangen wird, dass `cc` der Namespace für `objectgrid/config` ist:

```
<cc:backingMap name="RuntimeLifespan" lockStrategy="PESSIMISTIC" />
```

Pessimistisches Sperren

Verwenden Sie die pessimistische Sperrstrategie für Maps mit Lese-/Schreibzugriff, wenn keine anderen Sperrstrategien möglich sind. Wenn eine ObjectGrid-Map für die Verwendung der pessimistischen Sperrstrategie konfiguriert ist, wird eine pessimistische Transaktionssperre für einen Mapeintrag angefordert, wenn eine Transaktion den Eintrag zum ersten Mal aus der BackingMap abruft. Die pessimistische Sperre wird so lange gehalten, bis die Anwendung die Transaktion abschließt. Gewöhnlich wird die pessimistische Sperrstrategie in den folgenden Situationen verwendet:

- Die BackingMap ist mit oder ohne Loader (Ladeprogramm) konfiguriert, und es sind keine Versionsinformationen verfügbar.
- Die BackingMap wird direkt von einer Anwendung verwendet, die Hilfe von eXtreme Scale für die Steuerung des gemeinsamen Zugriffs benötigt.
- Es sind Versionsinformationen verfügbar, aber Aktualisierungstransaktionen für die Einträge in der BackingMap lösen häufig Kollisionen aus, was zu Fehlern bei der optimistische Aktualisierung führt.

Da die pessimistische Sperrstrategie die größten Auswirkungen auf Leistung und Skalierbarkeit hat, sollte diese Strategie nur für Maps mit Lese-/Schreibzugriff verwendet werden, wenn keine anderen Sperrstrategien angewendet werden können, z. B., wenn häufig Fehler bei der optimistischen Aktualisierung auftreten oder wenn die Wiederherstellung nach einem Fehler bei einer optimistischen Aktualisierung nur schwer für eine Anwendung durchzuführen ist.

8.6+ Wenn Sie pessimistisches Sperren verwenden, können Sie die Sperrmethode verwenden, um Daten oder Schlüssel zu sperren, ohne dass Datenwerte zurückgegeben werden. Mit der Sperrmethode können Sie den Schlüssel im Grid oder den Schlüssel sperren und feststellen, ob der Wert im Grid vorhanden ist. In früheren Releases haben Sie die APIs "get" und "getForUpdate" zum Sperren von Schlüsseln im Datengrid verwendet. Wenn Sie die Daten des Clients jedoch nicht benötigen, ist die Leistung vermindet, weil potenziell große Wertobjekt in den Client abgerufen werden. Außerdem hält "containsKey" momentan keine Sperren, und deshalb sind Sie gezwungen, die Methode "get" oder "getForUpdate" zu verwenden, um die entsprechenden Sperren anzufordern, wenn pessimistisches Sperren verwendet wird. Die API "lock" stellt Ihnen jetzt eine containsKey-Semantik für den Fall bereit, dass die Sperre gehalten wird. Sehen Sie sich die folgenden Beispiele an:

- `boolean ObjectMap.lock(Object key, LockMode lockMode);`

  Sperrt den Schlüssel in der Map und gibt "true" zurück, wenn der Schlüssel vorhanden ist, andernfalls "false".

- `List<Boolean> ObjectMap.lockAll(List keys, LockMode lockMode);`

  Sperrt eine Liste mit Schlüsseln in der Map und gibt eine Liste mit true- oder false-Werten zurück. Es wird `true` zurückgegeben, wenn der Schlüssel vorhanden ist, und `false`, wenn der Schlüssel nicht vorhanden ist.

LockMode ist eine Aufzählung mit den möglichen Werten SHARED, UPGRADABLE und EXCLUSIVE, in der Sie die Schlüssel angeben können, die Sie sperren möchten. Sehen Sie sich die folgende Tabelle an, um sich mit den Beziehungen zwischen diesen Sperrmoduswerten und dem Verhalten vorhandener Methoden vertraut zu machen:

Tabelle 6. LockMode-Werte und vorhandene Methodenäquivalente

Sperrmodus	Methodenäquivalent
SHARED	get()

Tabelle 6. LockMode-Werte und vorhandene Methodenäquivalente (Forts.)

Sperrmodus	Methodenäquivalent
UPGRADABLE	getForUpdate()
EXCLUSIVE	getNextKey() und commit()

Sehen Sie sich das folgende Codebeispiel für den Parameter "LockMode" an:

```
session.begin();
map.lock(key, LockMode.UPGRADABLE);
map.upsert();
session.commit()
```

Optimistisches Sperren

Bei der optimistischen Sperrstrategie wird davon ausgegangen, dass zwei gleichzeitig ausgeführte Transaktionen niemals versuchen, denselben Mapeintrag zu aktualisieren. Aufgrund dieser Annahme müssen die Sperren nicht für den gesamte Lebenszyklus der Transaktion gehalten werden, da es unwahrscheinlich ist, dass mehrere Transaktionen einen Mapeintrag gleichzeitig aktualisieren. Die optimistische Sperrstrategie wird gewöhnlich in den folgenden Situationen verwendet:

- Eine BackingMap ist mit oder ohne Loader (Ladeprogramm) konfiguriert, und es sind Versionsinformationen verfügbar.
- Es werden hauptsächlich nur Transaktionen für eine BackingMap ausgeführt, die Leseoperationen durchführen. Einfüge-, Aktualisierungs- und Entfernungsoperationen für Mapeinträge finden in der BackingMap nur selten statt.
- Eine BackingMap wird häufiger eingefügt, aktualisiert oder entfernt, als sie gelesen wird, aber es finden nur selten Kollisionen zwischen den Transaktionen statt, die sich auf denselben Mapeintrag beziehen.

Wie bei der pessimistischen Sperrstrategie bestimmen die Methoden in der Schnittstelle "ObjectMap", wie eXtreme Scale automatisch versucht, eine Sperre für den Mapeintrag anzufordern, auf den zugegriffen wird. Es bestehen jedoch die folgenden Unterschiede zwischen der pessimistischen und der optimistischen Sperrstrategie:

- Wie bei der pessimistischen Sperrstrategie wird bei der optimistischen Sperrstrategie beim Aufruf der Methoden "get" und "getAll" eine S-Sperre angefordert. Beim optimistischen Sperren wird die S-Sperre jedoch nicht bis zum Abschluss der Transaktion gehalten. Stattdessen wird die S-Sperre freigegeben, bevor die Methode zur Anwendung zurückkehrt. Die Anforderung der Sperre hat den Zweck, dass eXtreme Scale sicherstellen kann, dass nur festgeschriebene Daten von anderen Transaktionen für die aktuelle Transaktion sichtbar sind. Nachdem eXtreme Scale sichergestellt hat, dass die Daten festgeschrieben wurden, wird die S-Sperre freigegeben. Während der Festschreibung wird eine optimistische Versionsprüfung durchgeführt, um sicherzustellen, dass der Mapeintrag nicht von einer anderen Transaktion geändert wurde, nachdem die aktuelle Transaktion die S-Sperre freigegeben hat. Wenn ein Eintrag nicht aus der Map abgerufen wird, bevor er aktualisiert, ungültig gemacht oder gelöscht wird, ruft die Laufzeitumgebung von eXtreme Scale den Eintrag implizit aus der Map ab. Diese implizite get-Operation wird durchgeführt, um den aktuellen Wert abzurufen, den der Eintrag zum Zeitpunkt der Änderungsanforderung hatte.
- Anders als bei der pessimistischen Sperrstrategie werden die Methoden "getForUpdate" und "getAllForUpdate" bei der optimistischen Sperrstrategie genauso wie die Methoden "get" und "getAll" behandelt, d. h., beim Starten der Methode wird eine S-Sperre angefordert, die dann freigegeben wird, bevor die Methode zur Anwendung zurückkehrt.

Alle anderen ObjectMap-Methoden werden genauso behandelt, wie es bei der pessimistischen Sperrstrategie der Fall ist, d. h., wenn die Methode "commit" aufgerufen wird, wird eine X-Sperre für jeden Mapeintrag angefordert, der eingefügt, aktualisiert, entfernt, angerührt oder ungültig gemacht wurde, und diese X-Sperre wird so lange gehalten, bis die Commit-Verarbeitung der Transaktion abgeschlossen ist.

Bei der optimistischen Sperrstrategie wird davon ausgegangen, dass gleichzeitig ausgeführte Transaktionen nicht versuchen, denselben Mapeintrag zu aktualisieren. Aufgrund dieser Annahme müssen die Sperren nicht für die gesamte Lebensdauer der Transaktion gehalten werden, da es unwahrscheinlich ist, dass mehrere Transaktionen einen Mapeintrag gleichzeitig aktualisieren. Da eine Sperre jedoch nicht gehalten wird, ist es potenziell möglich, dass eine andere gleichzeitig ausgeführte Transaktion den Mapeintrag ändert, nachdem die aktuelle Transaktion ihre S-Sperre freigegeben hat.

Zur Behandlung dieser Möglichkeit ruft eXtreme Scale während der Festschreibung eine X-Sperre ab und führt eine optimistische Versionsprüfung durch, um sicherzustellen, dass der Mapeintrag nicht von einer anderen Transaktion geändert wurde, nachdem die aktuelle Transaktion den Mapeintrag aus der BackingMap gelesen hat. Wenn der Mapeintrag von einer anderen Transaktion geändert wurde, fällt die Versionsprüfung negativ aus, und es wird eine Ausnahme des Typs "OptimisticCollisionException" ausgegeben. Diese Ausnahme erzwingt ein Rollback der aktuellen Transaktion, und die Anwendung muss die vollständige Transaktion wiederholen. Die optimistische Sperrstrategie ist sehr hilfreich, wenn eine Map hauptsächlich nur gelesen wird und es unwahrscheinlich ist, dass gleichzeitige Aktualisierungen an demselben Mapeintrag vorgenommen werden.

Einschränkung: 8.6+

Wenn Sie optimistisches Sperren mit COPY_TO_BYTES verwenden, können Ausnahmen des Typs "ClassNotFoundException" beim Ausführen allgemeiner Operationen, wie z. B. beim Invalidieren von Cacheeinträgen, auftreten. Diese Ausnahme treten auf, weil der Mechanismus für optimistisches Sperren die Methode "equals(...)" des Cacheobjekts aufrufen muss, damit alle Änderungen erkannt werden, bevor die Transaktion festgeschrieben wird. Zum Aufrufen der Methode "equals(...)" muss der eXtreme-Scale-Server in der Lage sein, das zwischengespeicherte Objekt zu entserialisieren, d. h., eXtreme Scale muss die Objektklasse laden.

Zum Beheben dieser Ausnahmen können Sie die zwischengespeicherten Objektklassen packen, sodass der eXtreme-Scale-Server die Klassen in eigenständigen Umgebungen laden kann. Deshalb müssen Sie die Klassen in den Klassenpfad stellen.

Falls Ihre Umgebung das OSGi-Framework enthält, packen Sie die Klassen in ein Fragment des Bundles `objectgrid.jar`. Wenn Sie eXtreme-Scale-Server im Liberty-Profil ausführen, packen Sie die Klassen als OSGi-Bundle, und exportieren Sie die Java-Pakete für diese Klassen. Installieren Sie das Bundle anschließend, indem Sie es in das Verzeichnis `grids` kopieren.

In WebSphere Application Server packen Sie die Klassen in die Anwendung oder in eine gemeinsam genutzte Bibliothek, auf die die Anwendung zugreifen kann.

Alternativ können Sie angepasste Serialisierungsprogramme verwenden, die die in eXtreme Scale gespeicherten Byte-Arrays vergleichen, damit alle Änderungen erkannt werden.

Ohne Sperren

Wenn eine BackingMap ohne Sperrstrategie konfiguriert ist, werden keine Transaktionssperren für einen Mapeintrag angefordert.

Anmerkung: 8.6+ BackingMaps, für die keine Sperrstrategie konfiguriert ist, können nicht an einer Transaktion teilnehmen, an der mehrere Partitionen beteiligt sind.

Dies ist hilfreich, wenn eine Anwendung ein Persistenzmanager ist, wie z. B. ein EJB-Container, oder wenn eine Anwendung Hibernate für das Abrufen persistenter Daten verwendet. In diesem Szenario ist die BackingMap ohne Loader konfiguriert, und der Persistenzmanager verwendet die BackingMap als Datencache. Der Persistenzmanager übernimmt die Steuerung des gemeinsamen Zugriffs für Transaktionen, die auf dieselben Mapeinträge zugreifen.

Für die Steuerung des gemeinsamen Zugriffs muss WebSphere eXtreme Scale keine Transaktionssperren anfordern. In dieser Situation wird davon ausgegangen, dass der Persistenzmanager seine Transaktionssperren nicht freigibt, bevor die ObjectGrid-Map mit festgeschriebenen Änderungen aktualisiert wurde. Wenn der Persistenzmanager seine Sperren freigibt, muss eine pessimistische oder optimistische Sperrstrategie verwendet werden. Angenommen, der Persistenzmanager eines EJB-Containers aktualisiert eine ObjectGrid-Map mit Daten, die in der vom EJB-Container verwalteten Transaktion festgeschrieben wurden. Wenn die Aktualisierung der ObjectGrid-Map stattfindet, bevor der Persistenzmanager seine Transaktionssperren freigibt, können Sie die Strategie ohne Sperren verwenden. Wenn die Aktualisierung der ObjectGrid-Map stattfindet, nachdem der Persistenzmanager seine Transaktionssperren freigibt, müssen Sie die optimistische oder die pessimistische Sperrstrategie verwenden.

Ein weiteres Szenario, in dem die Strategie ohne Sperren verwendet werden kann, ist das, wenn die Anwendung eine BackingMap direkt verwendet und ein Loader für die Map konfiguriert ist. In diesem Szenario verwendet der Loader die Unterstützung für die Steuerung des gemeinsamen Zugriffs, die von einem Verwaltungssystem für relationale Datenbanken bereitgestellt wird, indem er entweder Java Database Connectivity (JDBC) oder Hibernate für den Zugriff auf die Daten in einer relationalen Datenbank verwendet. Die Loader-Implementierung kann einen optimistischen oder pessimistischen Ansatz verwenden. Mit einem Loader, der einen optimistischen Sperr- oder Versionssteuerungsansatz verwendet, kann die höchste Anzahl gemeinsamer Zugriffe und die höchste Leistung erzielt werden. Weitere Informationen zur Implementierung eines optimistischen Sperransatzes finden Sie im Abschnitt "OptimisticCallback" in den Hinweisen zu Loadern in der Veröffentlichung *Verwaltung*. Wenn Sie einen Loader verwenden, der die Unterstützung für pessimistisches Sperren eines zugrunde liegenden Back-Ends verwendet, können Sie den Parameter "forUpdate" verwenden, der an die Methode "get" der Schnittstelle "Loader" übergeben wird. Setzen Sie diesen Parameter auf "true", wenn die Methode "getForUpdate der Schnittstelle "ObjectMap von der Anwendung zum Abrufen der Daten verwendet wird. Der Loader kann diesen Parameter verwenden, um festzustellen, ob eine aktualisierbare Sperre für die Zeile, die gelesen wird, angefordert werden muss. DB2 fordert beispielsweise eine aktualisierbare Sperre an, wenn eine SQL-Anweisung SELECT eine Klausel `FOR UPDATE` enthält. Dieser Ansatz bietet dieselben Präventionsmechanismen für Deadlocks, die im Abschnitt „Pessimistisches Sperren” auf Seite 136 beschrieben sind.

Weitere Informationen finden Sie im Abschnitt zur Behandlung von Sperren in der Veröffentlichung *Programmierung* bzw. zum Sperren von Mapeinträgen in der Veröffentlichung *Verwaltung*.

Transaktionen verteilen

Java

Verwenden Sie Java Message Service (JMS) für die Verteilung von Transaktionsänderungen zwischen verschiedenen Schichten und in Umgebungen auf heterogenen Plattformen.

JMS ist ein ideales Protokoll für die Verteilung von Änderungen zwischen verschiedenen Schichten und in Umgebungen auf heterogenen Plattformen. Einige Anwendungen, die eXtreme Scale verwenden, können beispielsweise in IBM WebSphere Application Server Community Edition, Apache Geronimo oder Apache Tomcat implementiert sein, wohingegen andere Anwendungen in WebSphere Application Server Version 6.x ausgeführt werden. JMS eignet sich optimal für die Verteilung von Änderungen zwischen eXtreme-Scale-Peers in diesen unterschiedlichen Umgebungen. Der Nachrichtentransport des High Availability Manager ist sehr schnell, kann aber nur Änderungen an Java Virtual Machines verteilen, die sich in derselben Stammgruppe befinden. JMS ist zwar langsamer, unterstützt aber die gemeinsame Nutzung eines ObjectGrids durch größere und unterschiedlichere Gruppen von Anwendungsclients. JMS ist ideal, wenn Daten in einem ObjectGrid von einem Swing-Fat-Client und einer in WebSphere Extended Deployment implementierten Anwendung gemeinsam genutzt werden.

Der integrierte Mechanismus für die Inaktivierung von Clients und die Peer-to-Peer-Replikation sind Beispiele für JMS-basierte Verteilung von Transaktionsänderungen. Weitere Einzelheiten finden Sie in den Informationen zum Konfigurieren der Peer-to-Peer-Replikation mit JMS in der Veröffentlichung *Verwaltung*.

JMS implementieren

JMS wird für die Verteilung von Transaktionsänderungen über ein Java-Objekt implementiert, das sich wie ein ObjectGridEventListener verhält. Dieses Objekt kann den Status auf die folgenden vier Arten weitergeben:

1. Invalidate: (Invalidieren) Jeder Eintrag, der entfernt, aktualisiert oder gelöscht wird, wird in allen Peer-JVMs entfernt, wenn diese die Nachricht empfangen.
2. Invalidate conditional: (Bedingtes Invalidieren) Der Eintrag wird nur entfernt, wenn die lokale Version kleiner-gleich der Version im Veröffentlichungskomponente (Publisher) ist.
3. Push: (Übertragung mit Push) Jeder Eintrag, der entfernt, aktualisiert, gelöscht oder eingefügt wurde, wird in allen Peer-JVMs hinzugefügt bzw. überschrieben, wenn diese die JMS-Nachricht empfangen.
4. Push conditional: (Bedingte Übertragung mit Push) Der Eintrag wird auf Empfangsseite nur dann aktualisiert bzw. hinzugefügt, wenn der lokale Eintrag älter ist als die Version, die veröffentlicht wird.

Auf zu veröffentlichende Änderungen warten

Das Plug-in implementiert die Schnittstelle "ObjectGridEventListener", um das Ereignis "transactionEnd" abzufangen. Wenn eXtreme Scale diese Methode aufruft, versucht das Plug-in die LogSequence-Liste für jede Map, die von der Transaktion angerührt wurde, in eine JMS-Nachricht zu konvertieren und anschließend zu veröffentlichen. Das Plug-in kann so konfiguriert werden, dass Änderungen für alle

Maps oder einen Teil der Maps veröffentlicht werden. LogSequence-Objekte werden für die Maps verarbeitet, für die die Veröffentlichung aktiviert ist. Die ObjectGrid-Klasse "LogSequenceTransformer" serialisiert eine gefilterte LogSequence für jede Map in einen Datenstrom. Nachdem alle LogSequences in den Datenstrom serialisiert wurden, wird eine JMS-ObjectMessage erstellt und unter einem bekannten Topic veröffentlicht.

Auf JMS-Nachrichten warten und sie auf das lokale ObjectGrid anwenden

Dasselbe Plug-in startet auch einen Thread, der in einer Schleife ausgeführt wird und alle Nachrichten empfängt, die unter dem bekannten Topic veröffentlicht werden. Wenn eine Nachricht ankommt, wird der Nachrichteninhalt an die Klasse "LogSequenceTransformer" übergeben, wo sie in eine Gruppe von LogSequence-Objekten konvertiert wird. Anschließend wird eine Transaktion ohne Durchschreiben (no-write-through) gestartet. Jedes LogSequence-Objekt wird an die Methode "Session.processLogSequence" übergeben, die die lokalen Maps mit den Änderungen aktualisiert. Die Methode "processLogSequence" erkennt den Verteilungsmodus. Die Transaktion wird festgeschrieben, und der lokale Cache enthält die Änderungen. Weitere Einzelheiten zur Verwendung von JMS für die Verteilung von Transaktionsänderungen finden Sie in den Informationen zur Verteilung von Änderungen zwischen Peer-JVMs (Java Virtual Machines) in der Veröffentlichung *Verwaltung*.

Einzelpartitionstransaktionen und datengridübergreifende Partitionstransaktionen

Java

Der Hauptunterschied zwischen WebSphere eXtreme Scale und traditionellen Datenspeicherlösungen wie relationalen oder speicherinternen Datenbanken ist die Verwendung der Partitionierung, die eine lineare Skalierung des Caches ermöglicht. Die wichtigen Transaktionstypen, die berücksichtigt werden müssen, sind Einzelpartitionstransaktionen und datengridübergreifende Partitionstransaktionen.

Im Allgemeinen können Interaktionen mit dem Cache, wie im folgenden Abschnitt beschrieben, in die Kategorien "Einzelpartitionstransaktionen" und "Datengridübergreifende Partitionstransaktionen" eingeteilt werden.

Einzelpartitionstransaktionen

Einzelpartitionstransaktionen sind die vorzuziehende Methode für die Interaktion mit Caches in WebSphere eXtreme Scale. Wenn eine Transaktion auf eine Einzelpartition beschränkt ist, ist sie standardmäßig auf eine einzelne Java Virtual Machine und damit auf einen einzelnen Servercomputer beschränkt. Ein Server kann *M* dieser Transaktionen pro Sekunde ausführen, und wenn Sie *N* Computer haben, sind `M*N` Transaktionen pro Sekunde möglich. Wenn sich Ihr Geschäft erweitert und Sie doppelt so viele dieser Transaktionen pro Sekunde ausführen müssen, können Sie *N* verdoppeln, indem Sie weitere Computer kaufen. Auf diese Weise können Sie Kapazitätsanforderungen erfüllen, ohne die Anwendung zu ändern, Hardware zu aktualisieren oder die Anwendung außer Betrieb zu nehmen.

Zusätzlich zu der Möglichkeit, den Cache so signifikant skalieren zu können, maximieren Einzelpartitionstransaktionen auch die Verfügbarkeit des Caches. Jede Transaktion ist nur von einem einzigen Computer abhängig. Jeder der anderen `(N-1)` Computer kann ausfallen, ohne den Erfolg oder die Antwortzeit der Transaktion zu beeinflussen. Wenn Sie also mit 100 Computern arbeiten und einer dieser

Computer ausfällt, wird nur 1 Prozent der Transaktionen, die zum Zeitpunkt des Ausfalls dieses Servers unvollständig sind, rückgängig gemacht. Nach dem Ausfall des Servers verlagert WebSphere eXtreme Scale die Partitionen des ausgefallenen Servers auf die anderen 99 Computer. In diesem kurzen Zeitraum vor der Durchführung der Operation können die anderen 99 Computer weiterhin Transaktionen ausführen. Nur die Transaktionen, an denen die Partitionen beteiligt sind, die umgelagert werden, sind blockiert. Nach Abschluss des Failover-Prozesses ist der Cache mit 99 Prozent seiner ursprünglichen Durchsatzkapazität wieder vollständig betriebsbereit. Nachdem der ausgefallene Server ersetzt und der Ersatzserver dem Datengrid hinzugefügt wurde, kehrt der Cache zu einer Durchsatzkapazität von 100 Prozent zurück.

Datengridübergreifende Transaktionen

Was Leistung, Verfügbarkeit und Skalierbarkeit betrifft, sind datengridübergreifende Transaktionen das Gegenteil von Einzelpartitionstransaktionen. Datengridübergreifende Transaktionen greifen auf jede Partition und damit auf jeden Computer in der Konfiguration zu. Jeder Computer im Datengrid wird aufgefordert, einige Daten zu suchen und anschließend das Ergebnis zurückzugeben. Die Transaktion kann erst abgeschlossen werden, nachdem jeder Computer geantwortet hat, und deshalb wird der Durchsatz des gesamten Datengrids durch den langsamsten Computer beschränkt. Das Hinzufügen von Computern macht den langsamsten Computer nicht schneller und verbessert damit auch nicht den Durchsatz des Caches.

Datengridübergreifende Transaktionen haben einen ähnlichen Effekt auf die Verfügbarkeit. Wenn Sie mit 100 Servern arbeiten und ein Server ausfällt, werden 100 Prozent der Transaktionen, die zum Zeitpunkt des Serverausfalls in Bearbeitung sind, rückgängig gemacht. Nach dem Ausfall des Servers beginnt WebSphere eXtreme Scale mit der Verlagerung der Partitionen des ausgefallenen Servers auf die anderen 99 Computer. In dieser Zeit, d. h. bis zum Abschluss des Failover-Prozesses, kann das Datengrid keine dieser Transaktionen verarbeiten. Nach Abschluss des Failover-Prozesses ist der Cache wieder betriebsbereit, aber mit verringerter Kapazität. Wenn jeder Computer im Datengrid 10 Partitionen bereitstellt, erhalten 10 der verbleibenden 99 Computer während des Failover-Prozesses mindestens eine zusätzliche Partition. Eine zusätzliche Partition erhöht die Arbeitslast dieses Computers um mindestens 10 Prozent. Da der Durchsatz des Datengrids in einer datengridübergreifenden Transaktion auf den Durchsatz des langsamsten Computers beschränkt ist, reduziert sich der Durchsatz durchschnittlich um 10 Prozent.

Einzelpartitionstransaktionen sind im Hinblick auf die horizontale Vorwärtsskalierung (Scale-out) mit einem verteilten, hoch verfügbaren Objektcache wie WebSphere eXtreme Scale den datengridübergreifenden Transaktionen vorzuziehen. Die Maximierung der Leistung solcher Systemtypen erfordert die Verwendung von Techniken, die sich von den traditionellen relationalen Verfahren unterscheiden, aber Sie datengridübergreifende Transaktionen in skalierbare Einzelpartitionstransaktionen konvertieren.

Bewährte Verfahren für die Erstellung skalierbarer Datenmodelle

Die bewährten Verfahren für die Erstellung skalierbarer Anwendungen mit Produkten wie WebSphere eXtreme Scale sind in zwei Kategorien einteilbar: Grundsätze und Implementierungstipps. Grundsätze sind Kernideen, die im Design der Daten selbst erfasst werden müssen. Es ist sehr unwahrscheinlich, dass sich eine Anwendung, die diese Grundsätze nicht einhält, problemlos skalieren lässt, selbst für ihre Haupttransaktionen. Implementierungstipps werden für problematische

Transaktionen in einer ansonsten gut entworfenen Anwendung angewendet, die sich an die allgemeinen Grundsätze für skalierbare Datenmodelle hält.

Grundsätze

Einige wichtige Hilfsmittel für die Optimierung der Skalierbarkeit sind Basiskonzepte oder Grundsätze, die beachtet werden müssen.

Duplizieren an Stelle von Normalisieren

Der wichtigste Punkt, der bei Produkten wie WebSphere eXtreme Scale zu beachten ist, ist der, dass sie für die Verteilung von Daten auf sehr viele Computer konzipiert sind. Wenn das Ziel darin besteht, die meisten oder sogar alle Transaktionen auf einer einzelnen Partition auszuführen, muss das Datenmodelldesign sicherstellen, dass sich alle Daten, die die Transaktion unter Umständen benötigt, auf der Partition befinden. In den meisten Fällen kann dies nur durch Duplizierung der Daten erreicht werden.

Stellen Sie sich beispielsweise eine Anwendung wie ein Nachrichtenbrett vor. Zwei sehr wichtige Transaktionen für ein Nachrichtenbrett zeigen alle Veröffentlichungen eines bestimmten Benutzers und alle Veröffentlichungen unter einem bestimmten Topic an. Stellen Sie sich zunächst vor, wie diese Transaktionen mit einem normalisierten Datenmodell arbeiten, das einen Benutzerdatensatz, einen Topic-Datensatz und einen Veröffentlichungsdatensatz mit dem eigentlichen Text enthält. Wenn Veröffentlichungen mit Benutzerdatensätzen partitioniert werden, wird aus der Anzeige des Topics eine gridübergreifende Transaktion und umgekehrt. Topics und Benutzer können nicht gemeinsam partitioniert werden, da sie eine Viele-zu-viele-Beziehung haben.

Die beste Methode für die Skalierung dieses Nachrichtenbretts ist die Duplizierung der Veröffentlichungen, wobei eine Kopie mit dem Topic-Datensatz und eine Kopie mit dem Benutzerdatensatz gespeichert wird. Die anschließende Anzeige der Veröffentlichungen eines Benutzers ist eine Einzelpartitionstransaktion, die Anzeige der Veröffentlichungen unter einem Topic ist eine Einzelpartitionstransaktion, und die Aktualisierung oder das Löschen einer Veröffentlichung ist eine Transaktion, an der zwei Partitionen beteiligt sind. Alle drei Transaktionen können linear skaliert werden, wenn die Anzahl der Computer im Datengrid zunimmt.

Skalierbarkeit an Stelle von Ressourcen

Die größte Hindernis, das beim Einsatz denormalisierter Datenmodell überwunden werden muss, sind die Auswirkungen, die diese Modell auf Ressourcen haben. Die Verwaltung von zwei, drei oder mehr Kopien derselben Daten kann den Anschein erwecken, dass zu viele Ressourcen benötigt werden, als dass dieser Ansatz praktikabel ist. Wenn Sie mit diesem Szenario konfrontiert werden, berücksichtigen Sie die folgenden Fakten: Hardwareressourcen werden von Jahr zu Jahr billiger. Zweitens, und noch wichtiger, mit WebSphere eXtreme Scale fallen die meisten verborgenen Kosten weg, die bei der Implementierung weiterer Ressourcen anfallen.

Messen Sie Ressourcen anhand der Kosten und nicht anhand von Computerbegriffen wie Megabyte oder Prozessoren. Datenspeicher, die mit normalisierten relationalen Daten abreiten, müssen sich im Allgemeinen auf demselben Computer befinden. Diese erforderliche Co-Location bedeutet, dass ein einzelner größerer Unternehmenscomputer an Stelle mehrerer kleinerer Computer erworben werden muss. Bei Unternehmenshardware ist es nicht unüblich, dass ein einziger Computer, der in der Lage ist, eine Million

Transaktionen pro Sekunde zu verarbeiten, mehr kostet als 10 Computer zusammen, die in der Lage sind, jeweils 100.000 Transaktionen pro Sekunden auszuführen.

Außerdem fallen Geschäftskosten für die Implementierung der Ressourcen an. Irgendwann reicht die Kapazität in einem expandierenden Unternehmen einfach nicht mehr aus. In diesem Fall setzen Sie den Betrieb entweder aus, während Sie die Umstellung auf einen größeren und schnelleren Computer durchführen, oder Sie erstellen eine zweite Produktionsumgebung, auf die Sie den Betrieb dann umstellen können. In beiden Fällen fallen zusätzliche Kosten durch das ausgefallene Geschäft oder durch die Verwaltung der doppelten Kapazität in der Übergangsphase an.

Mit WebSphere eXtreme Scale muss die Anwendung nicht heruntergefahren werden, um Kapazität hinzuzufügen. Wenn die Prognose für Ihr Geschäft lautet, dass Sie 10 Prozent mehr Kapazität für das kommende Jahr benötigen, erhöhen Sie die Anzahl der Computer im Datengrid um 10 Prozent. Sie können diese Erweiterung ohne Anwendungsausfallzeit und ohne den Einkauf von Kapazitäten durchführen, die Sie hinterher nicht mehr benötigen.

Datenkonvertierungen vermeiden

Wenn Sie WebSphere eXtreme Scale verwenden, müssen Daten in einem Format gespeichert werden, das von der Geschäftslogik direkt konsumiert werden kann. Die Aufteilung der Daten in ein primitiveres Format ist kostenintensiv. Die Konvertierung muss durchgeführt werden, wenn die Daten geschrieben und wenn die Daten gelesen werden. Mit relationalen Datenbanken ist diese Konvertierung unumgänglich, weil die Daten letztendlich relativ häufig auf der Platte gespeichert werden, aber mit WebSphere eXtreme Scale fallen diese Konvertierungen weg. Der größte Teil der Daten wird im Hauptspeicher gespeichert und kann deshalb in genau dem Format gespeichert werden, das die Anwendung erfordert.

Durch die Einhaltung dieser einfachen Regel können Sie Ihre Daten dem ersten Grundsatz entsprechend denormalisieren. Der gängigste Konvertierungstyp für Geschäftsdaten ist die JOIN-Operation, die erforderlich ist, um normalisierte Daten in eine Ergebnismenge zu konvertieren, die den Anforderungen der Anwendung entspricht. Durch die implizite Speicherung der Daten im richtigen Format werden diese JOIN-Operationen vermieden, und es entsteht ein denormalisiertes Datenmodell.

Unbegrenzte Abfragen vermeiden

Unbegrenzte Abfragen lassen sich nicht gut skalieren, egal, wie gut Sie Ihre Daten auch strukturieren. Verwenden Sie beispielsweise keine Transaktion, die eine Liste aller Einträge nach Wert sortiert abfragt. Diese Transaktion funktioniert möglicherweise, wenn die Gesamtanzahl der Einträge bei 1000 liegt, aber wenn die die Gesamtanzahl der Einträge 10 Million erreicht, gibt die Transaktion alle 10 Millionen Einträge zurück. Wenn Sie diese Transaktion ausführen, sind zwei Ergebnisse am wahrscheinlichsten: Die Transaktion überschreitet das zulässige Zeitlimit, oder im Client tritt eine abnormale Speicherbedingung auf.

Die beste Option ist, die Geschäftslogik so zu ändern, dass nur die Top 10 oder 20 Einträge zurückgegeben werden können. Durch diese Änderung der Logik bleibt die Größe der Transaktion verwaltbar, unabhängig davon, wie viele Einträge im Cache enthalten sind.

Schema definieren

Der Hauptvorteil der Normalisierung von Daten ist der, dass sich das Datenbanksystem im Hintergrund um die Datenkonsistenz kümmern kann. Wenn Daten für Skalierbarkeit denormalisiert werden, ist diese automatische Verwaltung der Datenkonsistenz nicht mehr möglich. Sie müssen ein Datenmodell implementieren, das auf der Anwendungsebene oder als Plug-in für das verteilte Datengrid arbeiten kann, um die Datenkonsistenz zu gewährleisten.

Stellen Sie sich das Beispiel mit dem Nachrichtenbrett vor. Wenn eine Transaktion eine Veröffentlichung aus einem Topic entfernt, muss das Veröffentlichungsduplikat im Benutzerdatensatz entfernt werden. Ohne ein Datenmodell ist es möglich, dass ein Entwickler den Anwendungscode zum Entfernen der Veröffentlichung aus dem Topic schreibt und vergisst, die Veröffentlichung aus dem Benutzerdatensatz zu entfernen. Wenn der Entwickler jedoch ein Datenmodell verwendet, anstatt direkt mit dem Cache zu interagieren, kann die Methode "removePost" im Datenmodell die Benutzer-ID aus der Veröffentlichung extrahieren, den Benutzerdatensatz suchen und das Veröffentlichungsduplikat im Hintergrund entfernen.

Alternativ können Sie einen Listener implementieren, der auf der tatsächlichen Partition ausgeführt wird, das Topic überwacht und bei einer Änderung des Topics Benutzerdatensatz automatisch anpasst. Ein Listener kann von Vorteil sein, weil die Anpassung am Benutzerdatensatz lokal vorgenommen werden kann, wenn die Partition den Benutzerdatensatz enthält. Selbst wenn sich der Benutzerdatensatz auf einer anderen Partition befindet, findet die Transaktion zwischen Servern und nicht zwischen dem Client und dem Server statt. Die Netzverbindung zwischen Servern ist wahrscheinlich schneller als die Netzverbindung zwischen dem Client und dem Server.

Konkurrenzsituationen vermeiden

Szenarien wie die Verwendung eines globalen Zählers vermeiden. Das Datengrid kann nicht skaliert werden, wenn ein einzelner Datensatz im Vergleich mit den restlichen Datensätzen unverhältnismäßig oft verwendet wird. Die Leistung des Datengrids wird durch die Leistung des Computers beschränkt, der diesen Datensatz enthält.

Versuchen Sie in solchen Situationen, den Datensatz aufzuteilen, sodass er pro Partition verwaltet wird. Stellen Sie sich beispielsweise eine Transaktion vor, die die Gesamtanzahl der Einträge im verteilten Cache zurückgibt. Anstatt jede Einfüge- und Entfernungsoperation auf einen einzelnen Datensatz zugreifen zu lassen, dessen Zähler sich erhöht, können Sie einen Listener auf jeder Partition einsetzen, der die Einfüge- und Entfernungsoperation verfolgt. Mit dieser Listenerverfolgung können aus Einfüge- und Entfernungsoperationen Einzelpartitionsoperationen werden.

Das Lesen des Zählers wird zu einer datengridübergreifenden Operation, aber die Leseoperation war bereits vorher genauso ineffizient wie eine datengridübergreifende Operation, weil ihre Leistung an die Leistung des Computers gebunden war, auf dem sich der Datensatz befindet.

Implementierungstipps

Zum Erreichen der besten Skalierbarkeit können Sie außerdem die folgenden Tipps beachten.

Umgekehrte Suchindizes verwenden

Stellen Sie sich ein ordnungsgemäß denormalisiertes Datenmodell vor, in dem Kundendatensätze auf der Basis der Kunden-ID partitioniert werden. Diese Partitionierungsmethode ist die logische Option, weil nahezu jede Geschäftsoperation, die mit dem Kundendatensatz ausgeführt wird, die Kunden-ID verwendet. Eine wichtige Transaktion, in der die Kunden-ID jedoch nicht verwendet wird, ist die Anmeldetransaktion. Es ist üblich, dass Benutzernamen oder E-Mail-Adressen für die Anmeldung verwendet werden, und keine Kunden-IDs.

Der einfache Ansatz für das Anmeldeszenario ist die Verwendung einer datengridübergreifenden Transaktion, um den Kundendatensatz zu suchen. Wie zuvor erläutert, ist dieser Ansatz nicht skalierbar.

Die nächste Option ist die Partitionierung nach Benutzernamen oder E-Mail-Adressen. Diese Option ist nicht praktikabel, da alle Operationen, die auf der Kunden-ID basieren, zu datengridübergreifenden Transaktionen werden. Außerdem möchten die Kunden auf Ihrer Site möglicherweise ihren Benutzernamen oder ihre E-Mail-Adresse ändern. Produkte wie WebSphere eXtreme Scale benötigen den Wert, der für die Partitionierung der Daten verwendet wird, um konstant zu bleiben.

Die richtige Lösung ist die Verwendung eines umgekehrten Suchindex. Mit WebSphere eXtreme Scale kann ein Cache in demselben verteilten Grid wie der Cache erstellt werden, der alle Benutzerdatensätze enthält. Dieser Cache ist hoch verfügbar, partitioniert und skalierbar. Dieser Cache kann verwendet werden, um einen Benutzernamen oder eine E-Mail-Adresse einer Kunden-ID zuzuordnen. Dieser Cache verwandelt die Anmeldung in eine Operation, an der zwei Partitionen beteiligt sind, und nicht in eine gridübergreifende Transaktion. Dieses Szenario ist zwar nicht so effektiv wie eine Einzelpartitionstransaktion, aber der Durchsatz nimmt linear mit steigender Anzahl an Computern zu.

Berechnung beim Schreiben

Die Generierung häufig berechneter Werte wie Durchschnittswerte oder Summen kann kostenintensiv sein, weil bei diesen Operationen gewöhnlich sehr viele Einträge gelesen werden müssen. Da in den meisten Anwendungen mehr Leseoperationen als Schreiboperationen ausgeführt werden, ist es effizient, diese Werte beim Schreiben zu berechnen und das Ergebnis anschließend im Cache zu speichern. Durch dieses Verfahren werden Leseoperationen schneller und skalierbarer.

Optionale Felder

Stellen Sie sich einen Benutzerdatensatz vor, der eine geschäftliche Telefonnummer, eine private Telefonnummer und eine Handy-Nummer enthält. Ein Benutzer kann alle, keine oder eine beliebige Kombination dieser Nummern haben. Wenn die Daten normalisiert sind, sind eine Benutzertabelle und eine Telefonnummerntabelle vorhanden. Die Telefonnummern für einen bestimmten Benutzer können über eine JOIN-Operation zwischen den beiden Tabellen ermittelt werden.

Die Denormalisierung dieses Datensatzes erfordert keine Datenduplizierung, weil die meisten Benutzer nicht dieselben Telefonnummern haben. Stattdessen müssen freie Bereiche im Benutzerdatensatz zulässig sein. Anstatt eine Telefonnummerntabelle zu verwenden, können Sie jedem Benutzerdatensatz drei Attribute hinzufügen, eines für jeden Telefonnummerntyp. Durch das Hinzufügen dieser Attribut wird die JOIN-Operation vermieden, und die Suche der Telefonnummern für einen Benutzer wird zu einer Einzelpartitionsoperation.

Verteilung von Viele-zu-viele-Beziehungen

Stellen Sie sich eine Anwendung, die Produkte und die Länden verfolgt, in denen die Produkte verkauft werden. Ein Produkt wird in vielen Läden verkauft, und ein Laden verkauft viele Produkte. Angenommen, diese Anwendung verfolgt 50 große Einzelhändler. Jedes Produkt wird in maximal 50 Länden verkauft, wobei jeder Laden Tausende von Produkten verkauft.

Verwalten Sie eine Liste der Läden in der Produktentität (Anordnung A), anstatt eine Liste von Produkten in jeder Ladenentität zu verwalten (Anordnung B). Wenn Sie sich einige der Transaktionen ansehen, die diese Anwendung ausführen muss, ist leicht zu erkennen, warum Anordnung A skalierbarer ist.

Sehen Sie sich zuerst die Aktualisierungen an. Wenn bei Anordnung A ein Produkt aus dem Bestand eines Ladens entfernt wird, wird die Produktentität gesperrt. Enthält das Datengrid 10000 Produkte, muss nur 1/10000 des Grids gesperrt werden, um die Aktualisierung durchzuführen. Bei Anordnung B enthält das Datengrid nur 50 Länden, sodass 1/50 des Grids gesperrt werden muss, um die Aktualisierung durchzuführen. Obwohl beide Fälle als Einzelpartitionsoperationen eingestuft werden können, lässt sich Anordnung A effizienter skalieren.

Sehen Sie sich jetzt die Leseoperationen für Anordnung A an. Das Durchsuchen eines Ladens, in dem ein Produkt verkauft wird, ist eine Einzelpartitionsoperation, die skalierbar und schnell ist, weil die Transaktion nur einen kleine Datenmenge überträgt. Bei Anordnung B wird aus dieser Transaktion eine datengridübergreifende Transaktion, weil auf jede Ladenentität zugegriffen werden muss, um festzustellen, ob das Produkt in diesem Laden verkauft wird. Daraus ergibt sich ein enormer Leistungsvorteil für Anordnung A.

Skalierung mit normalisierten Daten

Eine zulässige Verwendung von datengridübergreifenden Transaktionen ist die Skalierung der Datenverarbeitung. Wenn ein Datengrid 5 Computer enthält und eine datengridübergreifende Transaktion zugeteilt wird, die 100.000 Datensätze auf jedem Computer durchsucht, durchsucht diese Transaktion insgesamt 500.000 Datensätze. Wenn der langsamste Computer im Datengrid 10 dieser Transaktionen pro Sekunde ausführen kann, ist das Datengrid in der Lage, 5.000.000 Datensätze pro Sekunde zu durchsuchen. Wenn sich die Daten im Grid verdoppeln, muss jeder Computer 200.000 Datensätze durchsuchen, und jede Transaktion durchsucht insgesamt 1.000.000 Datensätze. Diese Datenzunahme verringert den Durchsatz des langsamsten Computers auf 5 Transaktionen pro Sekunde und damit den Durchsatz des Datengrids auf 5 Transaktionen pro Sekunde. Das Datengrid durchsucht weiterhin 5.000.000 Datensätze pro Sekunde.

In diesem Szenario kann jeder Computer durch die Verdopplung der Computeranzahl zu seiner vorherigen Last von 100.000 Datensätzen zurückkehren, und der langsamste Computer kann wieder 10 dieser Transaktionen pro Sekunde verarbeiten. Der Durchsatz des Datengrids bleibt bei 10 Anforderungen pro Sekunde, aber jetzt verarbeitet jede Transaktion 1.000.000 Datensätze, sodass das Grid seine Kapazität in Bezug auf die Verarbeitung von Datensätzen auf 10.000.000 pro Sekunde verdoppelt hat.

Für Anwendungen wie Suchmaschinen, die sowohl in Bezug auf die Datenverarbeitung (angesichts der zunehmenden Größe des Internets) als auch in Bezug auf den Durchsatz (angesichts der zunehmenden Anzahl an Benutzern) skalierbar sein müssen, müssen Sie mehrere Grids mit einem

Umlaufverfahren für die Anforderungen zwischen den Datengrids erstellen. Wenn Sie den Durchsatz erhöhen müssen, fügen Sie Computer und ein weiteres Datengrid für die Bearbeitung der Anforderungen hinzu. Wenn die Datenverarbeitung erhöht werden muss, fügen Sie weitere Computer hinzu, und halten Sie die Anzahl der Datengrids konstant.

Anwendungen entwickeln, die mehrere Partitionen in einer einzigen Transaktion aktualisieren

Java 8.6+

Wenn Ihre Daten auf mehrere Partitionen im Datengrid verteilt sind, können Sie mehrere Partitionen in einer einzigen Transaktion lesen und aktualisieren. Dieser Typ von Transaktion wird als Mehrpartitionstransaktion bezeichnet und verwendet das Protokoll für zweiphasige Festschreibung für die Koordination und die Wiederherstellung der Transaktion beim Auftreten eines Fehlers.

Übersicht über die Sicherheit

WebSphere eXtreme Scale kann den Datenzugriff sichern, unter anderem durch Integration mit externen Sicherheitsprovidern.

Anmerkung: In einem vorhandenen nicht zwischengespeicherten Datenspeicher, z. B. einer Datenbank, haben Sie wahrscheinlich integrierte Sicherheitsfeatures, die Sie nicht aktiv konfigurieren oder aktivieren müssen. Nachdem Sie Ihre Daten jedoch mit eXtreme Scale zwischengespeichert haben, müssen Sie die daraus resultierende wichtige Tatsache berücksichtigen, dass die Sicherheitsfeatures Ihres Back-Ends nicht mehr wirksam sind. Sie können die Sicherheit von eXtreme Scale auf den erforderlichen Stufen konfigurieren, sodass Ihre neue zwischengespeicherte Datenarchitektur ebenfalls sicher ist.
Es folgt eine kurze Zusammenfassung der Sicherheitsfeatures von eXtreme Scale. Ausführlichere Informationen zur Konfiguration der Sicherheit finden Sie in der Veröffentlichung *Verwaltung* und in der Veröffentlichung *Programmierung*.

Grundlegende Informationen zur verteilten Sicherheit

Die verteilte Sicherheit von eXtreme Scale basiert auf drei Schlüsselkonzepten:

Vertrauenswürdige Authentifizierung
: Die Möglichkeit, die Identität des Anforderers zu bestimmen. WebSphere eXtreme Scale unterstützt Client/Server- und Server/Server-Authentifizierung.

Berechtigung
: Die Möglichkeit, dem Anforderer Zugriffsberechtigungen zu erteilen. WebSphere eXtreme Scale unterstützt verschiedene Berechtigungen für verschiedene Operationen.

Sicherer Transport
: Die sichere Übertragung von Daten über ein Netz. WebSphere eXtreme Scale unterstützt die Protokolle Layer Security/Secure Sockets Layer (TLS/SSL).

Authentifizierung

WebSphere eXtreme Scale unterstützt ein verteiltes Client/Server-Framework. Eine Client/Server-Sicherheitsinfrastruktur ist verfügbar, um den Zugriff auf Server von eXtreme Scale zu sichern. Wenn der Server von eXtreme Scale beispielsweise eine Authentifizierung erfordert, muss ein Client von eXtreme Scale Berechtigungsnach-

weise für die Authentifizierung beim Server vorlegen. Diese Berechtigungsnachweise können eine Kombination von Benutzername und Kennwort, ein Clientzertifikat, ein Kerberos-Ticket oder Daten sein, die in einem zwischen Client und Server vereinbarten Format präsentiert werden.

Berechtigung

Berechtigungen von WebSphere eXtreme Scale basieren auf Subject-Objekten und Berechtigungen. Sie können Java Authentication and Authorization Services (JAAS) für die Berechtigung des Zugriffs verwenden, oder Sie können eine angepasste Lösung wie Tivoli Access Manager (TAM) für die Behandlung der Berechtigungen integrieren. Die folgenden Berechtigungen können einem Client oder einer Gruppe erteilt werden:

Mapberechtigung
Berechtigung zum Durchführen von Einfüge-, Lese-, Aktualisierungs-, Bereinigungs- oder Löschoperationen in Maps.

ObjectGrid-Berechtigung
Berechtigung zum Ausführen von Objekt- oder Entitätsabfragen für ObjectGrid-Objekte.

DataGrid-Agentenberechtigung
Berechtigung für die Implementierung von DataGrid-Agenten in einem ObjectGrid.

Serverseitige Mapberechtigung
Berechtigung zum Replizieren einer Server-Map auf der Clientseite oder zum Erstellen eines dynamischen Index für die Server-Map.

Verwaltungsberechtigung
Berechtigung für die Ausführung von Verwaltungstasks.

Transportsicherheit

Zum Sichern der Client/Server-Kommunikation unterstützt WebSphere eXtreme Scale TLS/SSL. Diese Protokolle bieten Sicherheit auf Transportebene mit Authentizität, Integrität und Vertraulichkeit für eine sichere Verbindung zwischen einem Client und einem Server von eXtreme Scale.

Gridsicherheit

In einer sicheren Umgebung muss ein Server in der Lage sein, die Authentizität eines anderen Servers zu prüfen. WebSphere eXtreme Scale verwendet für diesen Zweck einen Mechanismus mit Shared-Secret-Schlüsselzeichenfolgen. Dieser Shared-Secret-Schlüsselmechanismus gleicht einem gemeinsam genutzten Kennwort. Alle Server von eXtreme Scale stimmen der Verwendung einer gemeinsamen Shared-Secret-Zeichenfolge zu. Wenn ein Server dem Datengrid beitritt, wird er aufgefordert, diese Shared-Secret-Zeichenfolge vorzulegen. Wenn die Shared-Secret-Zeichenfolge des beitretenden Servers der Zeichenfolge im Masterserver entspricht, kann der Server dem Grid beitreten. Andernfalls wird die Beitrittsanforderung zurückgewiesen.

Das Senden einer Shared-Secret-Zeichenfolge als Klartext ist nicht sicher. Die Sicherheitsinfrastruktur von eXtreme Scale stellt ein SecureTokenManager-Plug-in bereit, über das der Server den geheimen Schlüssel vor dem Senden sichern kann. Sie können festlegen, wie die Sicherungsoperation implementiert wird. WebSphere eXtreme Scale stellt eine Implementierung bereit, in der die Sicherungsoperation so

implementiert ist, dass das Shared Secret verschlüsselt und signiert wird.

JMX-Sicherheit (Java Management Extensions) in einer dynamischen Implementierungstopologie

Die JMX-MBean-Sicherheit wird in allen Versionen von eXtreme Scale unterstützt. Clients der Katalogserver-MBeans und Container-Server-MBeans können authentifiziert werden und auf die MBean-Operationen zugreifen.

Lokale Sicherheit von eXtreme Scale

Die lokale Sicherheit von eXtreme Scale unterscheidet sich vom verteilten eXtreme-Scale-Modell, weil die Anwendung direkt instanziiert wird und eine ObjectGrid-Instanz verwendet. Ihre Anwendung und eXtreme-Scale-Instanzen befinden sich in derselben Java Virtual Machine (JVM). Da es in diesem Modell kein Client/Server-Konzept gibt, wird die Authentifizierung nicht unterstützt. Ihre Anwendungen müssen ihre Authentifizierung selbst verwalten und anschließend das authentifizierte Subject-Objekt an eXtreme Scale übergeben. Der Berechtigungsmechanismus, der für das lokale Programmiermodell von eXtreme Scale verwendet wird, ist jedoch dasselbe wie beim Client/Server-Modell.

Konfiguration und Programmierung

Weitere Informationen zum Konfigurieren und Programmieren der Sicherheit finden Sie in den Abschnitten Sicherheitsintegration mit externen Providern und Sicherheits-API.

Übersicht über REST-Datenservices

Java

Der REST-Datenservice von WebSphere eXtreme Scale ist ein Java-HTTP-Service, der mit Microsoft WCF Data Services (offiziell ADO.NET Data Services) kompatibel ist und Open Data Protocol (OData) implementiert. Microsoft WCF Data Services ist mit dieser Spezifikation kompatibel, wenn Visual Studio 2008 SP1 und .NET Framework 3.5 SP1 verwendet werden.

Kompatibilitätsanforderungen

Der REST-Datenzugriff ermöglicht jedem HTTP-Client den Zugriff auf ein Datengrid. Der REST-Datenservice ist mit der Unterstützung der WCF Data Services kompatibel, die mit Microsoft .NET Framework 3.5 SP1 bereitgestellt wird. Anwendungen, die REST unterstützen, können mit den zahlreichen Tools, die im Lieferumfang von Microsoft Visual Studio 2008 SP1 enthalten sind, entwickelt werden. Die Abbildung enthält eine Übersicht über die Interaktion von WCF Data Services mit Clients und Datenbanken.

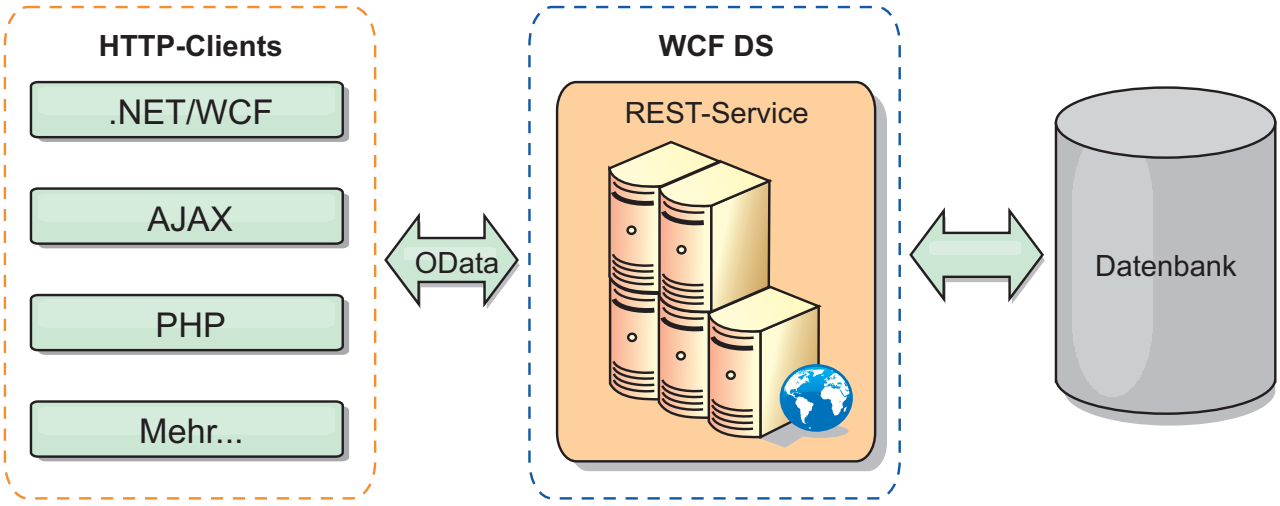


Abbildung 39. Microsoft WCF Data Services

WebSphere eXtreme Scale enthält einen umfassend ausgestatteten API-Satz für Java-Clients. Wie in der folgenden Abbildung gezeigt, ist der REST-Datenservice ein Gateway zwischen HTTP-Clients und dem eXtreme-Scale-Datengrid, das mit dem Grid über einen eXtreme-Scale-Client kommuniziert. Der REST-Datenservice ist ein Java-Servlet, das flexible Implementierungen für gängige JEE-Plattformen (Java Platform, Enterprise Edition) wie WebSphere Application Server unterstützt. Der REST-Datenservice kommuniziert mit dem eXtreme-Scale-Datengrid über die Java-APIs von WebSphere eXtreme Scale. Er unterstützt WCF-Data-Services-Clients und alle anderen Clients, die mit HTTP und XML kommunizieren können.

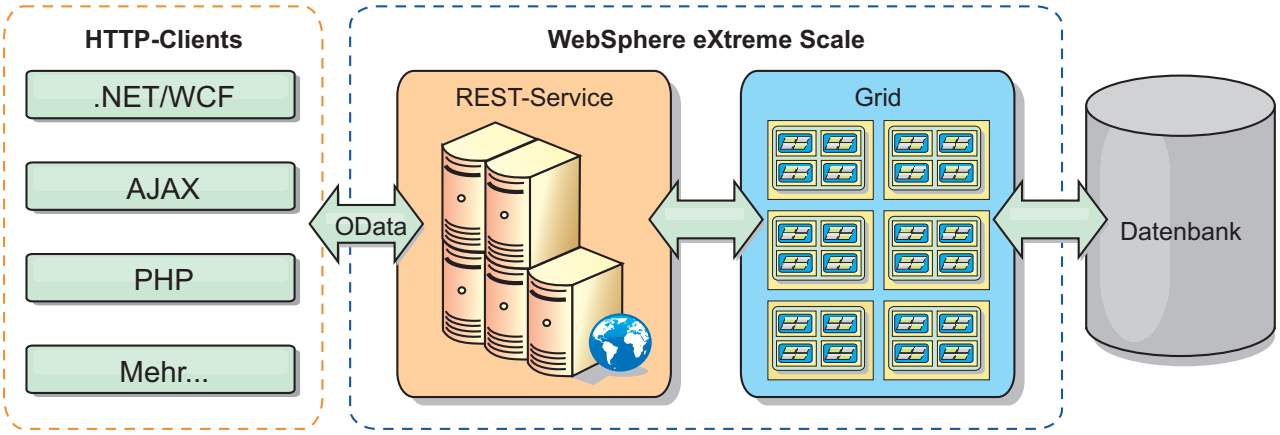


Abbildung 40. REST-Datenservice von WebSphere eXtreme Scale

Lesen Sie den Abschnitt REST-Datenservices konfigurieren, oder verwenden Sie die folgenden Links, um mehr über WCF Data Services zu erfahren.

- Microsoft WCF Data Services Developer Center
- ADO.NET Data Services overview on MSDN
- Whitepaper: Using ADO.NET Data Services
- Atom Publish Protocol: Data Services URI and Payload Extensions
- Conceptual Schema Definition File Format
- Entity Data Model for Data Services Packaging Format
- Open Data Protocol

- Open Data Protocol FAQ

Features

Diese Version des REST-Datenservice von eXtreme Scale unterstützt die folgenden Features:

- Automatische Modellierung von Entitäten der eXtreme-Scale-API "EntityManager" als Entitäten von WCF Data Services, die die folgende Unterstützung umfasst:
  - Konvertierung von Java-Datentypen in Typen des Entitätsdatenmodells
  - Unterstützung der Entitätszuordnung
  - Unterstützung der Zuordnung von Schemastammelementen und Schlüsseln, die für partitionierte Datengrids erforderlich ist

  Weitere Informationen finden Sie unter Entitätsmodell.
- XML von Atom Publish Protocol (AtomPub oder APP) und Nutzdatenformat von JavaScript Object Notation (JSON)
- CRUD-Operationen (Create, Read, Update and Delete, Erstellen, Lesen, Aktualisieren und Löschen), die die entsprechenden HTTP-Anforderungsmethoden, POST, GET, PUT und DELETE, verwenden. Außerdem wird die Microsoft-Erweiterung MERGE unterstützt.

  Anmerkung: 8.6+ Die Methoden upsert und upsertAll ersetzen die ObjectMap-Methoden put und putAll. Verwenden Sie die Methode upsert, um der BackingMap und dem Ladeprogramm mitzuteilen, dass ein Eintrag im Datengrid den Schlüssel und den Wert im Grid ablegen muss. Die BackingMap und das Ladeprogramm führen entweder eine insert- oder update-Operation aus, um den Wert im Grid und im Ladeprogramm abzulegen. Wenn Sie die API upsert in Ihrer Anwendung ausführen, ruft das Ladeprogramm den LogElement-Typ UPSERT ab, der es Ladeprogrammen ermöglicht, merge- oder upsert-Aufrufe anstelle von insert- oder update-Aufrufen für die Datenbank abzusetzen.
- Einfache Abfragen unter Verwendung von Filtern
- Stapelabruf- und Änderungssetanforderungen
- Unterstützung partitionierter Datengrids für hohe Verfügbarkeit
- Interoperabilität mit Clients der eXtreme-Scale-API "EntityManager"
- Unterstützung für Standard-JEE-Web-Server
- Optimistisches Sperren bei gemeinsamen Zugriffen
- Benutzerberechtigung und -authentifizierung zwischen dem REST-Datenservice und dem eXtreme-Scale-Datengrid

Bekannte Probleme und Einschränkungen

- Getunnelte Anforderungen werden nicht unterstützt.

Kapitel 2. Planung

Bevor Sie WebSphere eXtreme Scale installieren und Ihre Datengridanwendungen implementieren, müssen Sie Ihre Cachingtopologie festlegen, die Kapazitätsplanung durchführen, die Hardware- und Softwarevoraussetzungen prüfen, die Einstellungen für den Netzbetrieb und die Optimierung prüfen usw. Sie können auch die Prüfliste für Betriebsbereitschaft verwenden, um sicherzustellen, dass Ihre Umgebung für die Implementierung von Anwendungen bereit ist.

Eine Beschreibung der bewährten Verfahren für das Entwerfen von eXtreme-Scale-Anwendungen finden Sie im folgenden Artikel auf developerWorks: Principles and best practices for building high performing and highly resilient WebSphere eXtreme Scale applications.

Topologie planen

Mit WebSphere eXtreme Scale kann Ihre Architektur speicherinternes Daten-Caching oder verteiltes Client/Server-Daten-Caching verwenden. Die Architektur kann verschiedene Beziehungen zu Ihren Datenbanken haben. Sie können die Topologie auch so konfigurieren, dass sie mehrere Rechenzentren umspannt.

WebSphere eXtreme Scale erfordert für den Betrieb eine minimale zusätzliche Infrastruktur. Die Infrastruktur setzt sich aus Scripts für die Installation, das Starten und Stoppen von Java-EE-Anwendungen in einem Server zusammen. Die zwischengespeicherten Daten werden in den Container-Servern gespeichert, und Clients stellen über Fernzugriff eine Verbindung zum Server her.

Speicherinterne Umgebungen

Wenn Sie die Implementierung in einer lokalen, speicherinternen Umgebung durchführen, wird WebSphere eXtreme Scale in einer einzigen Java Virtual Machine ausgeführt und nicht repliziert. Zum Konfigurieren einer lokalen Umgebung können Sie eine ObjectGrid-XML-Datei oder die ObjectGrid-APIs verwenden.

Verteilte Umgebungen

Wenn Sie die Implementierung in einer verteilten Umgebung durchführen, wird WebSphere eXtreme Scale in einer Reihe von Java Virtual Machines ausgeführt, was die Leistung, die Verfügbarkeit und die Skalierbarkeit erhöht. Mit dieser Konfiguration können Sie Datenreplikation und Partitionierung verwenden. Zusätzliche Server können hinzugefügt werden, ohne die vorhandenen eXtreme Scale-Server erneut starten zu müssen. Wie bei einer lokalen Umgebung ist in einer verteilten Umgebung eine ObjectGrid-XML-Datei oder eine entsprechende programmgesteuerte Konfiguration erforderlich. Außerdem müssen Sie eine XML-Implementierungsrichtliniendatei mit Konfigurationsdetails bereitstellen.

Sie können einfache Implementierungen erstellen oder große Implementierungen in Terabytegröße, in denen Tausende von Servern erforderlich sind.

Lokaler Speichercache

Im einfachsten Fall kann WebSphere eXtreme Scale als lokaler (nicht verteilter) speicherinterner Datengrid-Cache verwendet werden. Dies kann insbesondere für

Anwendungen mit sehr vielen gemeinsamen Zugriffen von Vorteil sein, in denen mehrere Threads auf transiente Daten zugreifen und diese ändern müssen. Die in einem lokalen Datengrid gespeicherten Daten können indexiert und mit Abfragen abgerufen werden. Abfragen helfen Ihnen bei der Arbeit mit großen speicherinternen Datasets. Die mit Java Virtual Machine (JVM) bereitgestellte Unterstützung ist zwar einsatzfähig, besitzt aber eine eingeschränkte Datenstruktur.

Die lokale speicherinterne Cachetopologie für WebSphere eXtreme Scale wird verwendet, um einen konsistenten, transaktionsorientierten Zugriff auf temporäre Daten in einer einzelnen Java Virtual Machine zu unterstützen.

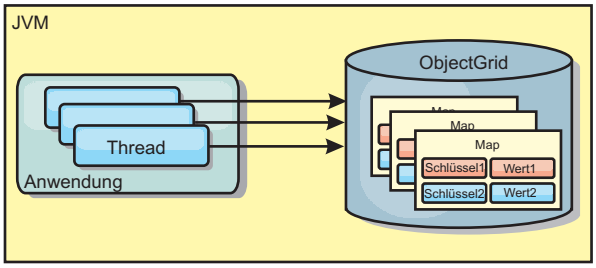


Abbildung 41. Szenario mit einem lokalen speicherinternen Speichercache

Vorteile

- Einfaches Setup: Ein ObjectGrid kann programmgesteuert oder deklarativ über die ObjectGrid-XML-Implementierungsdeskriptordatei oder mit anderen Frameworks wie Spring erstellt werden.
- Schnell: Jede BackingMap kann gesondert für eine optimale Speicherauslastung und gemeinsamen Zugriff optimiert werden.
- Ideal für Topologien mit einer einzigen JVM und einer kleinen Dateigruppe oder für das Caching von Daten, auf die häufig zugegriffen wird.
- Transaktionsorientiert: BackingMap-Aktualisierungen können zu einer einzigen Arbeitseinheit gruppiert und als letzter Teilnehmer in zweiphasige Transaktionen wie JTA-Transaktionen (Java Transaction Architecture) integriert werden.

Nachteile

- Keine Fehlertoleranz.
- Die Daten werden nicht repliziert. Speichercaches eignen sich am besten für schreibgeschützte Referenzdaten.
- Keine Skalierbarkeit. Die für die Datenbank erforderliche Speicherkapazität kann die JVM möglicherweise nicht bereitstellen.
- Es treten Probleme auf, wenn JVMs hinzugefügt werden.
  - Die Daten sind nicht so einfach partitionierbar.
  - Der Status muss in den JVMs manuell repliziert werden, da die einzelnen Cacheinstanzen ansonsten verschiedene Versionen derselben Daten enthalten könnten.
  - Die Invalidierung von Einträgen ist kostenintensiv.
  - Jeder Cache muss einzeln vorbereitet werden. Die Vorbereitungs- oder Aufwärmphase ist der Zeitraum, in dem eine Gruppe von Daten geladen wird, damit der Cache mit gültigen Daten gefüllt wird.

Einsatz

Die Implementierungstopologie mit dem lokalen Speichercache sollte nur verwendet werden, wenn die Menge der zwischenzuspeichernden Daten klein ist (in eine einzige JVM passt) und relativ stabil ist. Bei diesem Ansatz müssen veraltete Daten toleriert werden. Die Verwendung von Evictor (Bereinigungsprogramm), um nur die am häufigsten verwendeten oder die zuletzt verwendeten Daten im Cache zu verwalten, kann dabei helfen, den Cache klein zu halten und die Relevanz der Daten zu erhöhen.

Auf Peers replizierter lokaler Cache

Sie müssen sicherstellen, dass der Cache synchronisiert wird, wenn mehrere Prozesse mit unabhängigen Cacheinstanzen vorhanden sind. Um sicherzustellen, dass die Cacheinstanzen synchronisiert werden, richten Sie einen auf Peers replizierten Cache mithilfe von Java Message Service (JMS) ein.

WebSphere eXtreme Scale enthält zwei Plug-ins, die Transaktionsänderungen automatisch zwischen Peer-ObjectGrid-Instanzen weitergeben. Das Plug-in "JMSObjectGridEventListener" gibt eXtreme-Scale-Änderungen automatisch über JMS weiter.

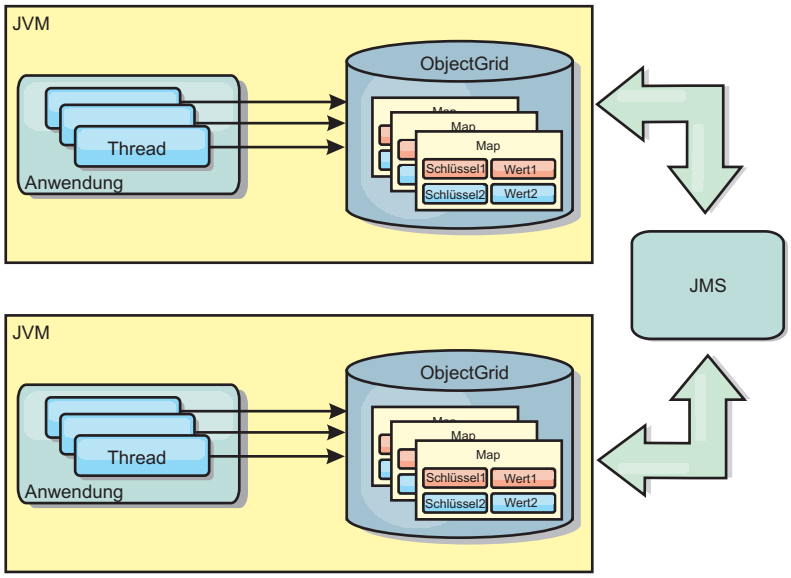


Abbildung 42. Auf Peers replizierter Cache mit Änderungen, die über JMS weitergegeben werden

Wenn Sie in einer Umgebung mit WebSphere Application Server arbeiten, ist auch das TranPropListener-Plug-in verfügbar. Das TranPropListener-Plug-in verwendet den High Availability Manager (kurz HA-Manager), um die Änderungen an jede Peer-Cacheinstanz weiterzugeben.

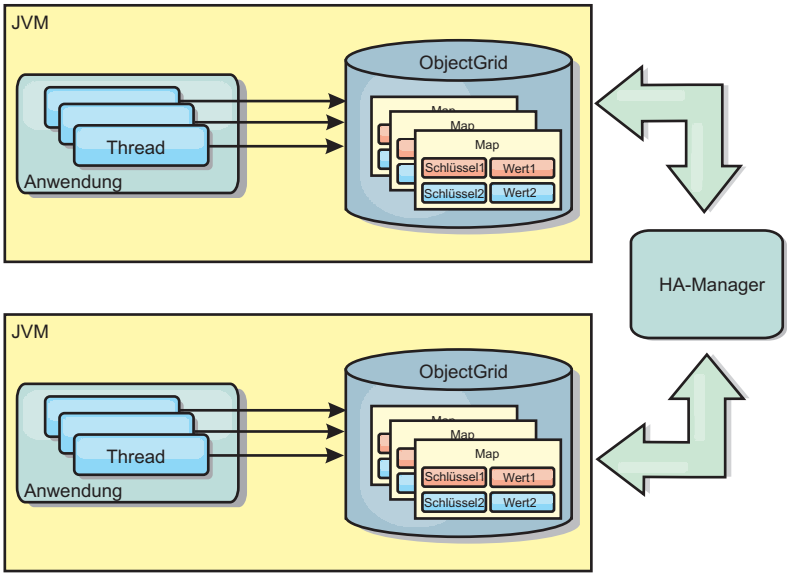


Abbildung 43. Auf Peers replizierter Cache mit Änderungen, die über den High Availability Manager weitergegeben werden

Vorteile

- Die Gültigkeit der Daten ist höher, weil sie häufiger aktualisiert werden.
- Mit dem TranPropListener-Plug-in kann eXtreme Scale wie die lokale Umgebung über das Programm oder deklarativ über die XML-Implementierungsdeskriptordatei von eXtreme Scale oder mit anderen Frameworks wie Spring erstellt werden. Die Integration mit dem High Availability Manager erfolgt automatisch.
- Jede BackingMap kann gesondert für eine optimale Speicherauslastung und gemeinsamen Zugriff optimiert werden.
- BackingMap-Aktualisierungen können zu einer einzigen Arbeitseinheit gruppiert und als letzter Teilnehmer in zweiphasige Transaktionen wie JTA-Transaktionen (Java Transaction Architecture) integriert werden.
- Ideal für Topologien mit wenigen JVMs und einer angemessen kleinen Dateigruppe oder für das Caching von Daten, auf die häufig zugegriffen wird.
- Änderungen an eXtreme Scale werden in allen Peer-Instanzen von eXtreme Scale repliziert. Die Änderungen sind so lange konsistent, wie eine permanente Subskription verwendet wird.

Nachteile

- Die Konfiguration und Verwaltung für JMSObjectGridEventListener kann komplex sein. eXtreme Scale kann über das Programm oder deklarativ über die XML-Implementierungsdeskriptordatei von eXtreme Scale oder mit anderen Frameworks wie Spring erstellt werden.
- Nicht skalierbar: Die für die Datenbank erforderliche Speicherkapazität kann die JVM möglicherweise nicht bereitstellen.
- Funktioniert nicht ordnungsgemäß, wenn Java Virtual Machines hinzugefügt werden:
  - Die Daten sind nicht so einfach partitionierbar.
  - Die Invalidierung von Einträgen ist kostenintensiv.
  - Jeder Cache muss einzeln vorbereitet werden.

Einsatz

Verwenden Sie die Implementierungstopologie nur, wenn das zwischenzuspeichernde Datenvolumen klein ist, in eine einzige JVM passt und relativ stabil ist.

Integrierter Cache

WebSphere-eXtreme-Scale-Grids können in vorhandenen Prozessen als integrierte eXtreme-Scale-Server ausgeführt oder als externe Prozesse verwaltet werden.

Integrierte Grids sind hilfreich, wenn Sie mit einem Anwendungsserver wie WebSphere Application Server arbeiten. Sie können eXtreme-Scale-Server, die nicht integriert sind, über Befehlszeilenscripts starten und in einem Java-Prozess ausführen.

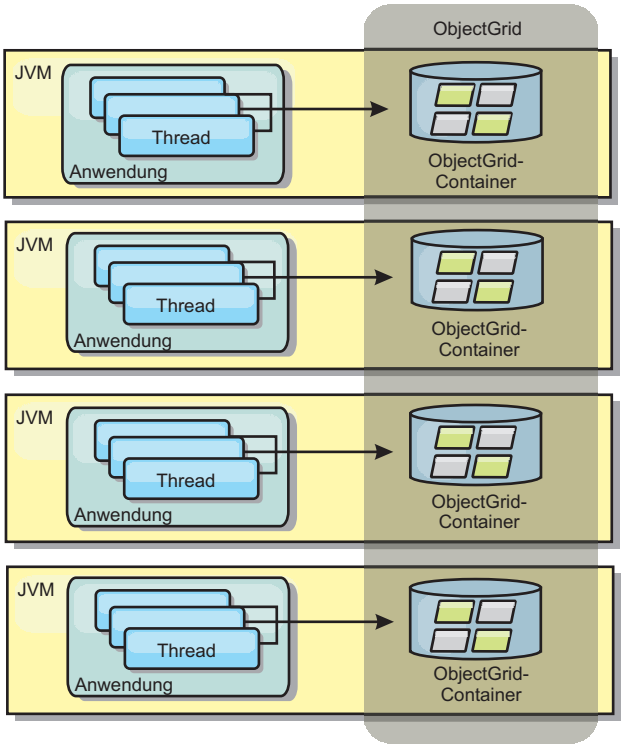


Abbildung 44. Integrierter Cache

Vorteile

- Vereinfachte Verwaltung, da weniger Prozesse zu verwalten sind
- Vereinfachte Anwendungsimplementierung, weil das Grid den Klassen-Loader der Clientanwendung verwendet
- Unterstützung von Partitionierung und hoher Verfügbarkeit

Nachteile

- Erhöhter Speicherbedarf im Clientprozess, weil alle Daten im Prozess erfasst werden
- Erhöhte CPU-Auslastung für die Bearbeitung von Clientanforderungen
- Erschwerte Verarbeitung von Anwendungsupgrades, da Clients dieselben Java-Anwendungsarchivdateien wie die Server verwenden

- Weniger Flexibilität. Die Skalierung von Clients und Grid-Servern ist nicht linear. Wenn Server extern definiert werden, haben Sie mehr Flexibilität bei der Verwaltung der Prozessanzahl.

Einsatz

Verwenden Sie integrierte Grids, wenn im Clientprozess reichlich Speicher für die Griddaten und potenzielle Failover-Daten frei ist.

Weitere Informationen finden Sie im Abschnitt zum Aktivieren des Clientinvalidierungsmechanismus in der Veröffentlichung *Verwaltung*.

Verteilter Cache

In den meisten Fällen wird WebSphere eXtreme Scale als gemeinsam genutzter Cache verwendet, um einen transaktionsgesteuerten Zugriff auf Dateien durch mehrere Komponenten zu ermöglichen, wo ansonsten eine traditionelle Datenbank verwendet werden würde. Bei der Verwendung eines gemeinsam genutzten Caches muss keine Datenbank konfiguriert werden.

Kohärenz des Caches

Der Cache ist kohärent, weil alle Clients dieselben Daten im Cache sehen. Jede einzelne Information wird im Cache eines einzigen Servers gespeichert. Auf diese Weise werden unnötige Datensatzkopien verhindert, die potenziell verschiedene Versionen der Daten enthalten könnten. Ein kohärenter Cache kann außerdem mehr Daten aufnehmen, wenn dem Datengrid weitere Server hinzugefügt werden. Die Skalierung des Caches erfolgt linear zum Wachstum des Grids. Da Clients über Prozedurfernaufrufe auf Daten in diesem Datengrid zugreifen, wird der Cache auch als ferner Cache bezeichnet. Durch die Datenpartitionierung enthält jeder Prozess einen eindeutigen Teil der Gesamtdatengruppe. Größere Datengrids können mehr Daten aufnehmen und mehr Anforderungen für diese Daten bearbeiten. Aufgrund der Kohärenz müssen auch keine Daten zur Invalidierung im Datengrid verteilt werden, weil es keine veralteten Daten gibt. Der kohärente Cache enthält jeweils nur die aktuelle Kopie jeder Information.

Wenn Sie in einer Umgebung mit WebSphere Application Server arbeiten, ist auch das TranPropListener-Plug-in verfügbar. Das TranPropListener-Plug-in verwendet die Komponente "High Availability Manager" (kurz HA-Manager) von WebSphere Application Server, um die Änderungen an die einzelnen Peer-ObjectGrid-Cacheinstanzen weiterzugeben.

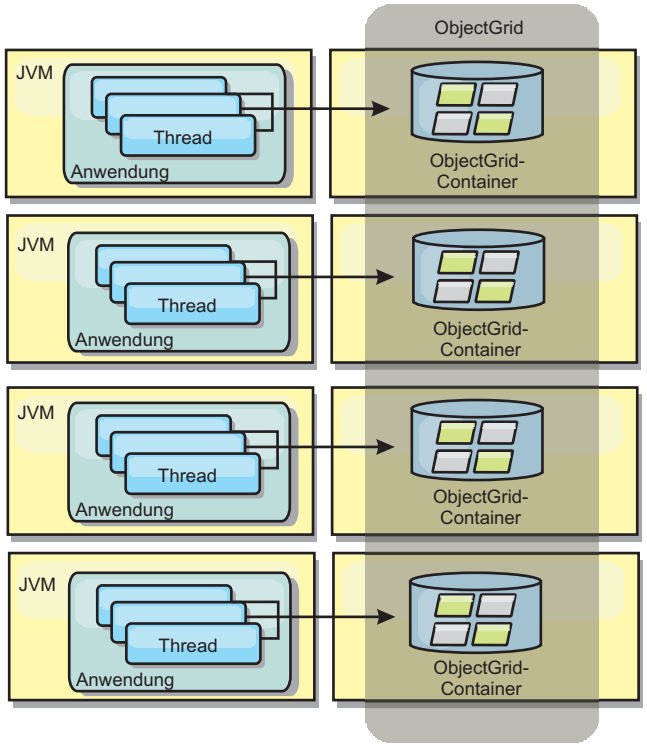


Abbildung 45. Verteilter Cache

Naher Cache

Clients können optional einen lokalen integrierten Cache haben, wenn eXtreme Scale in einer verteilten Topologie verwendet wird. Dieser optionale Cache wird als naher Cache bezeichnet. Er ist ein unabhängiges ObjectGrid in jedem Client, das als Cache für den fernen serverseitigen Cache dient. Der nahe Cache wird standardmäßig aktiviert, wenn eine optimistische Sperrstrategie oder keine Sperrstrategie konfiguriert ist, und kann nicht verwendet werden, wenn eine pessimistische Sperrstrategie konfiguriert ist.

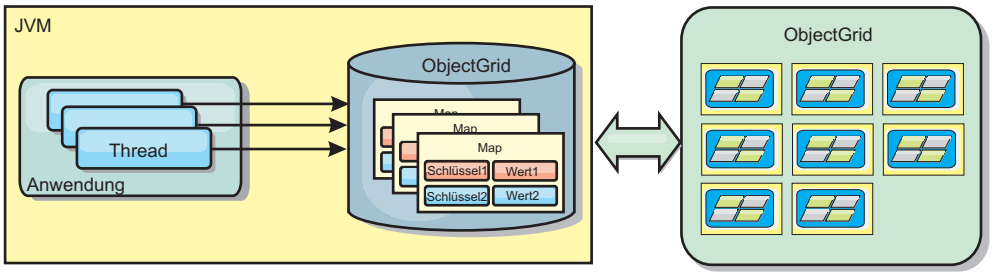


Abbildung 46. Naher Cache

Ein naher Cache ist sehr schnell, weil er den speicherinternen Zugriff auf einen Teil der gesamten zwischengespeicherten Daten ermöglicht, die fern in den Servern von eXtreme Scale gespeichert sind. Der nahe Cache ist nicht partitioniert und enthält Daten aus allen fernen eXtreme-Scale-Partitionen. WebSphere eXtreme Scale kann bis zu drei Cacheschichten haben. Diese sind im Folgenden erläutert:

1. Der Cache auf der Transaktionsschicht enthält alle Änderungen für eine einzelne Transaktion. Der Transaktionscache enthält eine Arbeitskopie der Daten, bis

die Transaktion festgeschrieben wird. Wenn eine Clienttransaktion Daten aus einer ObjectMap anfordert, wird zuerst die Transaktion geprüft.

2. Der nahe Cache auf der Clientschicht enthält einen Teil der Daten aus der Serverschicht. Wenn die Daten nicht auf der Transaktionsschicht zu finden sind, werden die Daten (sofern verfügbar) von der Cacheschicht abgerufen und in den Transaktionscache eingefügt.
3. Das Datengrid auf der Serverschicht enthält den Hauptteil der Daten und wird von allen Clients gemeinsam genutzt. Die Serverschicht kann partitioniert werden, was die Zwischenspeicherung großer Datenvolumen ermöglicht. Wenn der nahe Cache des Clients die Daten nicht enthält, werden die Daten von der Serverschicht abgerufen und in den Clientcache eingefügt. Die Serverschicht kann auch ein Loader-Plug-in (Ladeprogramm) haben. Wenn das Datengrid die angeforderten Daten nicht enthält, wird das Ladeprogramm aufgerufen, das die Daten aus dem Back-End-Datenspeicher abruft und in das Grid einfügt.

Informationen zum Inaktivieren des nahen Caches finden Sie unter Nahen Cache konfigurieren.

Vorteil

- Schnelle Antwortzeiten, weil alle Zugriffe auf die Daten lokal erfolgen. Indem die Daten zuerst im nahen Cache gesucht werden, wird eine Konsultation des Servergrid gespart, wodurch selbst die fernen Daten lokal zugänglich werden.

Nachteile

- Die Verweildauer veralteter Daten erhöht sich, weil der nahe Cache auf jeder Schicht unter Umständen nicht mit den aktuellen Daten im Datengrid synchronisiert sind.
- Es muss ein Bereinigungsprogramm zum Invalidieren von Daten verwendet werden, um Speicherengpässe zu verhindern.

Einsatz

Verwenden Sie diese Technik, wenn die Antwortzeiten wichtig und veraltete Daten tolerierbar sind.

Datenbankintegration: Write-behind-, Inline- und Neben-Caching

WebSphere eXtreme Scale wird als Front-End für eine traditionelle Datenbank verwendet und macht Leseaktivitäten überflüssig, die normalerweise an die Datenbank übertragen werden. Ein kohärenter Cache kann direkt oder indirekt über einen ORM (Object Relational Mapper) mit einer Anwendung verwendet werden. Der kohärente Cache kann dann die Datenbank bzw. das Back-End von Leseaktivitäten entlasten. In einem geringfügig komplexeren Szenario, wie z. B. beim transaktionsorientierten Zugriff auf einen Datenbestand, in dem nur einige der Daten traditionelle Persistenzgarantien erfordern, können Sie Filter verwenden, um selbst die Schreibtransaktionen auszulagern.

Sie können WebSphere eXtreme Scale als hoch flexiblen speicherinternen Datenbankverarbeitungsbereich konfigurieren. WebSphere eXtreme Scale ist jedoch kein ORM. Das Produkt weiß nicht, woher die Daten im Datengrid stammen. Eine Anwendung oder ein ORM kann Daten in einem eXtreme-Scale-Server ablegen. Die Datenquelle ist dafür verantwortlich sicherzustellen, dass sie mit der Datenbank, aus der die Daten stammen, konsistent bleibt. Das bedeutet, dass eXtreme Scale Daten, die automatisch aus einer Datenbank extrahiert werden, nicht ungültig ma-

chen kann. Die Anwendung bzw. der Mapper muss diese Funktion bereitstellen und die Daten verwalten, die in eXtreme Scale gespeichert werden.

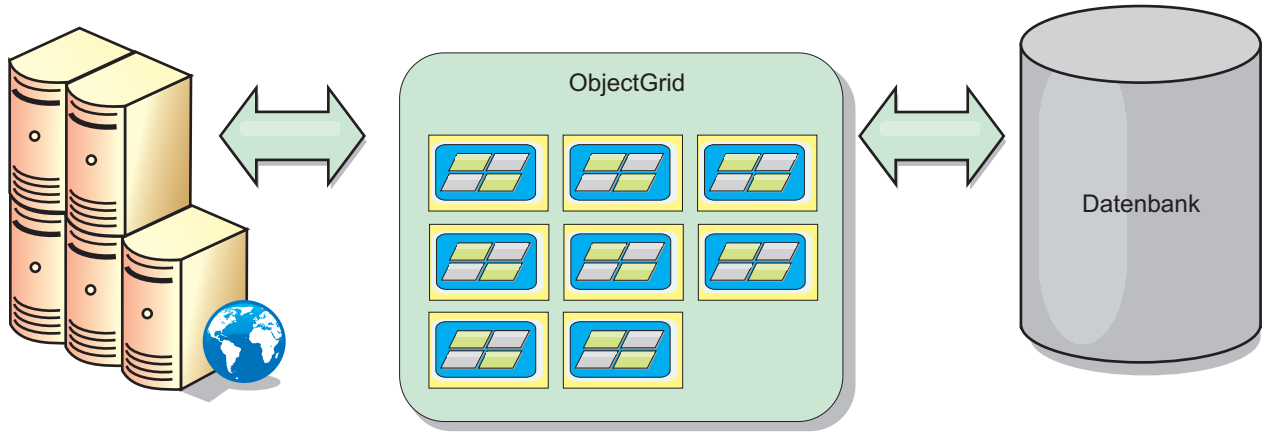


Abbildung 47. ObjectGrid als Datenbankpuffer

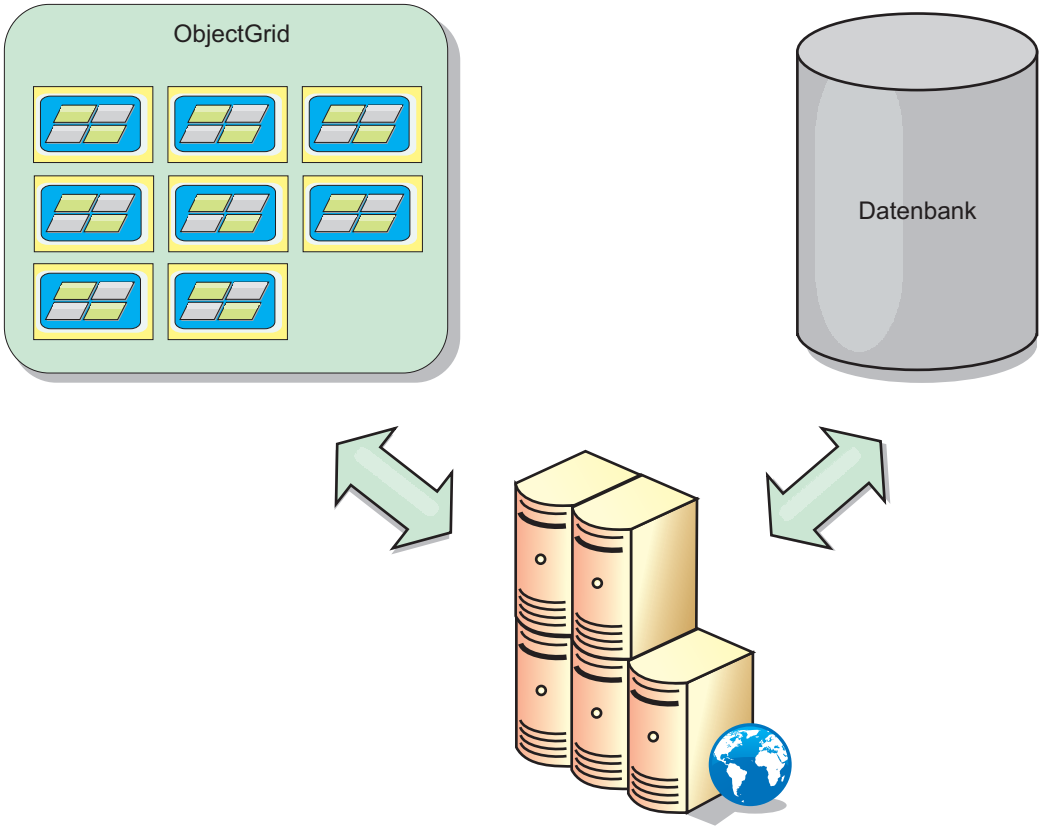


Abbildung 48. ObjectGrid als Nebencache

Teilcache und vollständiger Cache

WebSphere eXtreme Scale kann als Teilcache oder als vollständiger Cache eingesetzt werden. In einem Teilcache wird nur ein Teil der gesamten Daten gespeichert, wohingegen in einem vollständigen Cache alle Daten gespeichert werden. Ein Teilcache kann nach und nach bedarfsgesteuert gefüllt werden. Der Zugriff auf Teilcaches erfolgt gewöhnlich über Schlüssel (und nicht über Indizes oder Abfragen), da die Daten nur teilweise verfügbar sind.

Teilcache

Wenn ein Schlüssel nicht im Teilcache vorhanden ist oder wenn die Daten nicht verfügbar sind und ein Cachefehler auftritt, wird die nächste Schicht aufgerufen. Die Daten werden beispielsweise aus der Datenbank abgerufen und in die Cacheschicht des Datengrids eingefügt. Bei der Verwendung einer Abfrage oder eines Index wird nur auf die derzeit geladenen Werte zugegriffen, und die Anforderungen werden nicht an die anderen Schichten weitergeleitet.

Vollständiger Cache

Ein vollständiger Cache enthält alle erforderlichen Daten, und der Zugriff kann über Attribute ohne Schlüsselfunktion mit Indizes oder Abfragen erfolgen. Ein vollständiger Cache wird vorher mit Daten aus der Datenbank geladen, bevor die Anwendung versucht, auf die Daten zuzugreifen. Ein vollständiger Cache kann als Datenbankersatz dienen, nachdem die Daten geladen wurden. Da alle Daten verfügbar sind, können Abfragen und Indizes verwendet werden, um Daten zu suchen und zusammenzufassen.

Nebencache

Wenn WebSphere eXtreme Scale als Nebencache verwendet wird, wird das Back-End für das Datengrid verwendet.

Nebencache

Sie können das Produkt als Nebencache für die Datenzugriffsschicht einer Anwendung konfigurieren. In diesem Szenario wird WebSphere eXtreme Scale verwendet, um Objekte temporär zu speichern, die normalerweise aus einer Back-End-Datenbank abgerufen werden. Anwendungen prüfen, ob das Datengrid die Daten enthält. Wenn die Daten im Datengrid enthalten sind, werden die Daten an den Aufrufenden zurückgegeben. Wenn die Daten nicht vorhanden sind, werden die Daten aus der Back-End-Datenbank abgerufen. Anschließend werden die Daten in das Datengrid eingefügt, damit die nächste Anforderung die zwischengespeicherte Kopie verwenden kann. Die folgende Abbildung veranschaulicht, wie WebSphere eXtreme Scale als Nebencache mit einer beliebigen Datenzugriffsschicht wie OpenJPA oder Hibernate verwendet werden kann.

Nebencache-Plug-ins für Hibernate und OpenJPA

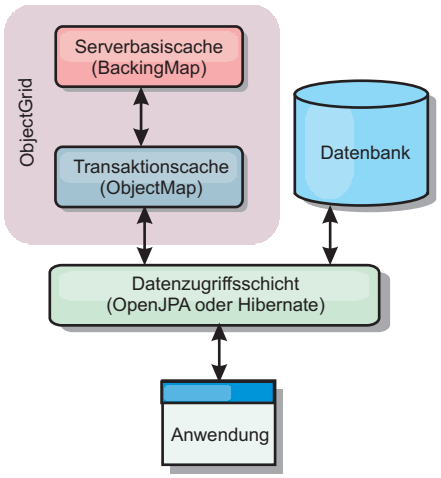


Abbildung 49. Nebencache

Cache-Plug-ins für OpenJPA und Hibernate sind in WebSphere eXtreme Scale enthalten. Damit können Sie das Produkt als automatischen Nebencache verwenden. Durch die Verwendung von WebSphere eXtreme Scale als Cache-Provider kann die Leistung beim Lesen und Abfragen von Daten verbessert und die Belastung der Datenbank verringert werden. WebSphere eXtreme Scale bietet im Vergleich mit integrierten Cacheimplementierungen verschiedene Vorteile, weil der Cache automatisch in allen Prozessen repliziert wird. Wenn ein Client einen Wert zwischenspeichert, können alle andere Clients den zwischengespeicherten Wert verwenden.

Inline-Cache

Sie können das Inline-Caching für ein Datenbank-Back-End oder als Nebencache für eine Datenbank konfigurieren. Beim Inline-Caching wird eXtreme Scale als primäres Mittel für die Interaktion mit den Daten verwendet. Bei der Verwendung von eXtreme Scale als Inline-Cache interagiert die Anwendung über ein Loader-Plug-in mit dem Back-End.

Inline-Cache

Bei Verwendung als Inline-Cache interagiert WebSphere eXtreme Scale über ein Loader-Plug-in mit dem Back-End. Dieses Szenario kann den Datenzugriff vereinfachen, weil Anwendungen direkt auf die APIs von eXtreme Scale zugreifen können. Es werden verschiedene Caching-Szenarien in eXtreme Scale unterstützt, um sicherzustellen, dass die Daten im Cache und die Daten im Back-End synchronisiert sind. Die folgende Abbildung veranschaulicht, wie ein Inline-Cache mit der Anwendung und dem Back-End interagiert.

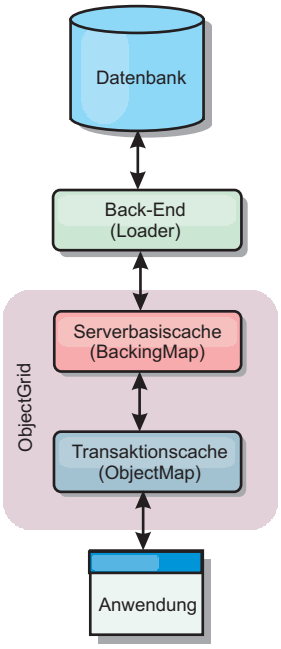


Abbildung 50. Inline-Cache

Die Option für Inline-Caching vereinfacht den Datenzugriff, weil sie Anwendungen den direkten Zugriff auf die eXtreme-Scale-APIs ermöglicht. WebSphere eXtreme Scale unterstützt mehrere Szenarien mit Inline-Caching:

- Read-through
- Write-through
- Write-behind

Szenario mit Read-through-Caching

Ein Read-through-Cache ist ein Teilcache, in den nach und nach Dateneinträge nach Schlüssel geladen werden, wenn diese angefordert werden. Dies geschieht, ohne dass der Aufrufende wissen muss, wie die Einträge geladen werden. Wenn die Daten nicht im eXtreme-Scale-Cache gefunden werden, ruft eXtreme Scale die fehlenden Daten vom Loader-Plug-in ab, das die Daten aus der Back-End-Datenbank lädt und in den Cache einfügt. Nachfolgende Anforderungen für denselben Datenschlüssel werden im Cache gefunden, bis der Eintrag gelöscht, ungültig gemacht oder durch Bereinigung entfernt wird.

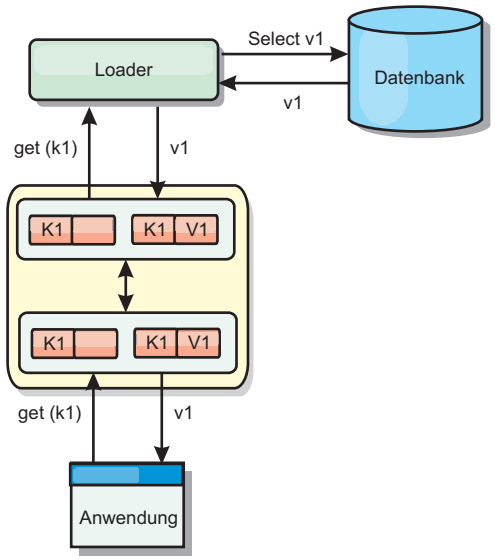


Abbildung 51. Read-through-Caching

Szenario mit Write-through-Caching

In einem Write-through-Cache (Durchschreibcache) erfolgt bei jedem Schreibvorgang in den Cache ein synchroner Schreibvorgang über den Loader in die Datenbank. Diese Methode gewährleistet die Konsistenz mit dem Back-End, verringert aber die Schreibleistung, weil die Datenbankoperation synchron erfolgt. Da der Cache und die Datenbank beide aktualisiert werden, werden bei nachfolgenden Leseoperationen dieselben Daten im Cache gefunden und Datenbankaufrufe vermieden. Ein Write-through-Cache wird häufig in Kombination mit einem Read-through-Cache verwendet.

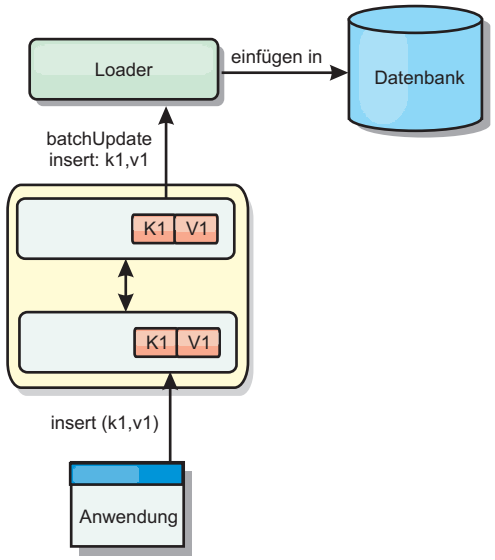


Abbildung 52. Write-through-Caching

Szenario mit Write-behind-Caching

Die Datenbanksynchronisation kann verbessert werden, indem Änderungen asynchron geschrieben werden. Dies wird als Write-behind- oder Write-back-Cache

(Rückschreibcache) bezeichnet. Änderungen, die normalerweise synchron in den Loader geschrieben werden, werden stattdessen in eXtreme Scale gepuffert und über einen Hintergrund-Thread in die Datenbank geschrieben. Die Schreibleistung wird erheblich verbessert, weil die Datenbankoperation aus der Clienttransaktion entfernt wird und die Schreibvorgänge in die Datenbank komprimiert werden können.

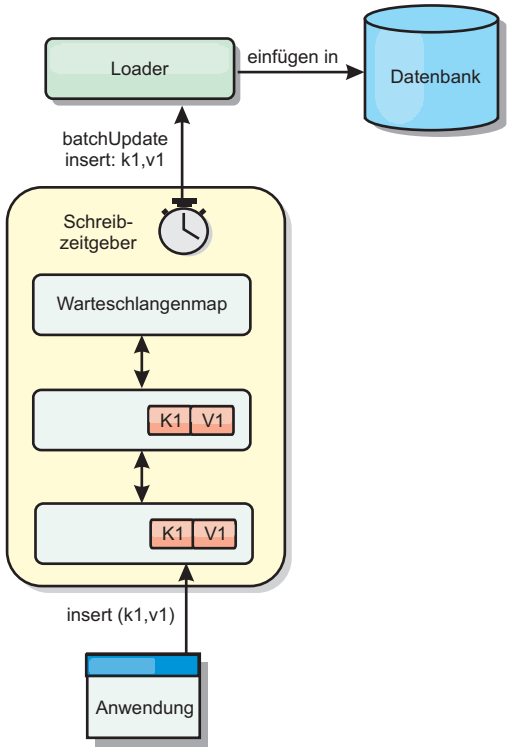


Abbildung 53. Write-behind-Caching

Write-behind-Caching

Java

Sie können Write-behind-Caching verwenden, um die Kosten für die Aktualisierung einer Datenbank, die Sie als Back-End verwenden, zu reduzieren.

Übersicht über das Write-behind-Caching

Beim Write-behind-Caching werden Aktualisierungen für das Loader-Plug-in asynchron in die Warteschlange eingereiht. Sie können die Leistung von Aktualisierungs-, Einfüge- und Entfernungsoperationen für die Map verbessern, indem Sie die eXtreme-Scale-Transaktion von der Datenbanktransaktion entkoppeln. Die asynchrone Aktualisierung wird nach einer zeitbasierten Verzögerung (z. B. fünf Minuten) oder einer eintragsbasierten Verzögerung (z. B. 1000 Einträge) durchgeführt.

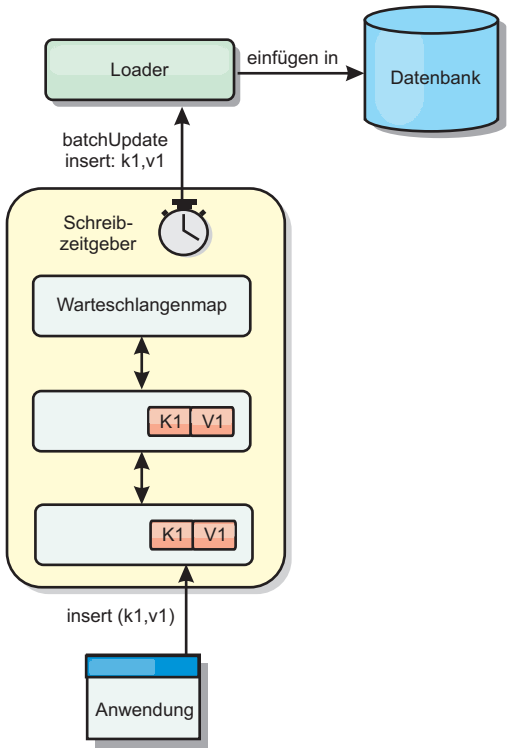


Abbildung 54. Write-behind-Caching

Bei der Write-behind-Konfiguration in einer BackingMap wird ein Thread zwischen dem Loader (Ladeprogramm) und der Map erstellt. Anschließend delegiert der Loader Datenanforderungen über den Thread gemäß den Konfigurationseinstellungen in der Methode "BackingMap.setWriteBehind". Wenn eine eXtreme-Scale-Transaktion einen Eintrag in einer Map einfügt, aktualisiert oder entfernt, wird ein LogElement-Objekt für jeden dieser Datensätze erstellt. Diese Elemente werden an den Write-behind-Loader gesendet und in eine spezielle ObjectMap, eine so genannte Warteschlangenmap, eingereiht. Jede BackingMap mit aktivierter Write-behind-Einstellung hat ihre eigenen Warteschlangenmaps. Ein Write-behind-Thread entfernt die in die Warteschlange eingereihten Daten aus den Warteschlangenmaps und überträgt sie mit Push in den echten Back-End-Loader.

Der Write-behind-Loader sendet nur LogElement-Objekte der Typen "insert" (Einfügen), "update" (Aktualisieren) und "delete" (Löschen) an den echten Loader. Alle anderen Typen von LogElement-Objekten, wie z. B. EVICT, werden ignoriert.

Die Write-behind-Unterstützung ist eine Erweiterung des Loader-Plug-ins, das Sie verwenden, um eXtreme Scale mit der Datenbank zu integrieren. Sehen Sie sich beispielsweise die Informationen zur Konfiguration eines JPA-Loaders im Abschnitt JPA-Loader konfigurieren an.

Vorteile

Das Aktivieren der Write-behind-Unterstützung hat die folgenden Vorteile:

- **Isolation von Back-End-Fehlern:** Durch das Write-behind-Caching können Back-End-Fehler isoliert werden. Wenn die Back-End-Datenbank ausfällt, werden Aktualisierungen in die Warteschlangenmap eingereiht. Die Anwendungen können

weiterhin Transaktionen an eXtreme Scale senden. Nach der Wiederherstellung des Back-Ends werden die Daten in der Warteschlangenmap mit Push an das Back-End übertragen.

- **Geringere Back-End-Last:** Der Write-behind-Loader fasst die Aktualisierungen auf Schlüsselbasis so zusammen, dass nur eine einzige zusammengefasste Aktualisierung pro Schlüssel in der Warteschlangenmap vorhanden ist. Bei dieser Zusammenfassung verringert sich die Anzahl der Aktualisierungen für die Back-End-Datenbank.
- **Verbesserte Transaktionsleistung:** Die Zeiten einzelner eXtreme-Scale-Transaktionen verringern sich, weil sie nicht auf die Synchronisation der Daten mit dem Back-End warten müssen.

Loader

Java

Mit einem Loader-Plug-in kann sich eine Daten-Grid-Map wie ein Speichercache für Daten verhalten, die gewöhnlich in einem persistenten Speicher auf demselben System oder einem anderen System gespeichert werden. Gewöhnlich wird eine Datenbank oder ein Dateisystem als persistenter Speicher verwendet. Es kann auch eine ferne Java Virtual Machine (JVM) als Datenquelle verwendet werden, was die Erstellung hubbasierter Caches mit eXtreme Scale ermöglicht. Ein Loader enthält die Logik für das Lesen aus einem und das Schreiben in einem persistenten Speicher.

Übersicht

Loader (Ladeprogramme) sind BackingMap-Plug-ins, die aufgerufen werden, wenn Änderungen an der BackingMap vorgenommen werden oder wenn die BackingMap eine Datenanforderung nicht bedienen kann (Cachefehler). Der Loader wird aufgerufen, wenn der Cache eine Anforderung für einen Schlüssel nicht bedienen kann. Er unterstützt Read-through-Funktionen und eine verzögerte Füllung des Caches. Ein Loader lässt außerdem Aktualisierungen in der Datenbank zu, wenn sich Cachewerte ändern. Alle Änderungen in einer Transaktion werden gruppiert, um die Anzahl der Datenbankinteraktionen zu minimieren. Zusammen mit dem Loader wird ein TransactionCallback-Plug-in verwendet, um die Abgrenzung der Back-End-Transaktion auszulösen. Die Verwendung dieses Plug-ins ist wichtig, wenn mehrere Maps an einer einzelnen Transaktion beteiligt sind oder wenn Transaktionsdaten ohne Festschreibung mit Flush in den Cache übertragen werden.

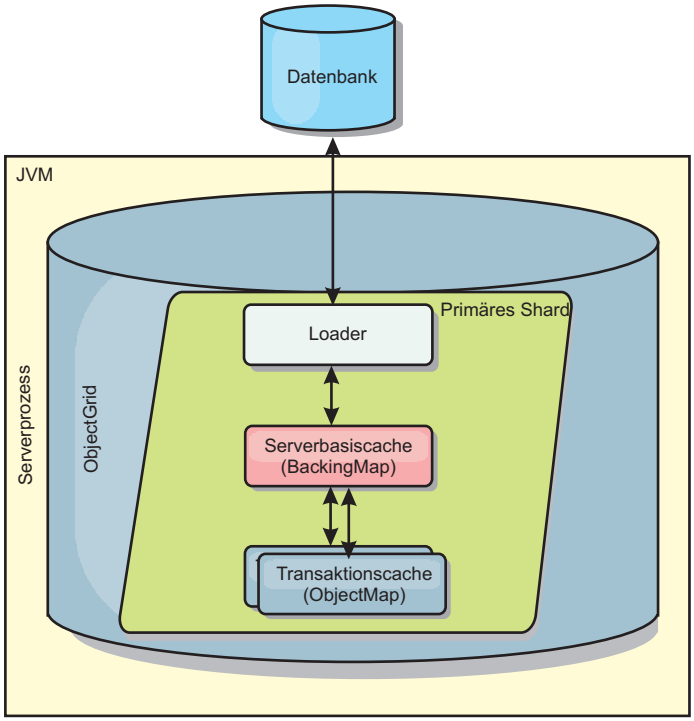


Abbildung 55. Loader

Der Loader kann auch überqualifizierte Aktualisierungen verwenden, um keine Datenbanksperren halten zu müssen. Anhand eines im Cachewert gespeicherten Versionsattributs kann der Loader das Vorher- und Nachher-Abbild des Werts erkennen, wenn dieser im Cache aktualisiert wird. Dieser Wert kann anschließend bei der Aktualisierung der Datenbank bzw. des Back-Ends verwendet werden, um sicherzustellen, dass die Daten nicht aktualisiert wurden. Ein Loader kann auch so konfiguriert werden, dass das Datengrid beim Start vorher geladen wird. Wenn mit Partitionierung gearbeitet wird, wird jeder Partition eine Loader-Instanz zugeordnet. Hat die Map "Company" beispielsweise zehn Partitionen, gibt es zehn Loader-Instanzen, eine für jede primäre Partition. Bei der Aktivierung des primären Shards für die Map wird die Methode "preloadMap" für den Loader synchron oder asynchron aufgerufen. Dies ermöglicht das automatische Laden von Daten aus dem Back-End in die Mappartition. Wenn die Methode synchron aufgerufen wird, werden alle Clienttransaktionen blockiert, um einen inkonsistenten Zugriff auf das Datengrid zu verhindern. Alternativ kann ein Client-Preloader zum Laden des vollständigen Datengrids verwendet werden.

Es gibt zwei integrierte Loader, die die Integration mit relationalen Datenbank-Back-Ends erheblich vereinfachen. Die JPA-Loader nutzen die ORM-Funktionen (Object-Relational Mapping, objektrelationale Abbildung) der OpenJPA- und Hibernate-Implementierungen der Spezifikation Java Persistence API (JPA). Weitere Informationen finden Sie unter „JPA-Loader” auf Seite 75.

Wenn Sie Loader in einer Konfiguration mit mehreren Rechenzentren verwenden, müssen Sie berücksichtigen, wie Revisionsinformationen und Cachekonsistenz zwischen den Datengrids verwaltet werden. Weitere Informationen finden Sie im Abschnitt „Hinweise zu Ladeprogrammen in einer Multimastertopologie” auf Seite 183.

Loader-Konfiguration

Wenn Sie der BackingMap-Konfiguration einen Loader hinzufügen möchten, können Sie die programmgesteuerte Konfiguration oder die XML-Konfiguration verwenden. Ein Loader steht mit einer BackingMap in folgender Beziehung.

- Eine BackingMap kann nur einen einzigen Loader haben.
- Eine Client-BackingMap (naher Cache) kann keinen Loader haben.
- Eine Loader-Definition kann auf mehrere BackingMaps angewendet werden, aber jede BackingMap hat eine eigene Loader-Instanz.

Vorheriges Laden von Daten und Vorbereitung

In vielen Szenarien, die die Verwendung eines Loaders (Ladeprogramms) beinhalten, können Sie Ihr Datengrid durch vorheriges Laden von Daten (Preload) vorbereiten.

Wenn das Grid als vollständiger Cache verwendet wird, muss das Datengrid alle Daten aufnehmen und geladen werden, bevor Clients eine Verbindung zum Grid herstellen können. Wenn Sie einen Teilcache verwenden, müssen Sie den Cache mit Daten vorbereiten (Aufwärmphase), sodass Clients sofortigen Zugriff auf die Daten haben, wenn sie eine Verbindung zum Grid herstellen.

Es gibt zwei Methoden für das vorherige Laden von Daten in das Datengrid: Verwendung eines Loader-Plug-ins (Ladeprogramm) oder Verwendung eines Client-Loaders. Diese beiden Methoden werden in den folgenden Abschnitten beschrieben.

Loader-Plug-in

Das Loader-Plug-in wird jeder Map zugeordnet und ist für die Synchronisation eines einzelnen primären Partitions-Shards mit der Datenbank zuständig. Die Methode "preloadMap" des Loader-Plug-ins wird automatisch aufgerufen, wenn ein Shard aktiviert wird. Wenn Sie beispielsweise 100 Partitionen haben, sind 100 Loader-Instanzen vorhanden, die jeweils die Daten für ihre Partition laden. Wenn die Loader-Instanzen synchron ausgeführt werden, werden alle Clients blockiert, bis das vorherige Laden der Daten (der so genannte Preload-Prozess) abgeschlossen ist.

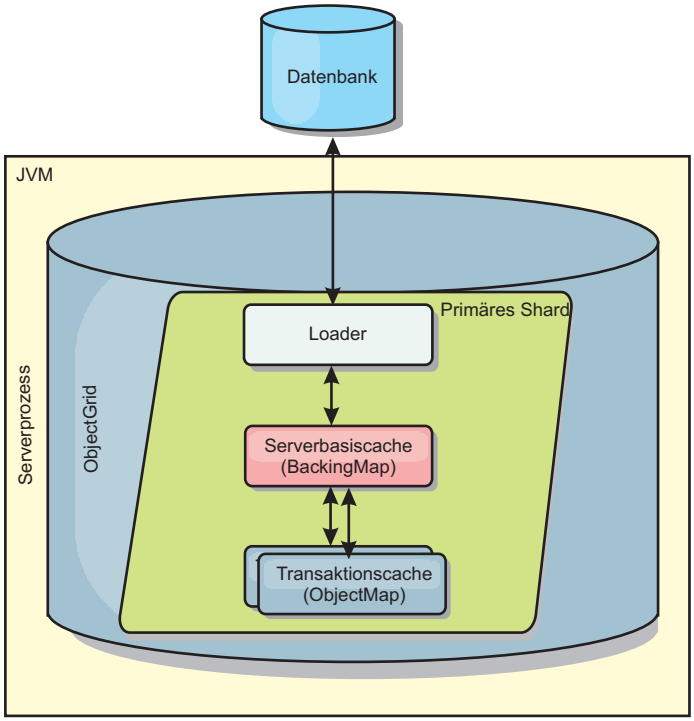


Abbildung 56. Loader-Plug-in

Client-Loader

Ein Client-Loader ist ein Muster für die Verwendung eines oder mehrerer Clients, um Daten in das Grid zu laden. Die Verwendung mehrerer Clients zum Laden von Griddaten kann effektiv sein, wenn das Partitionsschema nicht in der Datenbank gespeichert ist. Sie können Client-Loader manuell oder automatisch aufrufen, wenn das Datengrid gestartet wird. Client-Loader können optional die Schnittstelle "StateManager" verwenden, um den Status des Datengrids auf den Preload-Modus zu setzen, sodass Clients nicht auf das Grid zugreifen können, wenn das vorherige Laden der Daten in das Grid durchgeführt wird. WebSphere eXtreme Scale enthält einen JPA-basierten (Java Persistence API) Loader, den Sie verwenden können, um das Datengrid automatisch über die OpenJPA- oder Hibernate-JPA-Provider zu laden. Weitere Informationen zu Cache-Providern finden Sie unter „JPA-L2-Cache-Plug-in” auf Seite 42.

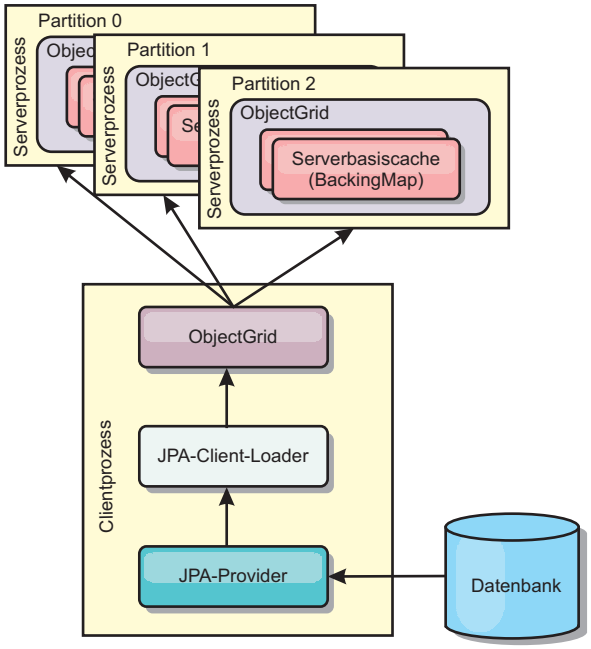


Abbildung 57. Client-Loader

Verfahren für die Datenbanksynchronisation

Wenn WebSphere eXtreme Scale als Cache verwendet wird, müssen Anwendungen so geschrieben werden, dass veraltete Daten toleriert werden, wenn die Datenbank unabhängig von einer eXtreme-Scale-Transaktion aktualisiert werden kann. Für den Einsatz als Verarbeitungsbereich für die synchronisierte speicherinterne Datenbank stellt eXtreme Scale mehrere Methoden für die konstante Aktualisierung des Caches bereit.

Verfahren für die Datenbanksynchronisation

Regelmäßige Aktualisierung

Der Cache kann mit Hilfe der zeitbasierten JPA-Datenbankaktualisierungskomponente (Java Persistence API) automatisch ungültig gemacht oder regelmäßig aktualisiert werden. Die Aktualisierungskomponente fragt die Datenbank in regelmäßigen Abständen über einen JPA-Provider nach Aktualisierungen oder Einfügungen ab, die seit der vorherigen Aktualisierung vorgenommen wurden. Alle gefundenen Änderungen werden automatisch ungültig gemacht oder aktualisiert, wenn ein Teilcache verwendet wird. Wenn ein vollständiger Cache verwendet wird, können die Einträge erkannt und in den Cache eingefügt werden. Es werden keine Einträge aus dem Cache entfernt.

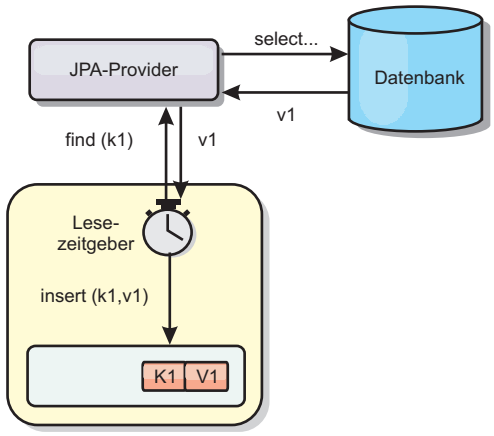


Abbildung 58. Regelmäßige Aktualisierung

Bereinigung

Teilcaches können Bereinigungsrichtlinien verwenden, um Daten ohne Beeinträchtigung der Datenbank automatisch aus dem Cache zu entfernen. Mit eXtreme Scale werden drei integrierte Richtlinien bereitgestellt: Lebensdauer (TTL, Time-to-Live), LRU (least recently used) und LFU (last frequently used). Alle drei Richtlinien können Daten aggressiver entfernen, wenn Speicherengpässe auftreten, indem die Option für speicherbasierte Bereinigung aktiviert wird.

Ereignisbasierte Invalidierung

Teilcaches und vollständige Caches können mit Hilfe eines Ereignisgenerators wie Java Message Service (JMS) ungültig gemacht oder aktualisiert werden. Die Invalidierung mit JMS kann manuell an jeden Prozess gebunden werden, der das Back-End über einen Datenbankauslöser aktualisiert. Es wird ein JMS-ObjectGridEvent-Listener-Plug-in in eXtreme Scale bereitgestellt, das Clients benachrichtigen kann, wenn Änderungen im Server-Cache vorgenommen wurden. Auf diese Weise kann das Zeitfenster, in dem der Client veraltete Daten sieht, verringert werden.

Programmgesteuerte Invalidierung

Die APIs von eXtreme Scale unterstützen die manuelle Interaktion zwischen nahem Cache und Server-Cache über die API-Methoden "Session.beginNoWriteThrough()", "ObjectMap.invalidate()" und "EntityManager.invalidate()". Wenn ein Client- oder Serverprozess einen Teil der Daten nicht mehr benötigt, können Sie mit den Invalidierungsmethoden Daten aus dem nahen Cache bzw. Server-Cache entfernen. Die Methode "beginNoWriteThrough" gilt für alle ObjectMap- und EntityManager-Operationen im lokalen Cache ohne Aufruf des Loaders. Wenn die Methode von einem Client aufgerufen wird, gilt die Operation nur für den nahen Cache (der ferne Loader wird nicht aufgerufen). Wird die Methode im Server aufgerufen, gilt die Operation nur für den Serverbasiscache ohne Aufruf des Loaders.

Dateninvalidierung

Sie können Invalidierungsmechanismen verwenden, um veraltete Cachedaten zu entfernen.

Administrative Invalidierung

Sie können die Webkonsole oder das Dienstprogramm **xscmd** verwenden, um Daten basierend auf dem Schlüssel ungültig zu machen. Sie können die Cachedaten mit einem regulären Ausdruck filtern und die Daten dann auf der Basis des regulären Ausdrucks ungültig machen.

Ereignisgesteuerte Invalidierung

Teilcaches und vollständige Caches können mit Hilfe eines Ereignisgenerators wie Java Message Service (JMS) ungültig gemacht oder aktualisiert werden. Die Invalidierung mit JMS kann manuell an jeden Prozess gebunden werden, der das Back-End über einen Datenbankauslöser aktualisiert. Es wird ein JMS-ObjectGridEvent-Listener-Plug-in in eXtreme Scale bereitgestellt, das Clients benachrichtigen kann, wenn Änderungen im Server-Cache vorgenommen wurden. Dieser Typ von Benachrichtigung verkleinert das Zeitfenster, in dem der Client veraltete Daten sieht.

Der ereignisgesteuerte Invalidierungsmechanismus setzt sich gewöhnlich aus den folgenden drei Komponenten zusammen:

- **Ereigniswarteschlange:** In einer Ereigniswarteschlange werden die Datenänderungsereignisse gespeichert. Die Ereigniswarteschlange kann eine JMS-Warteschlange, eine Datenbank, eine speicherinterne FIFO-Warteschlange oder ein beliebiges Manifest sein, das Datenänderungsereignisse verwalten kann.
- **Ereignis-Publisher:** Ein Ereignis-Publisher veröffentlicht die Datenänderungsereignisse in der Ereigniswarteschlange. Ein Ereignis-Publisher ist gewöhnlich eine Anwendung, die Sie erstellen, oder eine Implementierung eines eXtreme-Scale-Plug-ins. Der Ereignis-Publisher weiß, wann die Daten geändert werden, oder ändert die Daten selbst. Wenn eine Transaktion festgeschrieben wird, werden Ereignisse für die geänderten Daten generiert, und der Ereignis-Publisher veröffentlicht diese Ereignisse in der Ereigniswarteschlange.
- **Ereigniskonsument:** Ein Ereigniskonsument konsumiert Datenänderungsereignisse. Der Ereigniskonsument ist gewöhnlich eine Anwendung, die sicherstellt, dass die Daten im Zielgrid mit den neuesten Änderungen aus anderen Grids aktualisiert werden. Dieser Ereigniskonsument interagiert mit der Ereigniswarteschlange, um die neuesten Datenänderungen abzurufen, und wendet die Datenänderungen auf das Zielgrid an. Die Ereigniskonsumenten können APIs von eXtreme Scale verwenden, um veraltete Daten ungültig zu machen oder um das Grid mit den neuesten Daten zu aktualisieren.

JMSObjectGridEventListener hat beispielsweise eine Option für ein Client/Server-Modell, bei der die Ereigniswarteschlange eine festgelegte JMS-Destination ist. Alle Serverprozesse sind Ereignis-Publisher. Wenn eine Transaktion festgeschrieben wird, ruft der Server die Datenänderungen ab und veröffentlicht sie in der festgelegten JMS-Destination. Alle Clientprozesse sind Ereigniskonsumenten. Sie empfangen Datenänderungen von der festgelegten JMS-Destination und wenden die Änderungen auf den nahen Cache des Clients an.

Weitere Informationen finden Sie unter JMS-basierte Clientsynchronisation konfigurieren.

Programmgesteuerte Invalidierung

Die APIs von WebSphere eXtreme Scale unterstützen die manuelle Interaktion zwischen nahem Cache und Server-Cache über die API-Methoden "Session.beginNoWriteThrough()", "ObjectMap.invalidate()" und "EntityManager.invalidate()".

Wenn ein Client- oder Serverprozess einen Teil der Daten nicht mehr benötigt, können Sie mit den Invalidierungsmethoden Daten aus dem nahen Cache bzw. Server-Cache entfernen. Die Methode "beginNoWriteThrough" gilt für alle ObjectMap- und EntityManager-Operationen im lokalen Cache ohne Aufruf des Loaders. Wenn die Methode von einem Client aufgerufen wird, gilt die Operation nur für den nahen Cache (der ferne Loader wird nicht aufgerufen). Wird die Methode im Server aufgerufen, gilt die Operation nur für den Serverbasiscache ohne Aufruf des Loaders.

Sie können die programmgesteuerte Invalidierung zusammen mit anderen Techniken verwenden, um festzustellen, wann die Daten ungültig gemacht werden müssen. Diese Invalidierungsmethode verwendet beispielseise ereignisgesteuerte Invalidierungsmechanismen, um die Datenänderungsereignisse zu empfangen, und anschließend APIs, um die veralteten Daten ungültig zu machen.

8.6+
Invalidierung des nahen Caches

Wenn Sie einen nahen Cache verwenden, können Sie eine asynchrone Invalidierung konfigurieren, die jedes Mal ausgelöst wird, wenn eine Aktualisierungs-, Lösch- oder Invalidierungsoperation im Datengrid ausgeführt wird. Da die Operation asynchron ist, können Sie trotzdem veraltete Daten im Datengrid sehen.

Zum Aktivieren der Invalidierung des nahen Caches setzen Sie das Attribut **`nearCacheInvalidationEnabled`** in der BackingMap in der ObjectGrid-XML-Deskriptordatei.

Indexierung

Java

Verwenden Sie das Plug-in "MapIndexPlugin", um einen Index oder mehrere Indizes in einer BackingMap für die Unterstützung von Datenzugriffen ohne Schlüssel zu erstellen.

Indextypen und Konfiguration

Das Indexierungsfeature wird durch das Plug-in "MapIndexPlugin" oder kurz "Index" dargestellt. Index ist ein BackingMap-Plug-in. Für eine BackingMap können mehrere Index-Plug-ins konfiguriert werden, solange jedes Plug-in den Index-Konfigurationsregeln entspricht.

Sie können das Indexierungsfeature verwenden, um einen oder mehrere Indizes in einer BackingMap zu erstellen. Ein Index wird aus einem Attribut oder einer Liste von Attributen eines Objekts in der BackingMap erstellt. Das Feature bietet Anwendungen eine Möglichkeit, bestimmte Objekte schneller zu finden. Mit dem Indexierungsfeature können Anwendungen mit einem bestimmten Wert oder innerhalb eines bestimmten Wertebereichs indexierter Attribute finden.

Es gibt zwei Typen von Indexierung: statische Indexierung und dynamische Indexierung. Bei der statischen Indexierung müssen Sie das Index-Plug-in in der BackingMap konfigurieren, bevor Sie die ObjectGrid-Instanz initialisieren. Sie können diese Konfiguration durch XML- oder programmgesteuerte Konfiguration der BackingMap vornehmen. Die statische Indexierung beginnt mit der Erstellung eines Index während der ObjectGrid-Initialisierung. Der Index ist immer mit der BackingMap synchronisiert und zur Verwendung bereit. Nach dem Start des statischen Indexierungsprozesses erfolgt die Verwaltung des Index im Rahmen des

Transaktionsverwaltungsprozesses von eXtreme Scale. Wenn Transaktionen Änderungen festschreiben, werden diese Änderungen auch im statischen Index durchgeführt, und Indexänderungen werden rückgängig gemacht, wenn die Transaktion rückgängig gemacht wird.

Bei der dynamischen Indexierung können Sie einen Index in einer BackingMap vor oder nach der Initialisierung der übergeordneten ObjectGrid-Instanz erstellen. Anwendungen haben eine Lebenszykluskontrolle über den dynamischen Indexierungsprozess, d. h., Sie können einen dynamischen Index entfernen, wenn er nicht mehr benötigt wird. Wenn eine Anwendung einen dynamischen Index erstellt, ist der Index möglicherweise nicht zur sofortigen Verwendung bereit, weil die Erstellung des Index eine gewisse Zeit dauert. Da die Erstellungsdauer vom Volumen der zu indexierenden Daten abhängig ist, wird die Schnittstelle "DynamicIndexCallback" für Anwendungen bereitgestellt, die Benachrichtigungen empfangen möchten, wenn bestimmte Indexierungsereignisse eintreten. Zu diesen Ereignissen gehören die Bereitschaft des Index (ready), Fehler (error) und das Löschen des Index (destroy). Anwendungen können diese Callback-Schnittstelle implementieren und sich beim dynamischen Indexierungsprozess registrieren.

8.6+ Wenn eine BackingMap ein konfiguriertes Index-Plug-in hat, können Sie das Proxy-Objekt für den Anwendungsindex von der entsprechenden ObjectMap abrufen. Wenn Sie die Methode getIndex in der Schnittstelle "ObjectMap" aufrufen und den Namen des Index-Plug-ins übergeben, wird das Index-Proxy-Objekt zurückgegeben. Sie müssen das Index-Proxy-Objekt in die entsprechende Anwendungsindexschnittstelle, z. B. MapIndex, MapRangeIndex, MapGlobalIndex oder eine angepasste Indexschnittstelle, umsetzen. Nach dem Abrufen des Index-Proxy-Objekts können Sie in der Anwendungsindexschnittstelle definierte Methoden verwenden, um zwischengespeicherte Objekte zu suchen.

Die Schritte zur Verwendung der Indexierung sind in der folgenden Liste zusammengefasst:

- Fügen Sie statische oder dynamische Index-Plug-ins in der BackingMap hinzu.
- Rufen Sie mit der Methode "getIndex" von ObjectMap ein Proxy-Objekt für den Anwendungsindex ab.
- Setzen Sie das Proxy-Objekt für den Index in eine entsprechende Anwendungsindexschnittstelle um, wie z. B. MapIndex, MapRangeIndex oder eine angepasste Indexschnittstelle.
- Verwenden Sie die in der Anwendungsindexschnittstelle definierten Methoden, um zwischengespeicherte Objekte zu suchen.

8.6+ Die Klasse "HashIndex" ist die integrierte Index-Plug-in-Implementierung, die die folgenden integrierten Anwendungsindexschnittstellen unterstützen kann:

- MapIndex
- MapRangeIndex
- MapGlobalIndex

Sie können auch eigene Indizes erstellen. Sie können HashIndex als statischen oder dynamischen Index in der BackingMap hinzufügen, ein MapIndex-, MapRangeIndex- oder MapGlobalIndex-Index-Proxy-Objekt abrufen und das Index-Proxy-Objekt zum Suchen zwischengespeicherter Objekte verwenden.

8.6+

Globaler Index

Der globale Index ist eine Erweiterung der integrierten Klasse "HashIndex", die auf Shards in verteilten, partitionierten Datengridumgebungen ausgeführt wird. Der globale Index überwacht die Position indexierter Attribute und bietet effiziente Möglichkeiten, um Partitionen, Schlüssel, Werte oder Einträge anhand von Attributen in großen, partitionierten Datengridumgebungen zu finden.

Wenn der globale Index im integrierten HashIndex-Plug-in aktiviert ist, können Anwendungen ein Index-Proxy-Objekt in den Typ "MapGlobalIndex" umsetzen und diesen zum Suchen von Daten verwenden.

Standardindex

Wenn Sie durch die Schlüssel in einer lokalen Map iterieren möchten, können Sie den Standardindex verwenden. Dieser Index erfordert keine Konfiguration, aber er muss für das Shard über einen Agenten oder eine ObjectGrid-Instanz, die mit der Methode ShardEvents.shardActivated(ObjectGrid shard) abgerufen wird, verwendet werden.

Hinweis zur Datenqualität

Die Ergebnisse der Indexabfragemethoden stellen nur eine Momentaufnahme der Daten zu einem bestimmten Zeitpunkt dar. Es werden keine Sperren für Dateneinträge angefordert, nachdem die Ergebnisse an die Anwendung zurückgegeben wurden. Die Anwendung muss sich darüber im Klaren sein, dass Datenaktualisierungen für eine zurückgegebene Datengruppe vorgenommen werden können. Beispiel: Die Anwendung ruft den Schlüssel eines zwischengespeicherten Objekts mit der Methode findAll von MapIndex ab. Dieses zurückgegebene Schlüsselobjekt ist einem Dateneintrag im Cache zugeordnet. Die Anwendung muss in der Lage sein, die Methode "get" in ObjectMap auszuführen, um ein Objekt durch Übergabe des Schüsselobjekts zu suchen. Wenn eine andere Transaktion das Datenobjekt aus dem Cache entfernt, kurz bevor die Methode "get" aufgerufen wird, ist das zurückgegebene Ergebnis null.

Hinweise zur Leistung der Indexierung

Eine der Hauptzielsetzungen des Indexierungsfeatures ist die Verbesserung der Gesamtleistung der BackingMap. Wenn die Indexierung nicht ordnungsgemäß verwendet wird, kann dies die Leistung der Anwendung beeinträchtigen. Berücksichtigen Sie vor der Verwendung dieses Features die folgenden Faktoren.

- **Anzahl gleichzeitiger Transaktionen mit Schreibzugriff:** Die Indexverarbeitung kann jedesmal stattfinden, wenn eine Transaktion Daten in eine BackingMap schreibt. Schreiben viele Transaktionen gleichzeitig Daten in die Map, kann es zu Leistungseinbußen kommen, wenn eine Anwendung versucht, Indexabfrageoperationen durchzuführen.
- **Größe der von einer Abfrageoperation zurückgegebenen Ergebnismenge:** Je größer die Ergebnismenge wird, desto mehr nimmt die Abfrageleistung ab. Ab einer Ergebnismengengröße von 15 % der Gesamtgröße der BackingMap beginnen sich Leistungseinbußen abzuzeichnen.
- **Anzahl der für dieselbe BackingMap erstellten Indizes:** Jeder Index belegt Systemressourcen. Mit steigender Indexanzahl für die BackingMap nimmt die Leistung ab.

Die Indexierungsfunktion kann die Leistung einer BackingMap erheblich verbessern. Die besten Ergebnisse lassen sich erzielen, wenn hauptsächlich Leseoperationen für die BackingMap durchgeführt werden, wenn die Abfrageergebnismenge nur einen kleinen Prozentsatz der BackingMap-Einträge enthält und wenn nur einige wenige Indizes für die BackingMap erstellt werden.

Topologien mit mehreren Rechenzentren planen

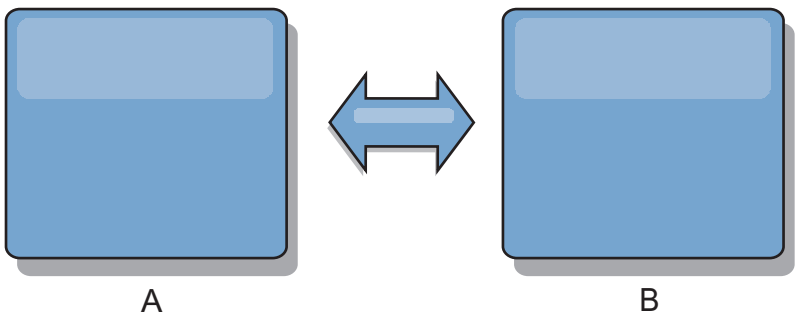
Wenn Sie eine asynchrone Multimasterreplikation verwenden, können zwei oder mehr Datengrids exakte Kopien voneinander werden. Jedes Datengrid ist in einer unabhängigen Katalogservicedomäne mit einem eigenen Katalogservice, eigenen Container-Servern und einem eindeutigen Namen enthalten. Bei asynchroner Multimasterreplikation können Sie Verbindungen verwenden, um eine Sammlung von Katalogservicedomänen zu verbinden. Die Katalogservicedomänen werden anschließend durch Replikation über die Verbindungen synchronisiert. Sie können fast jede Topologie durch die Definition von Verbindungen zwischen den Katalogservicedomänen erstellen.

Topologien für Multimasterreplikation

Sie haben verschiedene Optionen bei der Auswahl der Topologie für Ihre Umgebung mit Multimasterreplikation.

Verknüpfungen zu Katalogservicedomänen

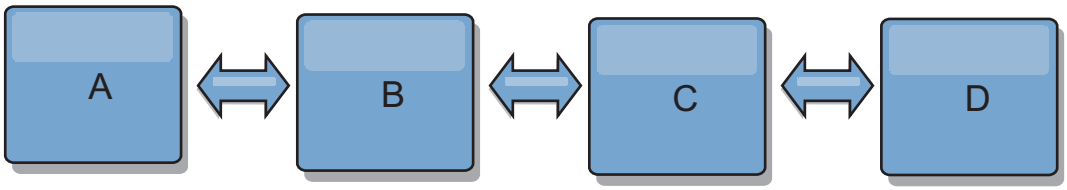
Eine Replikationsdatengridinfrastruktur ist ein verbundener Graph von Katalogservicedomänen mit bidirektionalen Verbindungen zwischen den Domänen. Über eine Verbindung können zwei Katalogservicedomänen Datenänderungen austauschen. Die einfachste Topologie ist beispielsweise ein Paar von Katalogservicedomänen mit einer einzigen Verbindung zwischen ihnen. Die Katalogservicedomänen werden alphabetisch von links nach rechts benannt: A, B, C usw. Eine Verbindung kann ein Weitverkehrsnetz (WAN) durchqueren und große Distanzen überwinden. Selbst wenn die Verbindung unterbrochen wird, können Daten in jeder Katalogservicedomäne trotzdem geändert werden. Die Topologie gleicht die Änderungen ab, sobald die Verbindung die Katalogservicedomänen wieder verbindet. Verbindungen versuchen nach der Unterbrechung der Netzverbindung automatisch, die Verbindung wiederherzustellen.



Nachdem Sie die Verbindungen konfiguriert haben, versucht das Produkt zuerst, alle Katalogservicedomäne zu synchronisieren. Anschließend versucht eXtreme Scale, die identischen Zustände aufrecht zu erhalten, wenn Änderungen in einer Katalogservicedomäne vorgenommen werden. Das Ziel ist, dass jede Katalogservicedomäne ein exakter Spiegel jeder anderen Katalogservicedomäne ist, mit der sie verbunden ist. Durch die Replikationsverbindungen zwischen den Katalogservicedomänen wird sichergestellt, dass alle Änderungen, die in einer Katalogservicedomäne vorgenommen werden, in die anderen Katalogservicedomänen kopiert werden.

Reihentopologien

Obwohl es sich um eine sehr einfache Implementierung handelt, veranschaulicht die Reihentopologie einige Qualitäten der Verbindungen. Zunächst ist es nicht erforderlich, dass eine Katalogservicedomäne direkt mit jeder anderen Katalogservicedomäne verbunden ist, damit sie Änderungen empfängt. Die Katalogservicedomäne B extrahiert Änderungen aus Katalogservicedomäne A. Die Katalogservicedomäne C empfängt Änderungen von Katalogservicedomäne A über Katalogservicedomäne B, die Katalogservicedomänen A und C miteinander verbindet. Katalogservicedomäne D empfängt Änderungen von den anderen Katalogservicedomänen über Katalogservicedomäne C. Diese Funktionalität verlagert die Last der Verteilung von Änderungen von der Änderungsquelle.



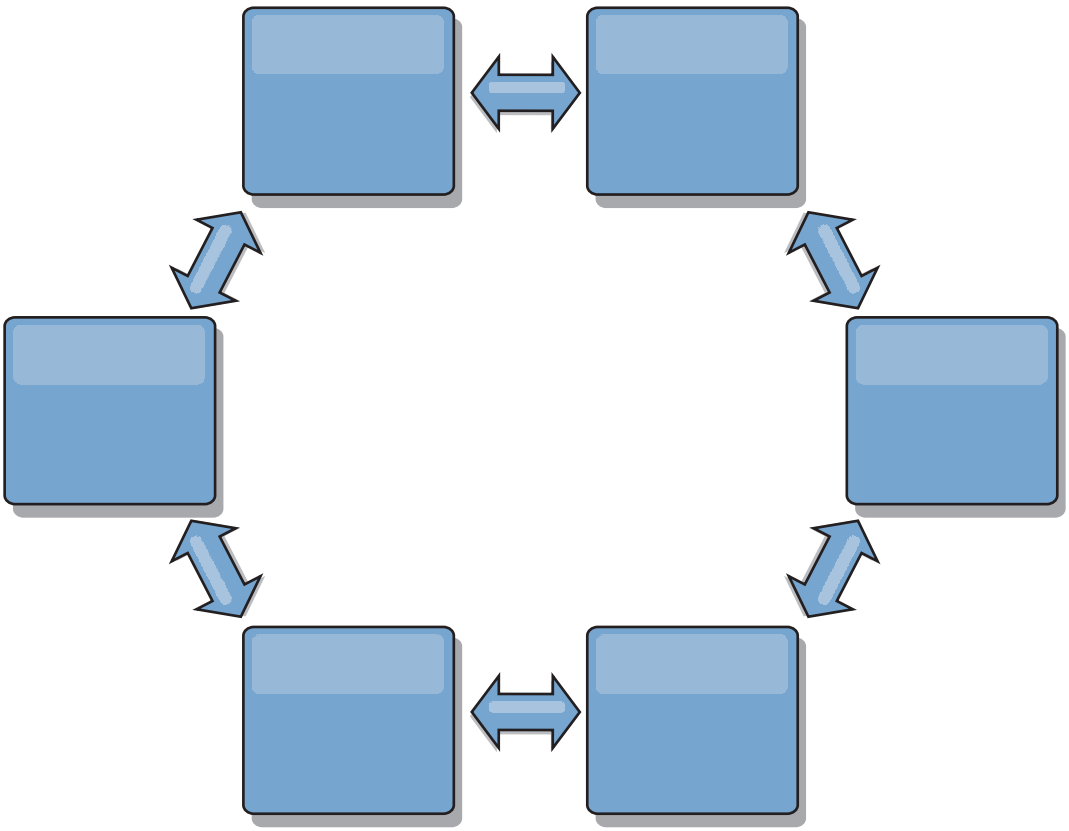
Wenn Katalogservicedomäne C ausfällt, werden die folgenden Aktionen ausgeführt:

1. Katalogservicedomäne D ist verwaist, bis Katalogservicedomäne C erneut gestartet wird.
2. Katalogservicedomäne C synchronisiert sich selbst mit Katalogservicedomäne B, die eine Kopie von Katalogservicedomäne A ist.
3. Katalogservicedomäne D verwendet Katalogservicedomäne C für die Synchronisation der Änderungen mit den Katalogservicedomänen A und B. Diese Änderungen sind eingetreten, als Katalogservicedomäne D verwaist war (als Katalogservicedomäne C ausgefallen ist).

Am Ende sind die Katalogservicedomänen A, B, C und D wieder identisch.

Ringtopologien

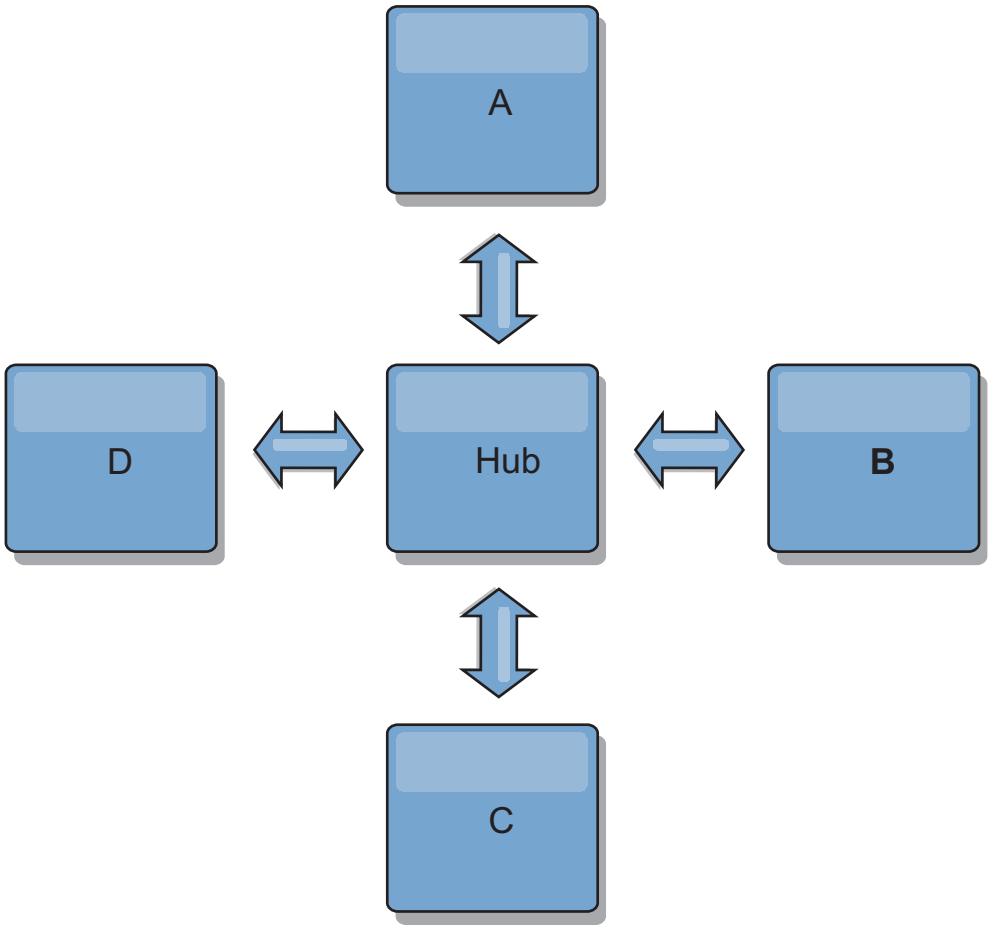
Ringtopologien sind ein Beispiel für eine Topologie mit erhöhter Ausfallsicherheit. Wenn eine Katalogservicedomäne oder eine einzelne Verbindung ausfällt, können die verbleibenden Katalogservicedomänen trotzdem Änderungen abrufen. Die Katalogservicedomänen bewegen sich ringförmig vom Ausfall weg. Jede Katalogservicedomäne hat maximal zwei Verbindungen zu anderen Katalogservicedomänen, unabhängig davon, wie große die Ringtopologie ist. Die Latenzzeit für die Weitergabe der Änderungen kann hoch sein. Die Änderungen einer bestimmten Katalogservicedomäne müssen möglicherweise über mehrere Verbindungen übertragen werden, bevor sie in allen Katalogservicedomänen vorhanden sind. Eine Reihentopologie weist dasselbe Merkmal auf.



Sie können auch eine fortgeschrittenere Ringtopologie mit einer Stammkatalogservicedomäne in der Mitte des Rings implementieren. Die Stammkatalogservicedomäne dient als zentraler Abgleichspunkt. Die anderen Katalogservicedomänen dienen als ferne Abgleichspunkte für Änderungen, die in der Stammkatalogservicedomäne vorgenommen werden. Die Stammkatalogservicedomäne kann Änderungen zwischen den Katalogservicedomänen arbitrieren. Wenn eine Ringtopologie mehrere Ringe um eine Stammkatalogservicedomäne herum enthält, kann die Katalogservicedomäne Änderungen nur im inneren Ring arbitrieren. Die Ergebnisse der Arbitrierung werden jedoch über die Katalogservicedomänen in den anderen Ringen verteilt.

Hub- und Peripherietopologien

Mit einer Hub- und Peripherietopologie werden Änderungen über eine Hubkatalogservicedomäne übertragen. Weil der Hub die einzige angegebene zwischengeschaltete Katalogservicedomäne ist, haben Hub- und Peripherietopologien geringere Latenzzeiten. Die Hubkatalogservicedomäne wird über eine Verbindung mit jeder Peripheriekatalogservicedomäne verbunden. Der Hub verteilt Änderungen an die Katalogservicedomänen. Der Hub dient als Abgleichspunkt für Kollisionen. In einer Umgebung mit einer hohen Aktualisierungsrate, muss der Hub im Hinblick auf die Synchronizität möglicherweise auf mehr Hardware als die Peripherie ausgeführt werden. WebSphere eXtreme Scale ist für eine lineare Skalierung konzipiert, d. h., Sie können den Hub bei Bedarf ohne Schwierigkeit vergrößern. Wenn der Hub jedoch ausfällt, werden die Änderungen erst nach einem Neustart des Hubs wieder verteilt. Alle Änderungen in den Peripheriekatalogservicedomänen werden verteilt, nachdem die Hubverbindung wiederhergestellt ist.



Sie können auch eine Strategie mit vollständig replizierten Clients verwenden, eine Topologievariante, in der ein Serverpaar als Hub verwendet wird. Jeder Client erstellt ein eigenständiges Einzelcontainerdatengrid mit einem Katalog in der Client-JVM. Ein Client verwendet sein Datengrid, um die Verbindung zum Hubkatalog herzustellen. Die Verbindung bewirkt, dass sich der Client mit dem Hub synchronisiert, sobald die Verbindung zum Hub hergestellt ist.

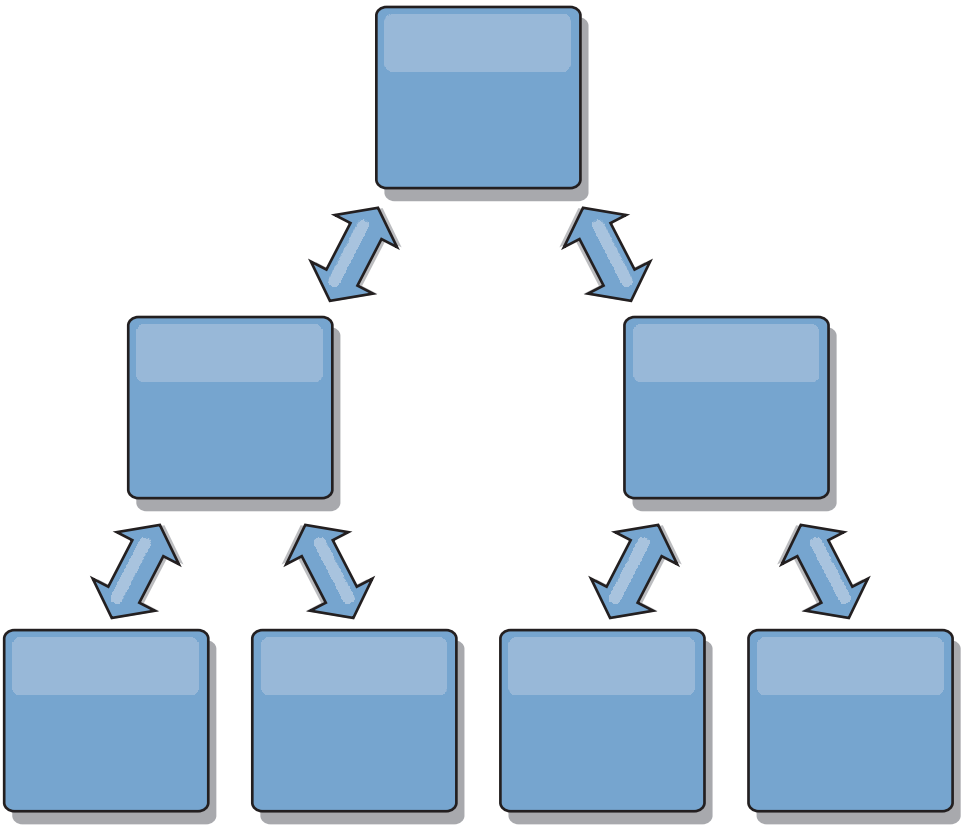
Alle vom Client vorgenommenen Änderungen sind lokal und werden asynchron im Hub repliziert. Der Hub dient als Arbitrierungskatalogservicedomäne und verteilt Änderungen an alle verbundenen Clients. Die Topologie mit vollständig replizierten Clients ist ein zuverlässiger L2-Cache für einen objektrelationalen Mapper wie OpenJPA. Änderungen werden über den Hub schnell an die Client-JVMs verteilt. Solange die Cachegröße im verfügbaren Heapspeicher untergebracht werden kann, ist die Topologie eine geeignete Architektur für diesen L2-Stil.

Verwenden Sie bei Bedarf mehrere Partitionen für die Skalierung der Hubkatalogservicedomäne in mehreren JVMs. Weil alle Daten immer noch in eine einzige Client-JVM passen müssen, kann die Kapazität des Hubs für die Verteilung und Arbitrierung von Änderungen durch mehrere Partitionen erhöht werden. Die Verwendung mehrerer Partitionen ändert die Kapazität einer einzelnen Katalogservicedomäne jedoch nicht.

Baumtopologien

Sie können auch eine azyklische gerichtete Baumstruktur verwenden. Eine azyklische Baumstruktur hat keine Zyklen oder Schleifen, und ein gerichtetes Setup be-

schränkt Verbindungen auf vorhandene Verbindungen zwischen übergeordneten und untergeordneten Komponenten. Diese Konfiguration ist hilfreich für Topologien mit vielen Katalogservicedomänen. In diesen Topologien ist es nicht empfehlenswert, einen zentralen Hub zu verwenden, der mit jedem möglichen Peripheriegerät verbunden ist. Dieser Typ von Topologie kann auch hilfreich sein, wenn Sie untergeordnete Katalogservicedomänen hinzufügen müssen, ohne die Stammkatalogservicedomäne zu aktualisieren.



Eine Baumstrukturtopologie kann einen zentralen Abgleichspunkt in der Stammkatalogservicedomäne haben. Die zweite Ebene kann weiterhin als ferner Abgleichspunkt für Änderungen dienen, die in der darunter liegenden Katalogservicedomäne vorgenommen werden. Die Stammkatalogservicedomäne kann Änderungen zwischen den Katalogservicedomänen nur auf der zweiten Ebene arbitrieren. Sie können auch k-näre Baumstrukturen verwenden, die jeweils N untergeordnete Komponenten auf jeder Ebene haben können. Jede Katalogservicedomäne stellt *n* ausgehende Verbindungen her.

Vollständig replizierte Clients

Diese Topologievariante beinhaltet ein Serverpaar, das als Hub ausgeführt wird. Jeder Client erstellt ein eigenständiges Einzelcontainer-Datengrid mit einem Katalog in der Client-JVM. Ein Client verwendet sein Datengrid, um die Verbindung zum Hubkatalog herzustellen, was bewirkt, dass sich der Client mit dem Hub synchronisiert, sobald die Verbindung zum Hub hergestellt ist.

Alle vom Client vorgenommenen Änderungen sind lokal und werden asynchron im Hub repliziert. Der Hub dient als Arbitrierungskatalogservicedomäne und verteilt Änderungen an alle verbundenen Clients. Die Topologie mit vollständig replizierten Clients ist ein guter L2-Cache für einen objektrelationalen Mapper wie

OpenJPA. Änderungen werden über den Hub schnell an die Client-JVMs verteilt. Solange die Cachegröße vom verfügbaren Heapspeicher der Clients untergebracht werden kann, ist diese Topologie eine geeignete Architektur für diesen L2-Stil.

Verwenden Sie bei Bedarf mehrere Partitionen für die Skalierung der Hubkatalogservicedomäne in mehreren JVMs. Da alle Daten weiterhin in eine einzige Client-JVM passen müssen, erhöht die Verwendung mehrerer Partitionen die Kapazität des Hubs für die Verteilung und Arbitrierung von Änderungen, aber nicht die Kapazität einer einzelnen Katalogservicedomäne.

Konfigurationshinweise für Multimastertopologien

Beachten Sie die folgenden Probleme, wenn Sie festlegen, ob und wie Multimasterreplikationstopologien verwendet werden.

- **Voraussetzungen für MapSets**

  MapSets müssen die folgenden Merkmale aufweisen, damit Änderungen über Verbindungen zwischen Katalogservicedomänen repliziert werden können:

  – Der ObjectGrid-Name und der MapSet-Name in einer Katalogservicedomäne müssen mit dem ObjectGrid-Namen und dem MapSet-Namen anderer Katalogservicedomänen übereinstimmen. ObjectGrid "og1" und MapSet "ms1" müssen beispielsweise in Katalogservicedomäne A und Katalogservicedomäne B konfiguriert werden, um die Daten im MapSet zwischen den Katalogservicedomänen zu replizieren.

  – Das Datengrid hat den Typ `FIXED_PARTITION`. Datengrids des Typs `PER_CONTAINER` können nicht repliziert werden.

  – Das MapSet enthält in allen Katalogservicedomänen dieselbe Anzahl von Partitionen. Das MapSet kann dieselbe Anzahl und dieselben Typen von Replikaten haben oder auch nicht.

  – Dieselben Datentypen werden in jeder Katalogservicedomäne für das MapSet repliziert.

  – Das MapSet enthält dieselben Maps und dieselben Schablonen für dynamische Maps in jeder Katalogservicedomäne.

  – Die MapSet verwendet keinen Entitätsmanager. Ein MapSet, das eine Entitätsmap enthält, wird in Katalogservicedomänen nicht repliziert.

  – Das MapSet verwendet keine Write-behind-Caching-Unterstützung. Ein MapSet, das eine Map enthält, die mit Write-behind-Unterstützung konfiguriert ist, wird in Katalogservicedomänen nicht repliziert.

  Alle MapSets mit den vorherigen Merkmalen werden repliziert, nachdem die Katalogservicedomänen in der Topologie gestartet wurden.

- **Klassenlader mit mehreren Katalogservicedomänen**

  Katalogservicedomänen müssen Zugriff auf alle Klassen haben, die als Schlüssel und Werte verwendet werden. Alle Abhängigkeiten müssen sich in allen Klassenpfaden für Grid-Container-JVMs für alle Domänen widerspiegeln. Wenn ein CollisionArbiter-Plug-in den Wert für einen Cacheeintrag abruft, müssen die Klassen für die Werte für die Domäne vorhanden sein, die den Arbiter startet.

Hinweise zu Ladeprogrammen in einer Multimastertopologie

Wenn Sie Ladeprogramme in einer Multimastertopologie verwenden, müssen Sie die möglichen Anforderungen in Bezug auf die Verwaltung von Kollisions- und Revisionsinformationen berücksichtigen. Das Datengrid verwaltet Revisionsinformationen zu den Elementen im Datengrid, sodass Kollisionen erkannt werden können, wenn andere primäre Shards in der Konfiguration Einträge in das Datengrid schreiben. Wenn Einträge von einem Ladeprogramm hinzugefügt werden, werden diese Revisionsinformationen nicht eingeschlossen, und der Eintrag verwendet eine

neue Überarbeitung. Da die Überarbeitung des Eintrags eine neue Einfügung zu sein scheint, könnte eine Fehlkollision auftreten, wenn ein anderes primäres Shard diesen Zustand ebenfalls ändert oder dieselben Daten aus einem Ladeprogramm extrahiert.

Replikationsänderungen rufen die Methode get im Ladeprogramm mit einer Liste der Schlüssel auf, die noch nicht im Datengrid enthalten sind, aber während der Replikationstransaktion geändert werden. Wenn die Replikation stattfindet, sind diese Einträge Kollisionseinträge. Wenn die Kollisionen arbitriert sind, wird beim Anwenden der Überarbeitung eine Stapelaktualisierung im Ladeprogramm aufgerufen, um die Änderungen auf die Datenbank anzuwenden. Alle Maps, die im Revisionsfenster geändert wurden, werden in derselben Transaktion aktualisiert.

Preload-Rätsel

Stellen Sie sich eine Topologie mit zwei Rechenzentren vor: Rechenzentrum A und Rechenzentrum B. Beide Rechenzentren haben unabhängige Datenbanken, aber nur Rechenzentrum A hat ein Datengrid, das aktiv ist. Wenn Sie eine Verbindung zwischen den Rechenzentren für eine Multimasterkonfiguration herstellen, beginnen die Datengrids in Rechenzentrum A, Daten mit Push an die neuen Datengrids im Rechenzentrum B zu übertragen, was bei jedem Eintrag zu einer Kollision führt. Ein weiteres großes Problem tritt bei allen Daten auf, die in der Datenbank in Rechenzentrum B enthalten sind, aber nicht in der Datenbank in Rechenzentrum A. Diese Zeilen werden nicht gefüllt und arbitriert, was zu Inkonsistenzen führt, die nicht aufgelöst werden.

Lösung des Preload-Rätsels

Da die Daten, die nur in der Datenbank enthalten sind, keine Überarbeitungen haben können, müssen Sie das Datengrid immer vollständig aus der lokalen Datenbank laden, bevor Sie die Multimasterverbindung herstellen. Anschließend können beide Datengrids die Daten überarbeiten und arbitrieren und schließlich einen konsistenten Status erreichen.

Teilcache-Rätsel

Mit einem Teilcache versucht die Anwendung zuerst, die Daten im Datengrid zu finden. Wenn die Daten nicht im Datengrid enthalten sind, werden die Daten mithilfe des Loaders in der Datenbank gesucht. Die Einträge im Datengrid werden in regelmäßigen Abständen bereinigt, um die Cachegröße klein zu halten.

Dieser Cachetyp kann in einem Szenario mit einer Multimasterkonfiguration problematisch sein, weil die Einträge im Datengrid Metadaten zur Überarbeitung enthalten, mit deren Hilfe Kollisionen und die Seite, auf der die Änderungen vorgenommen wurden, erkannt werden können. Wenn Verbindungen zwischen den Rechenzentren nicht funktionieren, kann ein Rechenzentrum einen Eintrag aktualisieren und schließlich die Datenbank aktualisieren und den Eintrag im Datengrid ungültig machen. Nach der Wiederherstellung der Verbindung versuchen die Rechenzentren, Überarbeitungen miteinander zu synchronisieren. Da die Datenbank jedoch aktualisiert und der Eintrag im Datengrid ungültig gemacht wurde, geht die Änderung aus der Perspektive des Rechenzentrums, das ausgefallen ist, verloren. Deshalb sind die beiden Seiten des Datengrids nicht mehr synchronisiert und nicht mehr konsistent.

Lösung des Teilcache-Rätsels

Hub- und Peripherietopologie:

Sie können den Loader nur im Hub einer Hub- und Peripherietopologie ausführen, in der die Konsistenz der Daten erhalten bleibt, während das Datengrid horizontal skaliert wird. Wenn Sie diese Implementierung in Erwägung ziehen, müssen Sie jedoch bedenken, dass die Loader ein partielles Laden des Datengrids zulassen können, d. h., dass ein Bereinigungsprogramm (Evictor) konfiguriert wurde. Wenn die Peripherie Ihrer Konfiguration aus Teilcaches besteht, die aber keinen Loader haben, besteht bei Cachefehlern keine Möglichkeit, die Daten aus der Datenbank abzurufen. Wegen dieser Einschränkung müssen Sie eine vollständig gefüllte Cachetopologie mit einer Hub- und Peripheriekonfiguration verwenden.

Invalidierungen und Bereinigung

Bei der Invalidierung entstehen Inkonsistenzen zwischen dem Datengrid und der Datenbank. Daten können über das Programm oder durch Bereinigung aus dem Datengrid entfernt werden. Wenn Sie Ihre Anwendung entwickeln, müssen Sie berücksichtigen, dass bei der Behandlung von Überarbeitungen keine Änderungen repliziert werden, die ungültig gemacht wurden, was zu Inkonsistenzen zwischen primären Shards führt.

Invalidierungsereignisse sind keine Cachestatusänderungen und führen nicht zur Replikation. Alle konfigurierten Bereinigungsprogramme werden unabhängig von anderen Bereinigungsprogrammen in der Konfiguration ausgeführt. Es kann beispielsweise ein Bereinigungsprogramm für einen Speicherschwellenwert in der einen Katalogservicedomäne konfiguriert sein, aber ein anderes, weniger aggressives Bereinigungsprogramm in der anderen verbundenen Katalogservicedomäne. Wenn Datengrideinträge aufgrund der Schwellenwertrichtlinie entfernt werden, sind die Einträge in der anderen Katalogservicedomäne nicht betroffen.

Datenbankaktualisierungen und Datengridinvalidierung

Es treten Probleme auf, wenn Sie in einer Multimasterkonfiguration die Datenbank direkt im Hintergrund aktualisieren, während die Invalidierung im Datengrid für die aktualisierten Einträge aufgerufen wird. Dieses Problem tritt auf, weil das Datengrid die Änderung erst dann auf den anderen primären Shards replizieren kann, wenn der Eintrag durch einen Cache in das Datengrid verschoben wird.

Mehrere Ausgabeprogramme für eine einzige logische Datenbank

Wenn Sie eine einzige Datenbank mit mehreren primären Shards verwenden, die über einen Loader verbunden sind, treten Transaktionskonflikte auf. Ihre Loaderimplementierung muss diese Typen von Szenarien in besonderer Weise behandeln.

Daten durch Multimasterreplikation spiegeln

Sie können unabhängige Datenbanken konfigurieren, die mit unabhängigen Katalogservicedomänen verbunden sind. In dieser Konfiguration kann der Loader Änderungen mit Push aus einem Rechenzentrum in das andere Rechenzentrum übertragen.

Designhinweise für die Multimasterreplikation

Wenn Sie die Multimasterreplikation implementieren, müssen Sie Aspekte wie Arbitrierung, Verbindungen und Leistung beim Design berücksichtigen.

Arbitrierungshinweise für das Topologiedesign

Änderungskollisionen können auftreten, wenn dieselben Datensätze gleichzeitig an zwei Stellen geändert werden können. Konfigurieren Sie alle Katalogservicedomänen mit denselben Werten für Prozessor-, Hauptspeicher und Netzressourcen. Sie können beobachten, dass Katalogservicedomänen, die für die Kollisionsbehandlung (Arbitrierung) zuständig sind, mehr Ressourcen als andere Katalogservicedomänen verbrauchen. Kollisionen werden automatisch erkannt. Sie werden mit einem der folgenden beiden Mechanismen behoben:

- **Standardkollisionsarbiter**: Standardmäßig werden die Änderungen aus der Katalogservicedomäne verwendet, deren Name in der lexikalischer Reihenfolge am niedrigsten steht. Wenn beispielsweise Katalogservicedomäne A und Domäne B einen Konflikt in Bezug auf einen Datensatz verursachen, wird die Änderung aus Katalogservicedomäne B ignoriert. Katalogservicedomäne A behält ihre Version, und der Datensatz in Katalogservicedomäne B wird geändert, sodass er dem Datensatz aus Katalogservicedomäne A entspricht. Dieses Verhalten gilt auch für Anwendungen, in denen Benutzer oder Sitzungen normalerweise gebunden sind oder eine Affinität zu einem der Datengrids haben.
- **Angepasster Kollisionsarbiter**: Anwendungen können einen angepassten Arbiter bereitstellen. Wenn eine Katalogservicedomäne eine Kollision erkennt, ruft sie den Arbiter auf. Informationen zum Entwickeln eines hilfreichen angepassten Arbiters finden Sie unter Angepasste Arbiter für Replikation mehrerer Master entwickeln.

Für Topologien, in denen Kollisionen möglich sind, können Sie eine Hub- und Peripherietopologie oder eine Baumtopologie implementieren. Diese beiden Topologien sind dienlich, um ständige Kollisionen zu vermeiden, die in den folgenden Szenarien auftreten können:

1. In mehreren Katalogservicedomänen tritt eine Kollision auftritt.
2. Jede Katalogservicedomäne behebt die Kollision lokal, was zu Überarbeitungen führt.
3. Die Überarbeitungen kollidieren, was zu Überarbeitungen von Überarbeitungen führt.

Zur Vermeidung von Kollisionen wählen Sie eine bestimmte Katalogservicedomäne, die so genannte *Arbitrierungskatalogservicedomäne*, als Kollisionsarbiter für einen Teil der Katalogservicedomänen aus. In einer Hub- und Peripherietopologie kann der Hub beispielsweise als Kollisionshandler verwendet werden. Der Peripheriekollisionshandler ignoriert alle von den Peripheriekatalogservicedomänen erkannten Kollisionen. Die Hubkatalogservicedomäne erstellt Überarbeitungen, was unerwartete Kollisionsüberarbeitungen verhindert. Die für die Behandlung von Kollisionen zugeordnete Katalogservicedomäne muss eine Verbindung zu allen Domänen haben, für die sie Kollisionen beheben soll. In einer Baumtopologie beheben alle internen übergeordneten Domänen Kollisionen für die ihnen unmittelbar untergeordneten Domänen. Wenn Sie eine Ringtopologie verwendet, ist es nicht möglich, eine einzige Katalogservicedomäne im Ring als Arbiter zu bestimmen.

In der folgenden Tabelle sind die kompatiblen Arbitrierungsansätze für die verschiedenen Topologien zusammengefasst.

Tabelle 7. Arbitrierungsansätze. Der folgenden Tabelle können Sie entnehmen, ob Anwendungsarbitrierung mit den verschiedenen Technologien kompatibel ist.

Topologie	Anwendungs-arbitrierung?	Anmerkungen
Eine Reihe von zwei Katalogservicedomänen	Ja	Wählen Sie eine Katalogservicedomäne als Arbiter aus.
Eine Reihe von drei Katalogservicedomäne	Ja	Die mittlere Katalogservicedomäne muss der Arbiter sein. Stellen Sie sich die mittlere Katalogservicedomäne als Hub in einer einfachen Hub- und Peripherietopologie vor.
Eine Reihe von mehr als drei Katalogservicedomänen	Nein	Anwendungsarbitrierung wird nicht unterstützt.
Ein Hub mit N Peripheriedomänen	Ja	Der Hub mit den Verbindungen zu allen Peripheriedomänen muss die Arbitrierungskatalogservicedomäne sein.
Ein Ring mit N Katalogservicedomänen	Nein	Anwendungsarbitrierung wird nicht unterstützt.
Eine azyklische gerichtete Baumstruktur (k-näre Baumstruktur)	Ja	Alle Stammknoten dürfen nur die ihnen direkt untergeordneten Knoten bewerten.

Verbindungshinweise für das Topologiedesign

Im Idealfall enthält eine Topologie die Mindestanzahl an Verbindungen und optimiert gleichzeitig Kompromisse zwischen Latenzzeit für Änderungen, Fehlertoleranz und Leistungsmerkmale.

- **Latenzzeit bei Änderungen**

  Die Latenzzeit bei Änderungen wird durch die Anzahl zwischengeschalteter Katalogservicedomänen bestimmt, die eine Änderung durchlaufen muss, bevor sie in einer bestimmten Katalogservicedomäne ankommt.

  Eine Topologie weist die beste Latenzzeit bei Änderungen auf, wenn zwischengeschaltete Katalogservicedomänen wegfallen, weil jede Katalogservicedomäne mit jeder anderen Katalogservicedomäne verbunden wird. Eine Katalogservicedomäne muss jedoch proportional zur Anzahl ihrer Verbindungen Replikationsarbeiten ausführen. In großen Topologien kann die reine Anzahl zu definierender Verbindungen einen hohen Verwaltungsaufwand darstellen.

  Die Geschwindigkeit, mit der eine Änderung in andere Katalogservicedomänen kopiert wird, richtet sich nach weiteren Faktoren, wie z. B.:

  – Prozessor- und Netzbandbreite in der Quellenkatalogservicedomäne
  – Anzahl zwischengeschalteter Katalogservicedomänen und Verbindungen zwischen der Quellen- und der Zielkatalogservicedomäne
  – Prozessor- und Netzressourcen, die der Quellenkatalogservicedomäne, der Zielkatalogservicedomäne und den zwischengeschalteten Katalogservicedomänen zur Verfügung stehen

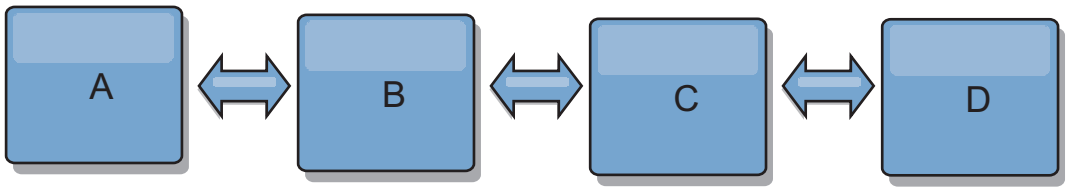
- **Fehlertoleranz**

  Die Fehlertoleranz wird durch die Anzahl der existierenden Pfade zwischen zwei Katalogservicedomänen für die Änderungsreplikation bestimmt.

  Wenn Sie nur eine einzige Verbindung zwischen zwei Katalogservicedomänen haben, wird die Weitergabe von Änderungen aufgrund eines Verbindungsfehlers nicht zugelassen. Änderungen zwischen Katalogservicedomänen werden auch

nicht weitergegeben, wenn bei einer der zwischengeschalteten Domänen ein Verbindungsfehler auftritt. Ihre Topologie kann eine einzige Verbindung von einer Katalogservicedomäne zu einer anderen Katalogservicedomäne enthalten, die über zwischengeschaltete Domänen führt. In diesem Fall werden Änderungen nicht weitergegeben, wenn eine der zwischengeschalteten Katalogservicedomänen ausfällt.

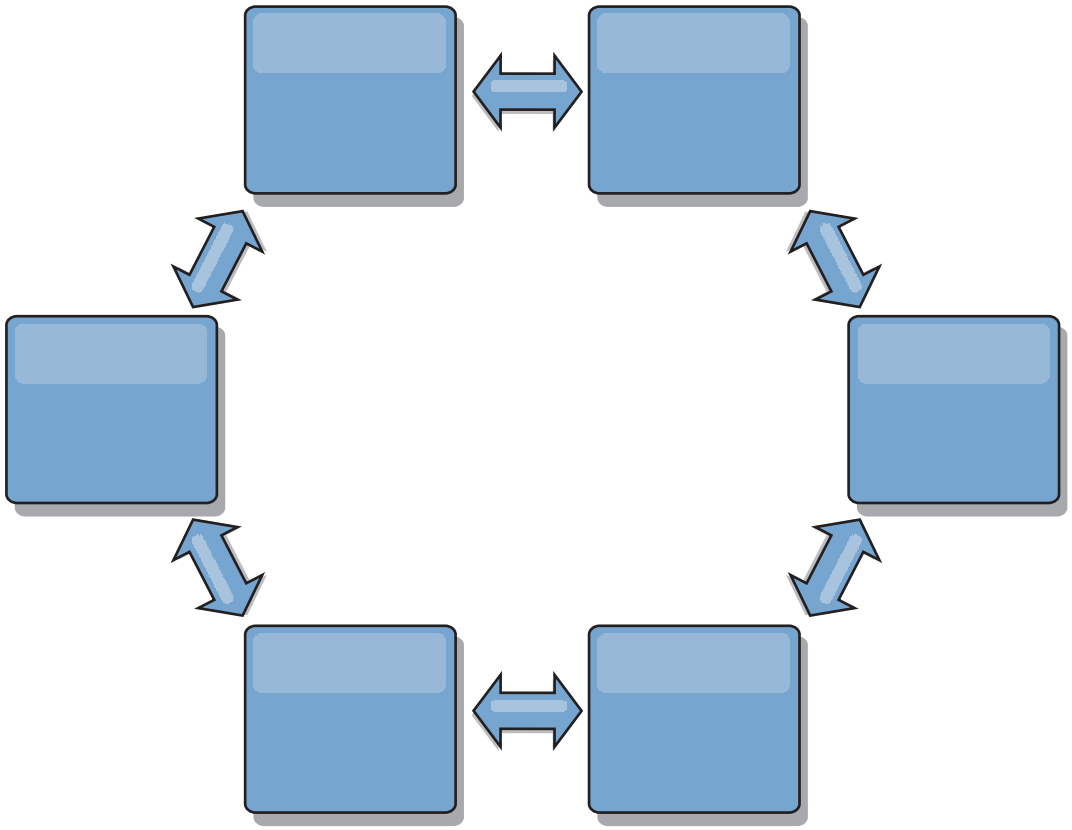
Stellen Sie sich eine Reihentopologie mit vier Katalogservicedomänen, A, B, C und D, vor:



Wenn eine der folgenden Bedingungen zutrifft, sieht die Domäne D Änderungen von Domäne A nicht:

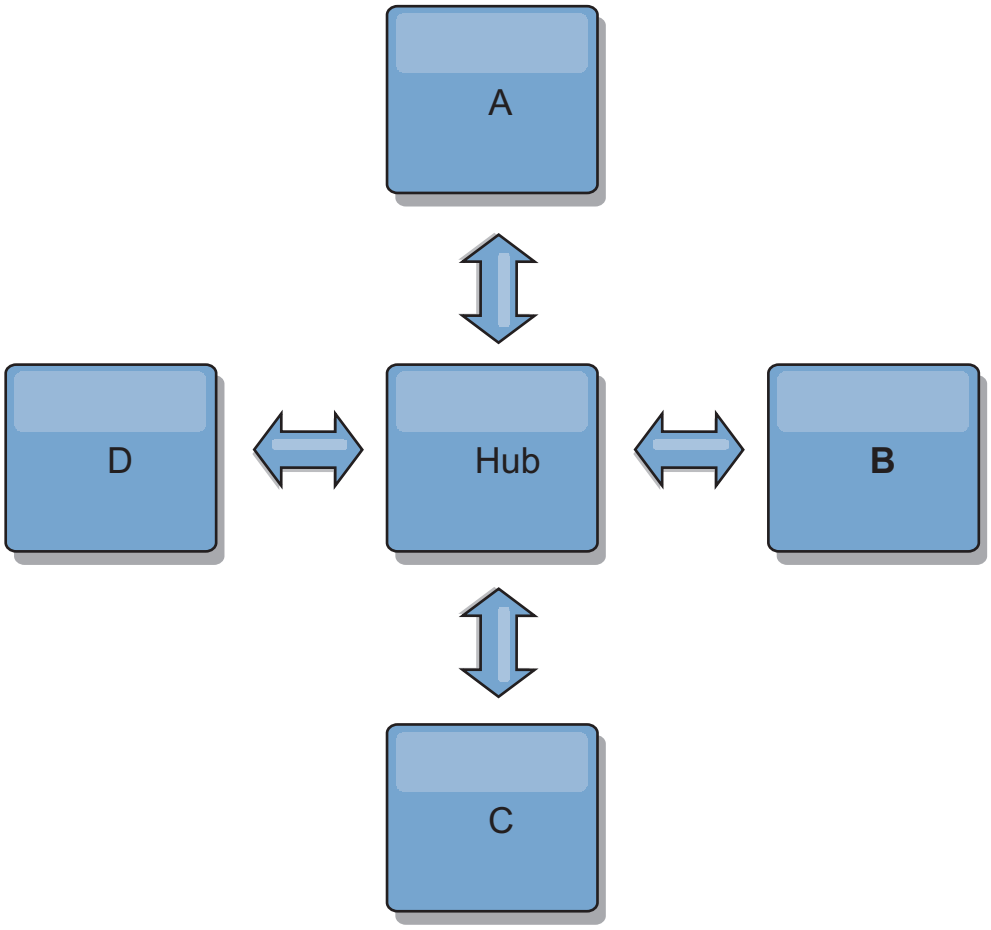
- Die Domäne A ist aktiv, und die Domäne B ist inaktiv.
- Die Domänen A und B sind aktiv, und die Domäne C ist inaktiv.
- Die Verbindung zwischen A und B ist inaktiv.
- Die Verbindung zwischen B und C ist inaktiv.
- Die Verbindung zwischen C und D ist inaktiv.

In einer Ringtopologie hingegen kann jede Katalogservicedomäne Änderungen aus jeder Richtung empfangen.



Wenn beispielsweise ein bestimmter Katalogservice in Ihrer Ringtopologie inaktiv ist, können die beiden benachbarten Domänen weiterhin Änderungen direkt voneinander extrahieren.

Alle Änderungen werden über den Hub weitergegeben. Damit ist das Hub- und Peripheriedesign im Gegensatz zu den Reihen- und Ringtopologien für Störungen anfällig, wenn der Hub ausfällt.



Ein einzige Katalogservicedomäne ist gegen eine bestimmte Anzahl von Serviceausfällen widerstandsfähig. Weiterreichende Ausfälle wie beispielsweise Netzausfälle oder Ausfälle von Verbindungen zwischen physischen Rechenzentren können jedoch den Betrieb Ihrer Katalogservicedomänen unterbrechen.

- **Verbindungen und Leistung**

  Die Anzahl der in einer Katalogservicedomäne definierten Verbindungen wirkt sich auf die Leistung aus. Je mehr Verbindungen definiert werden, desto mehr Ressourcen werden benötigt, und deshalb kann die Replikationsleistung abnehmen. Die Möglichkeit, Änderungen für Domäne A über andere Domänen anzurufen, entlastet die Domäne A effektiv von der Replikation ihrer Transaktionen in allen Domänen. Die Änderungsverteilungslast in einer Domäne wird durch die Anzahl der verwendeten Verbindungen und nicht durch die Anzahl der Domänen in der Topologie beschränkt. Diese Lasteigenschaft unterstützt Skalierbarkeit, damit sich die Domänen in der Topologie die Last der Änderungsverteilung teilen können.

  Eine Katalogservicedomäne kann Änderungen indirekt über andere Katalogservicedomänen abrufen. Stellen Sie sich eine Reihentopologie mit fünf Katalogservicedomänen vor.

  ```
  A <=> B <=> C <=> D <=> E
  ```

  – A bezieht Änderungen von B, C, D und E über B.
  – B bezieht Änderungen von A und C direkt und Änderungen von D und E über C.

– C bezieht Änderungen von B und D direkt und Änderungen von A über B und Änderungen von E über D.
– D bezieht Änderungen von C und E direkt und Änderungen von A und B über C.
– E bezieht Änderungen von D direkt und Änderungen von A, B und C über D.

Die Verteilungslast ist in den Katalogservicedomänen A und E am geringsten, weil sie jeweils nur eine Verbindung zu einer einzigen Katalogservicedomäne haben. Die Domänen B, C und D haben jeweils eine Verbindung zu zwei Domänen. Somit ist die Verteilungslast in den Domänen B, C und D doppelt so hoch wie die Last in den Domänen A und E. Die Arbeitslast richtet sich nach der Anzahl der Verbindungen in jeder Domäne und nicht nach der Gesamtanzahl der Domänen in der Topologie. Die beschriebene Verteilung der Last bliebe damit auch bei 1000 Domänen in der Reihe konstant.

Leistungshinweise für die Multimasterreplikation

Berücksichtigen Sie bei der Verwendung von Multimaster-Replikationstopologien die folgenden Einschränkungen:

- **Optimierung der Änderungsverteilung**, die im vorherigen Abschnitt beschrieben wurde
- **Leistung der Replikationsverbindungen**: WebSphere eXtreme Scale erstellt einen einzigen TCP/IP-Socket zwischen jedem JVM-Paar. Der gesamte Datenverkehr zwischen den JVMs findet über diesen Socket statt. Dies gilt auch für den Datenverkehr aus der Multimasterreplikation. Die Katalogservicedomänen werden in mindestens *n* Container-JVMs gehostet, wodurch mindestens *n* TCP-Verbindungen zu Peerkatalogservicedomänen bereitgestellt werden. Deshalb haben die Katalogservicedomänen mit einer höheren Anzahl an Containern höhere Replikationsleistungsstufen. Je mehr Container vorhanden sind, desto mehr Prozessor- und Netzressourcen sind erforderlich.
- **Optimierung des TCP-Sliding-Window und RFC 1323**: Die Unterstützung von RFC 1323 auf beiden Seiten einer Verbindung führt zu mehr Daten bei einem Umlauf. Diese Unterstützung führt zu einem höheren Durchsatz, was die Größe des Sliding-Windows um den Faktor 16 erhöht.

  TCP-Sockets verwenden, wie bereits erwähnt, einen Sliding-Window-Mechanismus, um den Fluss von Massendaten zu steuern. Dieser Mechanismus beschränkt den Socket gewöhnlich auf 64 KB in einem Umlaufintervall. Wenn das Umlaufintervall 100 ms lang ist, ist die Bandbreite ohne Optimierung auf 640 KB/Sekunde beschränkt. Um die verfügbare Bandbreite einer Verbindung vollständig nutzen zu können, müssen unter Umständen spezielle Optimierungstasks für ein Betriebssystem ausgeführt werden. Die meisten Betriebssysteme haben Optimierungsparameter, einschließlich Optionen für RFC 1323, um den Durchsatz über Verbindungen mit hoher Latenzzeit zu verbessern.

  Es gibt mehrere Faktoren, die sich auf die Replikationsleistung auswirken können:

  – Geschwindigkeit, mit der eXtreme Scale Änderungen abruft
  – Geschwindigkeit, mit der eXtreme Scale Replikationsanforderungen bei Abruf verarbeiten kann
  – Kapazität des Sliding-Windows
  – Optimierung der Netzpuffer auf beiden Seiten einer Verbindung, was eXtreme Scale ermöglicht, Änderungen effizient über den Socket abzurufen

- **Objektserialisierung**: Alle Daten müssen serialisierbar sein. Wenn eine Domäne `COPY_TO_BYTES` nicht verwendet, muss die Katalogservicedomäne Java-Serialisierung oder ObjectTransformer verwenden, um die Serialisierungsleistung zu optimieren.
- **Komprimierung**: WebSphere eXtreme Scale komprimiert standardmäßig alle Daten, die zwischen Katalogservicedomänen gesendet werden. Die Komprimierung kann momentan nicht inaktiviert werden.
- **Hauptspeicheroptimierung**: Die Speicherbelegung für eine Multimasterreplikationstopologie ist weitgehend unabhängig von der Anzahl der Katalogservicedomänen in der Topologie.

  Bei der Multimasterreplikation entsteht ein fester Verarbeitungsaufwand pro Mapeintrag für die Versionssteuerung. Außerdem überwacht jeder Container ein festes Datenvolumen für jede Katalogservicedomäne in der Topologie. Eine Topologie mit zwei Katalogservicedomänen belegt ungefähr denselben Speicher wie eine Topologie mit 50 Katalogservicedomänen. WebSphere eXtreme Scale verwendet keine Wiedergabeprotokolle oder ähnlichen Warteschlangen in der Implementierung. Deshalb ist keine Wiederherstellungsstruktur verfügbar, wenn eine Replikationsverbindung über einen längeren Zeitraum hinweg und nach späteren Neustarts nicht verfügbar ist.

Interoperabilität mit anderen Produkten

Sie können WebSphere eXtreme Scale mit anderen Produkten wie WebSphere Application Server und WebSphere Application Server Community Edition integrieren.

WebSphere Application Server

Sie können WebSphere Application Server in verschiedene Aspekte Ihrer Konfiguration von WebSphere eXtreme Scale integrieren. Sie können Datengridanwendungen implementieren und WebSphere Application Server als Host für Container- und Katalogserver verwenden. Alternativ können Sie eine heterogene Umgebung verwenden, in der WebSphere eXtreme Scale Client in der Umgebung von WebSphere Application Server zusammen mit eigenständigen Katalog- und Container-Servern ausgeführt wird. Außerdem können Sie die Sicherheit von WebSphere Application Server in Ihrer Umgebung von WebSphere eXtreme Scale verwenden.

WebSphere-Produkte für Geschäftsprozessmanagement und Konnektivität

WebSphere-Produkte für Geschäftsprozessmanagement und Konnektivität, einschließlich WebSphere Integration Developer, WebSphere Enterprise Service Bus und WebSphere Process Server, lassen sich mit Back-End-Systemen wie CICS, Web-Services, Datenbanken oder JMS-Topics und -Warteschlangen integrieren. Sie können WebSphere eXtreme Scale der Konfiguration hinzufügen, um die Ausgabe dieser Back-End-Systeme zwischenzuspeichern und damit die Gesamtleistung Ihrer Konfiguration zu erhöhen.

WebSphere Commerce

WebSphere Commerce kann das Caching von WebSphere eXtreme Scale als Ersatz für den dynamischen Cache nutzen. Da bei Verwendung dieses Cachings doppelte Einträge im dynamische Cache und häufige Invalidierungsverarbeitungen entfallen, um den Cache in Zeiten mit Spitzenlast synchronisiert zu halten, können Sie Leistung, Skalierbarkeit und hohe Verfügbarkeit verbessern.

WebSphere Portal

Sie können HTTP-Sitzungen über WebSphere Portal persistent in einem Datengrid in WebSphere eXtreme Scale speichern. Außerdem kann IBM Web Content Manager in IBM WebSphere Portal dynamische Cacheinstanzen verwenden, um wiedergegebenen Inhalt zu speichern, der von Web Content Manager abgerufen wird, wenn erweitertes Caching aktiviert ist. WebSphere eXtreme Scale bietet als Alternative zur Standardimplementierung des dynamischen Caches eine Implementierung des dynamischen Caches, in der zwischengespeicherter Inhalt in einem elastischen Datengrid gespeichert wird.

WebSphere Application Server Community Edition

WebSphere Application Server Community Edition kann den Sitzungsstatus zwar zur gemeinsamen Nutzung bereitstellen, aber nicht auf effiziente und skalierbare Weise. WebSphere eXtreme Scale stellt eine verteilte Persistenzschicht mit hoher Leistung bereit, die zum Replizieren des Status verwendet werden kann, sich aber nicht problemlos mit Anwendungsservern außerhalb von WebSphere Application Server integrieren lässt. Sie können diese beiden Produkte integrieren, um eine skalierbare Lösung für die Sitzungsverwaltung zu erhalten.

WebSphere Real Time

Mit der Unterstützung für WebSphere Real Time, dem branchenführenden Java-Angebot für die Echtzeitverarbeitung, ermöglicht WebSphere eXtreme Scale XTP-Anwendungen (Extreme Transaction Processing) konsistentere und voraussagbare Antwortzeiten.

Überwachung

WebSphere eXtreme Scale kann mit verschiedenen gängigen Lösungen für die Unternehmensüberwachung überwacht werden. Es werden Plug-in-Agenten für IBM Tivoli Monitoring and Hyperic HQ bereitgestellt, die WebSphere eXtreme Scale mit Hilfe öffentlich zugänglicher Management-Beans überwachen. CA Wily Introscope verwendet die Java-Methodeninstrumentierung, um Statistiken zu erfassen.

.NET 8.6+

Umgebungen mit Microsoft Visual Studio, IIS und .NET

Weitere Informationen zu unterstützten Umgebungen mit Microsoft Visual Studio, IIS und .NET finden Sie unter Hinweise zu Microsoft .NET.

Kapitel 3. Szenarien

Die Szenarien enthalten Informationen aus der realen Welt, um ein vollständiges Bild zu erstellen. Führen Sie ein Szenario aus, um sich mit den neuen Konzepten vertraut zu machen oder gängige Tasks in WebSphere eXtreme Scale auszuführen.

Szenario: Unternehmensdatengrid konfigurieren

Konfigurieren Sie ein Unternehmensdatengrid, wenn Sie möchten, dass Java- und .NET-Anwendungen Verbindungen zu demselben Datengrid herstellen.

Vorbereitende Schritte

- Installieren Sie das Produkt. Sie müssen die Serverlaufzeitumgebung und die Clients installieren. Für Clients können Sie Java- und .NET-Clients verwenden. Weitere Informationen finden Sie unter Installation.
- Wenn Sie in Upgrade von einem früheren Release durchführen, müssen alle Container- und Katalogserver dasselbe Release-Level haben. Weitere Informationen finden Sie unter Upgrade und Migration von WebSphere eXtreme Scale durchführen.

Informationen zu diesem Vorgang

Übersicht über Unternehmensdatengrids

Unternehmensdatengrids verwenden den Transportmechanismus "eXtremeIO" und ein neues Serialisierungsformat. Mit dem neuen Transport und Serialisierungsformat können Sie Java- und .NET-Clients mit demselben Datengrid verbinden.

Mit dem Unternehmensdatengrid können Sie mehrere Typen von Anwendungen erstellen, die in verschiedenen Programmiersprachen geschrieben sind, um auf dieselben Objekte im Datengrid zuzugreifen. In früheren Releases durften Datengridanwendungen nur in der Programmiersprache Java geschrieben werden. Mit der Unternehmensdatengridfunktion können Sie .NET-Anwendungen schreiben, die Objekte in demselben Datengrid erstellen, abrufen, aktualisieren und löschen, wie die Java-Anwendung.

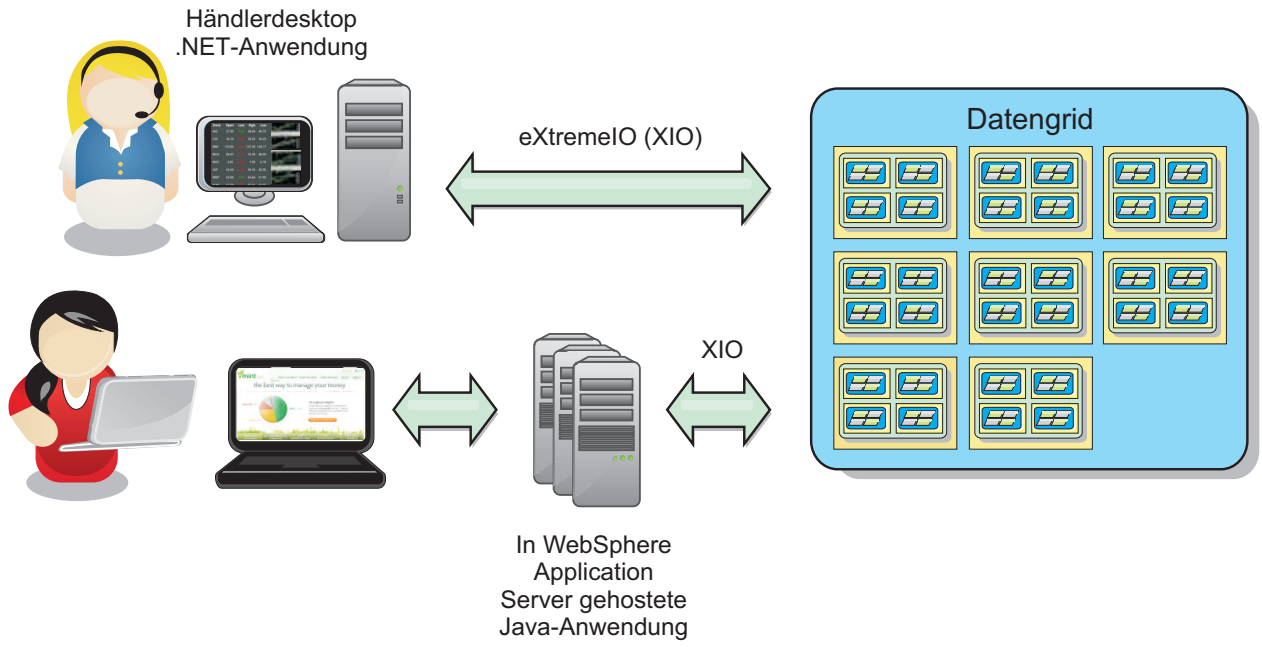


Abbildung 59. Allgemeine Übersicht über Unternehmensdatengrids

Objektaktualisierungen über verschiedene Anwendungen

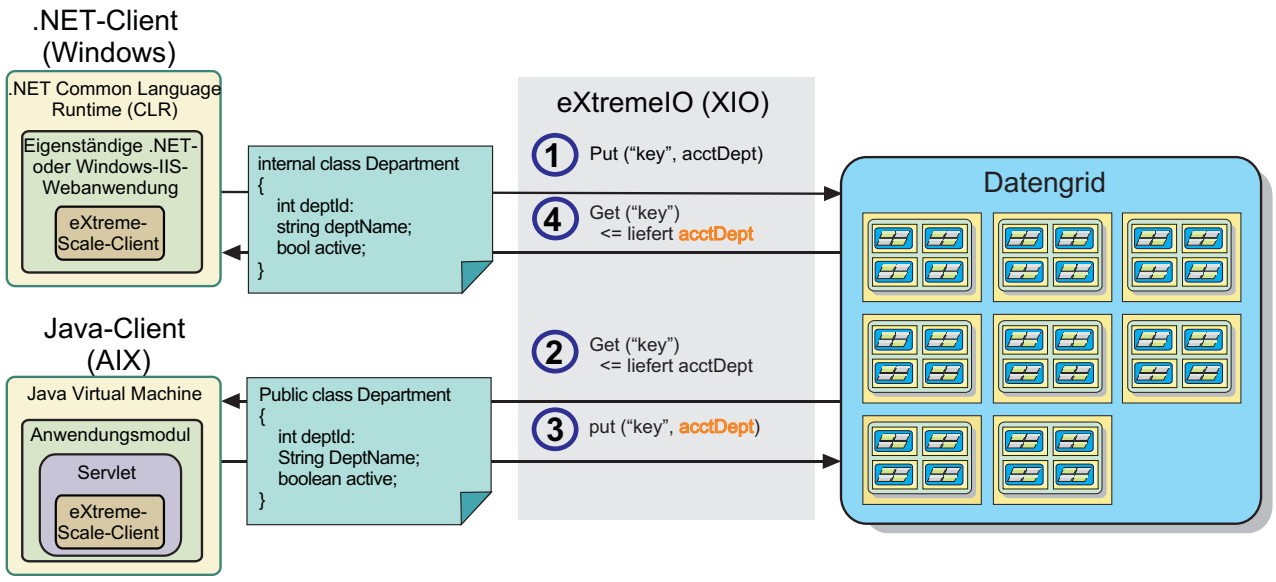


Abbildung 60. Ablauf der Objektaktualisierung in einem Unternehmensdatengrid

1. Der .NET-Client speichert Daten in seinem eigenen Format im Datengrid.
2. Die Daten werden in einem universellen Format gespeichert, sodass die Daten bei Anforderung durch den Java-Client in das Java-Format konvertiert werden können.
3. Der Java-Client aktualisiert die Daten und speichert sie dann erneut.
4. Beim Zugriff auf die aktualisierten Daten durch den .NET-Client werden die Daten in das .NET-Format konvertiert.

Transportmechanismus

eXtremeIO (XIO) ist ein plattformübergreifendes Transportprotokoll XIO ersetzt den Java-gebundenen Object Request Broker (ORB). Mit dem ORB wird WebSphere eXtreme Scale an native Java-Clientanwendungen gebunden. XIO ist ein angepasster Transportmechanismus, der speziell für das Datencaching bestimmt ist und Clientanwendungen, die in anderen Programmiersprachen geschrieben sind, ermöglicht, die Verbindung zum Datengrid herzustellen.

Serialisierungsformat

XDF (eXtreme Data Format) ist ein plattformübergreifendes Serialisierungsformat. XDF ersetzt die Java-Serialisierung in Maps, in denen das copyMode-Attribut in der ObjectGrid-XML-Deskriptordatei auf den Wert `COPY_TO_BYTES` gesetzt ist. Mit XDF ist die Leistung höher, und die Daten sind kompakter. Außerdem ermöglicht die Einführung von XDF Clientanwendungen, die in verschiedenen Programmiersprachen geschrieben sind, eine Verbindung zu demselben Datengrid herzustellen.

IBM eXtremeIO (XIO) konfigurieren

IBM eXtremeIO (XIO) ist ein Transportmechanismus, der den Object Request Broker (ORB) ersetzt.

Vorbereitende Schritte

- **8.6** Zum Konfigurieren von XIO müssen alle Katalog- und Container-Server das Release-Level Version 8.6 haben. Weitere Informationen finden Sie unter eXtreme-Scale-Server aktualisieren.

8.6+ Sie können XIO für alle Container-Server in Ihrer Katalogservicedomäne konfigurieren, indem Sie XIO in den Katalogservern aktivieren. Die Container-Server erkennen den Transporttyp des Katalogservers und verwenden diesen Transporttyp.

Vorgehensweise

8.6+ Wie Sie XIO aktivieren, richtet sich nach dem Typ der verwendeten Server:

- Aktivieren Sie XIO in Ihren eigenständigen Katalogservern.

  XIO ist standardmäßig aktiviert, wenn Sie Ihren Katalogserver mit dem Befehl **`startXsServer`** starten. Weitere Informationen finden Sie unter Container-Server starten, die den Transport IBM eXtremeIO (XIO) verwenden.
- Aktivieren Sie XIO in den Servern, die in WebSphere Application Server ausgeführt werden.

  Sie können XIO in Ihrer Katalogservicedomäne über die Administrationskonsole von WebSphere Application Server aktivieren. Klicken Sie auf **Systemverwaltung** > **WebSphere eXtreme Scale** > **Katalogservicedomänen** > ***Katalogservicedomäne***. Wählen Sie **Kommunikation über IBM eXtremeIO (XIO) aktivieren**. Wenden Sie Ihre Änderungen an. Weitere Informationen finden Sie unter Katalogservice in WebSphere Application Server konfigurieren.
- Aktivieren Sie XIO in den Servern, die im Liberty-Profil ausgeführt werden.

  Zum Aktivieren von XIO in einem Liberty-Profilserver setzen Sie das Attribut `transport` in der Datei `server.xml` auf `XIO`. Sehen Sie sich beispielsweise die hervorgehobene Eigenschaft im folgenden Codebeispiel an:

```
<featureManager>
  ...
  <feature>eXtremeScale.server-1.1</feature>
```

```
</featureManager>

<xsServer isCatalog="true" transport="XIO" listenerPort="2809" ... />
```

Achtung: Der Server muss ein Katalogserver sein, und deshalb muss `isCatalog` auf "true" gesetzt werden, wenn Sie XIO konfigurieren. Die Einstellung `listenerPort` ist nicht erforderlich. XIO kann diesen Port jedoch erkennen, wenn Sie ihn aktivieren. Wenn Sie XIO nicht aktivieren, wird stattdessen der ORB an diesem Port verwendet.

Führen Sie als Nächstes den Befehl **start** aus, um Ihre Liberty-Profilserver zu starten. Weitere Informationen finden Sie unter Server im Liberty-Profil starten und stoppen.

8.6+ Sie können Befehlszeilenargumente und Servereigenschaften verwenden, um das XIO-Verhalten zu konfigurieren:

- Optional: Aktualisieren Sie die Servereigenschaftendatei für jeden Container-Server in der Konfiguration, um XIO-Eigenschaften zu aktivieren. Nachdem Sie Ihre Entscheidung bezüglich der zu definierenden Eigenschaften getroffen haben, können Sie die Werte in der Servereigenschaftendatei oder über das Programm mit der Schnittstelle ServerProperties setzen. Weitere Informationen zu den Eigenschaften, die Sie definieren können, finden Sie unter „IBM eXtremeIO (XIO) optimieren” auf Seite 203.

8.6+ Ergebnisse

Die von Ihnen konfigurierten Server verwenden den XIO-Transport. Verwenden Sie die Informationen im Abschnitt Transporttyp Ihrer Katalogservicedomäne anzeigen, um sicherzustellen, dass die Konfiguration korrekt ist.

Nächste Schritte

Sie können IBM eXtremeMemory verwenden, um Garbage-Collection-Pausen zu verhindern, was zu einer konstanteren Leistung und vorhersagbaren Antwortzeiten führt. Weitere Informationen finden Sie unter IBM eXtremeMemory konfigurieren.

Datengrids für die Verwendung von eXtreme Data Format (XDF) konfigurieren

Wenn Sie ein Unternehmensdatengrid verwenden, müssen Sie XDF aktivieren, sodass Java und .NET auf dieselben Datengridobjekte zugreifen können. Verwenden Sie XDF, um Schlüssel und Werte im Datengrid in einem sprachenunabhängigen Format zu serialisieren und zu speichern.

Vorbereitende Schritte

Aktivieren Sie IBM eXtremeIO (XIO) in der Umgebung. Weitere Informationen finden Sie im Abschnitt „IBM eXtremeIO (XIO) konfigurieren” auf Seite 195.

Informationen zu diesem Vorgang

Aktivieren Sie eXtreme Data Format (XDF), um serialisierte Objekte in einem sprachenunabhängigen Format zu speichern. XDF ist jetzt die Standardserialisierungstechnologie, die verwendet wird, wenn Sie XIO ausführen und der Kopiermodus für Maps auf `COPY_TO_BYTES` gesetzt ist. Wenn Sie dieses Feature aktivieren, können Java- und C#-Objekte Daten in demselben Datengrid gemeinsam nutzen. Sie können den XDF-Modus für Installationen von WebSphere eXtreme Scale in einer ei-

genständigen Umgebung und für Installationen von WebSphere eXtreme Scale in einer Umgebung von WebSphere Application Server festlegen.

Wenn Sie XDF verwenden, können Sie die folgenden Vorteile nutzen:

- Serialisierung der Daten für die gemeinsame Nutzung durch Java- und C#/ .NET-Anwendungen
- Indexierung von Daten auf dem Server, ohne dass die Benutzerklassen vorhanden sein müssen, wenn der Feldzugriff verwendet wird
- Automatische Versionssteuerung Ihrer Klassen, sodass Sie die Klassendefinitionen erweitern können, wenn Sie Anwendungen hinzufügen, die neuere Versionen der Dateien erfordern. Ältere Versionen der Daten können verwendet werden, indem die Schnittstelle "Mergable" genutzt wird.
- Partitionierung der Daten mit Annotationen in Java und C# für eine konsistente Partitionierung über die Anwendung

Vorgehensweise

Setzen Sie in der ObjectGrid-XML-Deskriptordatei das Attribut **CopyMode** im Element "backingMap" auf XDF.

```
<backingMap name="Employee" lockStrategy="PESSIMISTIC" copyMode="COPY_TO_BYTES">
```

Nächste Schritte

Entwickeln Sie Anwendungen, die Daten gemeinsam nutzen können. Weitere Informationen finden Sie unter „Unternehmensdatengridanwendungen entwickeln”.

Unternehmensdatengridanwendungen entwickeln

Nach der Konfiguration von IBM eXtremeIO können Sie Anwendungen schreiben, die auf das Unternehmensdatengrid zugreifen.

Vorbereitende Schritte

- Richten Sie Ihre Entwicklungsumgebung ein, und sehen Sie sich die API-Dokumentation an. Weitere Informationen finden Sie unter Einführung in die Entwicklung von Anwendungen.
- Sie müssen vorhandene Java- oder .NET-Anwendungen haben, die auf das Datengrid zugreifen. Weitere einführende Informationen zum Schreiben von Anwendungen finden Sie unter Modul 2 des Lernprogramms "Einführung": Clientanwendung erstellen.

Weiterentwicklung von Klassen

eXtreme Data Format (XDF) lässt die Weiterentwicklung von Klassen zu. Bei der Weiterentwicklung von Klassen können Sie die Klassendefinitionen, die im Datengrid verwendet werden, ohne Beeinflussung älterer Anwendungen, die frühere Versionen der Klassen verwenden, weiterentwickeln. Diese älteren Klassen greifen auf Daten in derselben Map wie die neuen Anwendungen zu.

Übersicht

Die Weiterentwicklung von Klassen ist eine weitere Erweiterung der Identifikation von Klassen und Feldern, die bestimmen, ob zwei Typen ausreichend kompatibel sind, um miteinander zu funktionieren. Klassen können miteinander funktionieren, wenn eine der Klassen weniger Felder als die andere Klasse hat. Die folgenden Benutzerszenarien sind in der XDF-Implementierung abgebildet:

Mehrere Versionen derselben Objektklasse

In diesem Szenario gibt es eine Map in einer Einzelhandelsanwendung, die zur Überwachung von Kunden verwendet wird. Diese Map hat zwei verschiedene Schnittstellen. Eine Schnittstelle ist für die Webeinkäufe bestimmt. Die zweite Schnittstelle ist für Telefoneinkäufe bestimmt. In Version 2 dieser Einzelhandelsanwendung entscheiden Sie, Webeinkäufern basierend auf deren Kaufgewohnheiten Rabatte einzuräumen. Dieser Rabatt wird zusammen mit dem Kundenobjekt (Customer) gespeichert. Die Mitarbeiter für Telefoneinkäufe verwenden weiterhin Version 1 der Anwendung, die das neue Rabattfeld in der Webversion nicht kennt. Sie möchten, dass Kundenobjekte aus Version 2 der Anwendung mit Kundenobjekten funktionieren, die mit Version 1 der Anwendung erstellt wurden, und umgekehrt.

Mehrere Versionen einer anderen Objektklasse

In diesem Szenario gibt es eine Einzelhandelsanwendung, die in Java geschrieben ist und eine Map von Kundenobjekten (Customer) verwaltet. Außerdem gibt es eine andere Anwendung, die in C# geschrieben ist und dazu verwendet wird, den Bestand im Lager zu verwalten und Waren an Kunden auszuliefern. Diese Klassen sind momentan basierend auf den Namen der Klassen, Felder und Typen kompatibel. In der Java-Einzelhandelsanwendung möchten Sie nun dem Kundendatensatz eine Option hinzufügen, um den Verkäufer einem Kundenkonto zuzuordnen. Sie möchten die Lageranwendung jedoch nicht mit diesem Feld aktualisieren, weil das Feld im Lager nicht erforderlich ist.

Mehrere inkompatible Versionen derselben Klasse

In diesem Szenario enthalten die Einzelhandelsanwendung und die Anwendung zur Bestandsführung jeweils ein Kundenobjekt (Customer). Die Anwendung zur Bestandsführung verwendet ein ID-Feld mit dem Datentyp "String", und die Einzelhandelsanwendung verwendet ein ID-Feld mit dem Datentyp "Integer". Diese Datentypen sind nicht kompatibel. Deshalb werden die Objekte wahrscheinlich nicht in derselben Map gespeichert. Diese Objekte müssen von der XDF-Serialisierung verarbeitet und als zwei unterschiedliche Datentypen behandelt werden. Obwohl dieses Szenario eigentlich keine Weiterentwicklung von Klassen ist, muss es während des Gesamtanwendungsentwurfs berücksichtigt werden.

Entschluss zur Weiterentwicklung

XDF versucht, eine Klasse weiterzuentwickeln, wenn die Klassennamen übereinstimmen und die Feldnamen keine widersprüchlichen Typen haben. Die Verwendung der Annotationen "ClassAlias" und "FieldAlias" sind hilfreich, wenn Sie versuchen, Klassen von C#- und Java-Anwendungen abzugleichen, in denen die Namen der Klassen oder Felder geringfügig verschieden sind. Sie können diese Annotationen in die Java- und/oder C#-Anwendung einfügen. Die Suche der Klasse in der Java-Anwendung kann jedoch weniger effizient sein als die Definition der Annotation "ClassAlias" in der C#-Anwendung. Weitere Informationen zu den Annotationen "ClassAlias" und "FieldAlias" finden Sie unter „ClassAlias- und FieldAlias-Annotationen” auf Seite 200.

Auswirkung fehlender Felder in serialisierten Daten

Der Konstruktor der Klasse wird während der Deserialisierung nicht aufgerufen. Deshalb haben alle fehlenden Felder einen Standardwert, der den Feldern basierend auf der Sprache zugeordnet wird. Die Anwendung, die neue Felder hinzufügt, muss in der Lage sein, die fehlenden Felder zu erkennen und zu reagieren,

wenn eine ältere Version der Klasse abgerufen wird.

Aktualisierung der Daten ist die einzige Möglichkeit für ältere Anwendungen, die neueren Felder zu verwalten

Eine Anwendung könnte eine Abrufoperation ausführen und die Map mit einer älteren Version der Klasse aktualisieren, in der einige Felder im serialisierten Wert des Clients fehlen. Der Server führt dann die Werte auf dem Server zusammen und bestimmt, ob Felder aus der ursprünglichen Version in den neuen Datensatz eingefügt werden müssen. Wenn eine Anwendung eine Abrufoperation ausführt und dann einen Eintrag entfernt und einfügt, gehen die Felder aus dem ursprünglichen Wert verloren.

Zusammenführungsfunktionen

Objekte in einem Array oder in einer Sammlung werden von XDF nicht zusammengeführt. Es ist nicht immer klar, ob eine Aktualisierung eines Arrays oder einer Sammlung dazu bestimmt ist, die Elemente dieses Arrays oder den Typ zu ändern. Wenn beim Verschieben eines Eintrags im Array eine Zusammenführung basierend auf der Positionierung stattfindet, kann XDF Felder zusammenführen, die nicht für eine Zuordnung bestimmt sind. Deshalb versucht XDF nicht, den Inhalt von Arrays oder Sammlungen zusammenzuführen. Wenn Sie jedoch ein Array in eine neuere Version einer Klassendefinition hinzufügen, wird das Array wieder mit der früheren Version der Klasse zusammengeführt.

ClassAlias- und FieldAlias-Annotationen für die Korrelation von Java- und .NET-Klassen definieren

Verwenden Sie ClassAlias- und FieldAlias-Annotationen, um die gemeinsame Nutzung von Datengriddaten durch Ihre Java- und .NET-Klassen zu aktivieren.

Vorbereitende Schritte

- IBM eXtremeIO muss konfiguriert sein. Weitere Informationen finden Sie im Abschnitt „IBM eXtremeIO (XIO) konfigurieren” auf Seite 195.
- Das Attribut "copyMode" in der ObjectGrid-XML-Deskriptordatei muss auf "COPY_TO_BYTES" gesetzt werden. Weitere Informationen finden Sie unter „Datengrids für die Verwendung von eXtreme Data Format (XDF) konfigurieren” auf Seite 196.

Informationen zu diesem Vorgang

Sie können ClassAlias- und FieldAlias-Annotationen verwenden, wenn Sie eine vorhandene Java-Klasse haben und eine entsprechende C#-Klasse erstellen möchten. In diesem Szenario können Sie der C#-Klasse die Annotationen hinzufügen, die den Java-Klassennamen enthalten. Weitere Informationen zu den ClassAlias- und FieldAlias-Annotationen finden Sie unter „ClassAlias- und FieldAlias-Annotationen” auf Seite 200.

Vorgehensweise

Verwenden Sie ClassAlias- und FieldAlias-Annotationen, um Objekte aus einer Java-Klasse und einer C#-Klasse zu korrelieren. Java

Java

```
@ClassAlias("Employee")
class com.company.department.Employee {

     @FieldAlias("id")
     int myId;

      String name;
}
```

Abbildung 61. Java-Beispiel mit ClassAlias- und FieldAlias-Annotationen

.NET

.NET

```
[ ClassAlias( "Employee") ]
class Com.MyCompany.Employee {

     [ FieldAlias("id") ]
      int identifier;

      string name;
}
```

Abbildung 62. .NET-Beispiel mit ClassAlias- und FieldAlias-Attributen

ClassAlias- und FieldAlias-Annotationen:

Verwenden Sie ClassAlias- und FieldAlias-Annotationen, um mehreren Klassen zu ermöglichen, Datengriddaten gemeinsam zu nutzen. Daten können von zwei Java-Klassen oder von einer Java- und einer .NET-Klasse gemeinsam genutzt werden.

Wenn Sie zwei Klassen mit demselben Namen und denselben Feldern definieren, werden die Datengriddaten automatisch von den Klassen gemeinsam genutzt. Angenommen, Sie haben eine Klasse "Customer1" in Ihrer Java-Anwendung und eine Klasse "Customer1" in Ihrer .NET-Anwendung, die dieselben Felder haben. In diesem Fall werden die Daten von den Klassen gemeinsam genutzt. Hierbei wird angenommen, dass der Klassenname auch das Klassenqualifikationsmerkmal enthält, das auch der Paketname in Java und der Namespace in C# ist. Der Paketname und der Namespace werden automatisch gemeinsam verwendet, weil die Namen von Namespace und Paket übereinstimmen. Sehen sich das folgende Beispiel an, in dem bei beiden Namen nicht zwischen Groß- und Kleinschreibung unterschieden wird:

```
Java:
package com.mycompany.app
public class SampleClass {
int field1;
String field2;
}

C#
namespace Com.MyCompany.App
public class SampleClass {
int field1;
string field2;
}
```

Sie können jedoch auch Daten zwischen Klassen korrelieren, die unterschiedliche Namen haben. Wenn Sie Daten, die im Datengrid gespeichert werden sollen, zwi-

schen Klassen korrelieren möchten, die unterschiedliche Namen haben, verwenden Sie ClassAlias- oder FieldAlias-Annotationen.

Zwischen zwei Java-Anwendungen: Sie können zwei verschiedene Klassen mit unterschiedlichen Namen in separaten Java-Anwendungsumgebungen definieren. Durch Markierung der Klassen mit derselben ClassAlias-Annotation werden alle Felder Feldtypen dieser beiden Klassen abgeglichen. Die Klassen werden mit derselben Klassentyp-ID korreliert, obwohl sie unterschiedliche Klassennamen haben. Die Klassentyp-ID und die Metadaten können anschließend von den Klassen in den verschiedenen Java-Anwendungslaufzeitumgebungen wiederverwendet werden.

Zwischen einer Java-Anwendung und einer .NET-Anwendung: Sie können ähnliche Annotationen in Ihrer C#-Anwendung verwenden, um die C#-Klasse mit einer Java-Klasse zu korrelieren. Die ClassAlias-Attribute, die für die Klasse in C# definiert sind, und die Felder werden einer Java-Klasse mit derselben ClassAlias-Annotation zugeordnet.

Schlüssel mit PartitionKey-Annotationen Partitionen zuordnen

Ein PartitionKey-Alias wird verwendet, um die Felder oder Attribute anzugeben, für die eine Hash-Code-Berechnung ausgeführt werden soll, um die Partition zu bestimmen, in der Daten gespeichert werden. Die Annotation PartitionKey ist nur für Schlüsselattribute gültig.

Vorbereitende Schritte

Sie müssen eXtreme Data Format (XDF) verwenden. Weitere Informationen finden Sie unter „Datengrids für die Verwendung von eXtreme Data Format (XDF) konfigurieren” auf Seite 196.

Informationen zu diesem Vorgang

Sie definieren einen PartitionKey-Alias, um sicherzustellen, dass mehrere Klassen Daten in derselben Partition speichern. Wenn Sie den PartitionKey-Wert beispielsweise auf den Schlüssel "departmentID" setzen, werden Mitarbeiterdatensätze durch Kollokation in derselben Partition zusammengefasst.

Die Schnittstelle "PartionableKey" ist die vorhandene Java-Schnittstelle und hat Vorrang vor der Annotation "PartitionableKey" in C#.

Vorgehensweise

- Java PartitionKey-Annotationen für ein Feld in einer Java-Anwendung definieren. Java

```
class Employee {
    int empId;

    @PartitionKey(order = 0)
    int deptId;

}
```

  Sie können PartitionKey-Annotationen für mehrere Schlüssel definieren, oder Sie können den PartitionKey-Alias für eine Klasse definieren. Weitere Beispiele zum Definieren von PartitionKey-Annotationen in Java-Anwendungen finden Sie unter Java API documentation: Annotation Type PartitionKeys.

- .NET PartitionKey-Attribute für ein Feld in einer .NET-Anwendung definieren.

```
class Employee {
    int empId;

    [PartitionKey]
    int deptId;
}
```

  Sie können PartitionKey-Attribute auch für .NET-Klassen definieren. Weitere Informationen finden Sie unter .NET API documentation: PartitionKeyAttribute Class.

Java- und C#-Datentypäquivalente

Wenn Sie Unternehmensdatengridanwendungen entwickeln, müssen die Datentypen Ihrer Java- und C#-Anwendungen kompatibel sein.

Tabelle 8. Datentypäquivalente von Java und C#

Java-Typ	C#-Typ
boolean	bool
java.lang.Boolean	bool
Byte	sbyte oder byte
java.lang.Byte	sbyte
short	short, ushort
java.lang.Short	short, ushort
int	int, uint, ushort
java.lang.Integer	int, uint
long	long, ulong, uint
java.lang.Long	long, ulong, uint
short oder int	ushort
java.lang.Short oder java.lang.Integer	ushort
int oder long	uint
java.lang.Integer oder java.lang.Long	uint
long oder BigInteger	ulong
java.lang.Long oder java.lang.BigInteger	ulong
char, java.lang.Character	char
float, java.lang.Float	float
double, java.lang.Double	double
java.math.BigDecimal	decimal
java.math.BigInteger	decimal, long oder ulong
java.lang.String	string
java.util.Date, java.util.Calendar	System.DateTime
java.util.Date(rounding), java.util.Calendar(rounding)	System.DateTime
java.util.GregorianCalendar	

Tabelle 8. Datentypäquivalente von Java und C# (Forts.)

Java-Typ	C#-Typ
java.util.ArrayList	System.Collections.ArrayList, System.Collections.Generic.List, System.Collections.SortedList
java.util.HashMap	System.Collections.Generic.Dictionary, System.Collections.Hashtable
java.util.LinkedList	System.Collections.Generic.LinkedList
java.util.ArrayList, java.util.Vector	System.Collections.Generic.List
java.util.Stack	System.Collections.Generic.Stack
java.util.Vector	System.Collections.ArrayList, System.Collections.Generic.List

Eigenständige Server starten (XIO)

Wenn Sie eine eigenständige Konfiguration verwenden, setzt sich die Umgebung aus Katalogservern, Container-Servern und Clientprozessen zusammen. Server von WebSphere eXtreme Scale können mit Hilfe der integrierten Server-API auch in vorhandene Java-Anwendungen integriert werden. Sie müssen diese Prozesse manuell konfigurieren und starten.

Vorbereitende Schritte

Sie können Server von WebSphere eXtreme Scale in einer Umgebung ohne WebSphere Application Server starten. Wenn Sie WebSphere Application Server verwenden, lesen Sie den Abschnitt WebSphere eXtreme Scale mit WebSphere Application Server konfigurieren.

IBM eXtremeIO (XIO) optimieren

Sie können XIO-Servereigenschaften verwenden, um das Verhalten des XIO-Transports im Datengrid zu optimieren.

Servereigenschaften für die XIO-Optimierung

Sie können die folgenden Eigenschaften in der Servereigenschaftendatei definieren:

maxXIONetworkThreads
Legt die maximale Anzahl an Threads fest, die dem Thread-Pool des eXtremeIO-Transportnetzes zugeordnet werden.

Standardeinstellung: `50`

minXIONetworkThreads
Legt die Mindestanzahl an Threads fest, die dem Thread-Pool des eXtremeIO-Transportnetzes zugeordnet werden.

Standardeinstellung: `50`

maxXIOWorkerThreads
Legt die maximale Anzahl an Threads fest, die dem Thread-Pool für die Verarbeitung von eXtremeIO-Transportanforderungen zugeordnet werden.

Standardeinstellung: `128`

minXIOWorkerThreads

Legt die Mindestanzahl an Threads fest, die dem Thread-Pool für die Verarbeitung von eXtremeIO-Transportanforderungen zugeordnet werden.

Standardeinstellung: 128

8.6+ Transport

Gibt den für alle Server in der Katalogservicedomäne zu verwendenden Transporttyp an. Sie können die Eigenschaft auf `XIO` oder `ORB` setzen.

Wenn Sie den Befehl **`startOgServer`** oder **`startXsServer`** verwenden, müssen Sie diese Eigenschaft nicht setzen. Das Script überschreibt diese Eigenschaft. Wenn Sie Server jedoch mit einer anderen Methode starten, wird der Wert dieser Eigenschaft verwendet.

Diese Eigenschaft gilt nur für den Katalogservice.

Wenn Sie den Parameter **`-transport`** im Startscript und die Servereigenschaft **`transport`** auf einem Katalogserver definieren, wird der Wert des Parameters **`-transport`** verwendet.

8.6+ xioTimeout

Legt das Zeitlimit für Serveranforderungen (in Sekunden) fest, die den Transport IBM eXtremeIO (XIO) verwenden. Sie können einen beliebigen Wert größer-gleich 1 Sekunde angeben.

Standardeinstellung: 30 Sekunden

Szenario: Datengrid in eXtreme Scale sichern

In Datengrids von WebSphere eXtreme Scale werden sensible Daten gespeichert, die geschützt werden müssen.

Vorbereitende Schritte

- Installieren Sie das Produkt. Sie müssen die Serverlaufzeitumgebung und die Clients installieren. Für Clients können Sie Java- und .NET-Clients verwenden. Weitere Informationen finden Sie unter Installation.
- Wenn Sie in Upgrade von einem früheren Release durchführen, müssen alle Container- und Katalogserver dasselbe Release-Level haben. Weitere Informationen finden Sie unter Upgrade und Migration von WebSphere eXtreme Scale durchführen.

Informationen zu diesem Vorgang

Für eine sichere Implementierung verwenden Sie mehrere Zugriffsschutzebenen, um eine optimale Sicherheit zu erhalten. Das erste Element des Zugriffsschutzes ist die Verwendung von Firewalls zur Segmentierung des Netzes. Das Standardschichtenmodell für Webanwendungen setzt sich aus Web-Clients, einer Darstellungsschicht von HTTP-Servern, einer Anwendungsschicht aus Anwendungsservern, einer Datenschicht und einer Speicherschicht zusammen.

Die Daten-Grid-Server von eXtreme Scale werden in der Datenschicht implementiert. Standardmäßig werden die Server der Darstellungsschicht in eine Demilitarized Zone (DMZ) gestellt, die durch eine Firewall geschützt ist, und die Anwendungs-, Daten- und Speicherschichten in Netzsegmente, die durch weitere Firewalls geschützt sind. Implementieren Sie eXtreme-Scale-Server nicht in einer DMZ. eXtreme-Scale-Server müssen gemäß Standardbranchenpraxis wie alle Elemente der Datenschicht geschützt werden.

Um jedoch einen optimalen Schutz vor Sicherheitsbedrohung zu erreichen, verwenden Sie einen umfassenden Abwehrmechanismus, der eine Reihe weiterer Maßnahmen besitzt, um die eXtreme-Scale-Operationen und die Daten, die im Datengrid gespeichert werden, zu schützen. Diese zusätzlichen Maßnahmen dienen nicht nur der Abwehr von Sicherheitsbedrohungen, sondern verhindern auch den unbefugten Datenzugriff durch Mitarbeiter und Auftragnehmer, die Zugriff auf Netzsegmente haben, in denen sich die eXtreme-Scale-Server befinden.

Verwenden Sie die folgenden durchgängigen Schritte, um die Sicherheit in WebSphere eXtreme Scale zu konfigurieren, je nachdem, ob Sie eigenständige Server, das Liberty-Profil, das OSGi-Framework oder WebSphere Application Server in Ihrer Umgebung installiert haben:

eXtreme-Scale-Verbindungen zwischen Servern authentifizieren

Verbindungen zwischen Servern müssen authentifiziert werden, um den Zugriff auf das Datengrid durch nicht berechtigte Server zu verhindern.

Nächste Schritte

„Anforderungen von Clients an Server authentifizieren” auf Seite 209

eXtreme-Scale-Serververbindungen in eigenständigen Umgebungen authentifizieren

Verbindungen zwischen eXtreme-Scale-Servern müssen authentifiziert werden, um den Zugriff auf das Datengrid durch nicht berechtigte Server zu verhindern.

Informationen zu diesem Vorgang

Die folgenden Einstellungen in der Datei `server.properties` bestimmen, wie Server einander authentifizieren:

- **`securityEnabled=true`**
- **`secureTokenManagerType=autoSecret`**
- **`authenticationSecret=OurGridServersExampleSecret`**

Alle eXtreme-Scale-Server in einer Domäne sowie alle Server in verknüpften Domänen müssen dieselben Werte für diese drei Eigenschaften in der Datei `server.properties` verwenden. Andernfalls schlägt die Kommunikation fehl. Weitere Informationen zum Festlegen dieser Eigenschaften in der Servereigenschaftendatei finden Sie unter Servereigenschaftendatei.

Vorgehensweise

1. Aktivieren Sie die Server-Server-Authentifizierung. Setzen Sie die Eigenschaft "securityEnabled" auf `true`, z. B.:

   ```
   securityEnabled=true
   ```

   Der Standardwert für diese Eigenschaft ist `false`.
2. Erstellen Sie eine sichere Serverkonfiguration.

   secureTokenManagerType ist eine Eigenschaft, die Sie in der Servereigenschaftendatei definieren.

   Ein Wert für "secureTokenManagerType", den Sie für eine sichere Konfiguration verwenden können, ist "autoSecret". Wenn Sie "autoSecret" angeben, wird eine Tokenverschlüsselung und -signatur mithilfe von Schlüsseln durchgeführt, die von der Eigenschaft "authenticationSecret" abgeleitet werden. Sichere Token

werden bei der Server-Server-Authentifizierung und auch für Client-SSO-Token verwendet. Der Wert `none` für "secureTokenManagerType" ist nicht sicher, weil diese Einstellung die Erstellung verschlüsselter Token verhindert.

Sie können auch die Einstellung "secureTokenManagerType=default" angeben. Diese Option erfordert jedoch die Konfiguration von Keystore- und zugehörigen Artefakten.

3. Geben Sie einen langen Zeichenfolgewert für "authenticationSecret" an (ein einziges Wort!!!), das sich nur schwer erraten lässt. Sie können diesen Wert mit dem Dienstprogramm "FilePasswordEncoder" codieren. Weitere Informationen finden Sie unter „Sicherheitsartefakte für autorisierte Benutzer speichern” auf Seite 228. Verwenden Sie nicht die Eigenschaft "ObjectGridDefaultSecret". Dies ist der Wert, der in der Datei `sampleServer.properties` verwendet wird.

Ergebnisse

Wenn Sie einen eigenständigen eXtreme-Scale-Server starten, geben Sie den Namen der Eigenschaftendatei in der Befehlszeile an. Wenn Sie die Servereigenschaftendatei angeben, werden die Authentifizierungseigenschaften, die Sie hinzugefügt haben, beim Start des Servers geladen. Weitere Informationen finden Sie unter Sichere Server in einer eigenständigen Umgebung starten.

Nächste Schritte

„Clientanforderungen in eigenständigen Umgebungen authentifizieren” auf Seite 209

Authentifizierung von eXtreme-Scale-Serververbindungen im Liberty-Profil

Verbindungen zwischen eXtreme-Scale-Servern im Liberty-Profil müssen authentifiziert werden, um zu verhindern, dass nicht berechtigte Server auf das Datengrid zugreifen.

Informationen zu diesem Vorgang

Die folgenden Einstellungen in der Datei `server.properties` bestimmen, wie Server einander authentifizieren:

- **`securityEnabled=true`**
- **`secureTokenManagerType=autoSecret`**
- **`authenticationSecret=OurGridServersExampleSecret`**

Alle eXtreme-Scale-Server in einer Domäne sowie alle Server in verknüpften Domänen müssen dieselben Werte für diese Eigenschaften in der Datei `server.properties` verwenden. Andernfalls schlägt die Kommunikation fehl.

Vorgehensweise

1. Aktivieren Sie die Server-Server-Authentifizierung. Setzen Sie die Eigenschaft "securityEnable" auf `true`, z. B.:

   ```
   securityEnabled=true
   ```

   Der Standardwert für diese Eigenschaft ist `false`.
2. Erstellen Sie eine sichere Serverkonfiguration. Ein Wert für "secureTokenManagerType", den Sie für eine sichere Konfiguration verwenden können, ist "autoSecret". Wenn Sie "autoSecret" angeben, wird eine Tokenverschlüsselung und -signatur mithilfe von Schlüsseln durchgeführt, die von der Eigenschaft "authenticationSecret" abgeleitet werden. Sichere Token werden bei der gegenseiti-

gen Serverauthentifizierung und auch für Client-SSO-Token verwendet. Der Wert `none` für "secureTokenManagerType" ist nicht sicher, weil diese Einstellung die Erstellung verschlüsselter Token verhindert.

Sie können auch die Einstellung "secureTokenManagerType=default" angeben. Diese Option erfordert jedoch die Konfiguration von Keystore- und zugehörigen Artefakten.

3. Geben Sie ein langes und verschlüsseltes Authentifizierungssecret an, das für andere nur schwer zu entschlüsseln ist. Verwenden Sie nicht das ObjectGridDefaultSecret. Dies ist der Wert, der in der Datei `sampleServer.properties` verwendet wird.
4. Konfigurieren Sie die Datei `server.xml` mit derselben Konfiguration, die Sie für eine eigenständige Serverkonfiguration verwenden können. Geben Sie in der Datei `server.xml` den Dateipfad zur Eigenschaftendatei in einem Attribut "serverProps" im Element `xsSever` an. Im Folgenden sehen Sie ein Beispiel aus der Datei `server.xml`:

```
<server>
...
<xsSever ... serverProps="/Pfad/zu/myServerProps.properties" ... />
</server>
```

Nächste Schritte

„Clientanforderungen im Liberty-Profil authentifizieren" auf Seite 210

Authentifizierung von eXtreme-Scale-Serververbindungen im OSGi-Framework

Verbindungen zwischen eXtreme-Scale-Servern im OSGi-Framework müssen authentifiziert werden, um zu verhindern, dass nicht berechtigte Server auf das Datengrid zugreifen.

Vorbereitende Schritte

Sie müssen das OSGi-Framework installieren, bevor Sie das Datengrid sichern. Weitere Informationen finden Sie unter „Eclipse-Equinox-OSGi-Framework mit Eclipse Gemini für Clients und Server installieren" auf Seite 232.

Informationen zu diesem Vorgang

Die folgenden Einstellungen in der Datei `server.properties` bestimmen, wie Server einander authentifizieren:

- **`securityEnabled=true`**
- **`secureTokenManagerType=autoSecret`**
- **`authenticationSecret=OurGridServersExampleSecret`**

Alle eXtreme-Scale-Server in einer Domäne sowie alle Server in verknüpften Domänen müssen dieselben Werte für diese Eigenschaften in der Datei `server.properties` verwenden. Andernfalls schlägt die Kommunikation fehl.

Vorgehensweise

1. Aktivieren Sie die Server-Server-Authentifizierung. Setzen Sie die Eigenschaft **`securityEnabled`** in der Servereigenschaftendatei auf `true`. Beispiel:

   ```
   securityEnabled=true
   ```

   Der Standardwert für diese Eigenschaft ist `false`.

2. Erstellen Sie eine sichere Serverkonfiguration. Ein Wert für "secureTokenManagerType", den Sie für eine sichere Konfiguration verwenden können, ist "autoSecret". Wenn Sie "autoSecret" angeben, wird eine Tokenverschlüsselung und -signatur mithilfe von Schlüsseln durchgeführt, die von der Eigenschaft "authenticationSecret" abgeleitet werden. Sichere Token werden bei der gegenseitigen Serverauthentifizierung und auch für Client-SSO-Token verwendet. Der Wert `none` für "secureTokenManagerType" ist nicht sicher, weil diese Einstellung die Erstellung verschlüsselter Token verhindert.

   Sie können auch die Einstellung "secureTokenManagerType=default" angeben. Diese Option erfordert jedoch die Konfiguration von Keystore- und zugehörigen Artefakten.
3. Geben Sie einen langen Zeichenfolgewert für das Element "authenticationSecret" an. Dieser Wert sollte für andere schwer zu erraten sein. Sie können diesen Wert mit dem Dienstprogramm "FilePasswordEncoder" codieren. Verwenden Sie nicht das Element "ObjectGridDefaultSecret". Dies ist der Wert, der in der Datei `sampleServer.properties` verwendet wird.
4. Verweisen Sie auf die Servereigenschaftendatei. Erstellen Sie mit den folgenden Befehlen eine verwaltete Server-PID (Persistente ID) für die Servereigenschaftendatei in der OSGi-Konsole:

```
osgi> cm create com.ibm.websphere.xs.server
osgi> cm put com.ibm.websphere.xs.server objectgrid.server.props /mypath/server.properties
```

Nächste Schritte

„Clientanforderungen im OSGi-Framework authentifizieren” auf Seite 212

eXtreme-Scale-Serververbindungen in WebSphere Application Server authentifizieren

Die eXtreme-Scale-Server, die unter WebSphere Application Server ausgeführt werden, authentifizieren sich gegenseitig genauso wie eigenständige eXtreme-Scale-Server.

Vorbereitende Schritte

Informationen zu diesem Vorgang

Es gibt drei Einstellungen in der Datei `server.properties`, die bestimmen, wie sich Server gegenseitig authentifizieren. Alle eXtreme-Scale-Server in einer Domäne sowie alle Server in verknüpften Domänen müssen dieselben Werte für diese drei Eigenschaften in der Datei `server.properties` verwenden. Andernfalls schlägt die Kommunikation fehl. Weitere Informationen zu den Sicherheitseigenschaften finden Sie unter XML-Sicherheitsdeskriptordatei.

Vorgehensweise

1. Erstellen Sie die Servereigenschaftendatei, und aktivieren Sie die gegenseitige Serverauthentifizierung. Erstellen Sie mithilfe der Beispielservereigenschaftendatei eine Servereigenschaftendatei, die die Eigenschaft **`securityEnabled`** mit dem Wert `true` enthält, z. B.:

   ```
   securityEnabled=true
   ```

   Der Standardwert für diese Eigenschaft ist `false`.
2. Konfigurieren Sie eine sichere Serverkonfiguration. Ein Wert für "secureTokenManagerType", den Sie für eine sichere Konfiguration verwenden können, ist "autoSecret". Wenn Sie "autoSecret" angeben, wird eine Tokenverschlüsselung und -signatur mithilfe von Schlüsseln durchgeführt, die von der Eigenschaft

"authenticationSecret" abgeleitet werden. Sichere Token werden bei der gegenseitigen Serverauthentifizierung und auch für Client-SSO-Token verwendet. Der Wert `none` für "secureTokenManagerType" ist nicht sicher, weil diese Einstellung die Erstellung verschlüsselter Token verhindert.

Sie können auch die Einstellung "secureTokenManagerType=default" angeben. Diese Option erfordert jedoch die Konfiguration von Keystore- und zugehörigen Artefakten.

3. Geben Sie ein langes und verschlüsseltes Authentifizierungssecret an, das für andere nur schwer zu entschlüsseln ist. Verwenden Sie nicht das ObjectGridDefaultSecret. Dies ist der Wert, der in der Datei `sampleServer.properties` verwendet wird.
4. Konfigurieren Sie eine Servereigenschaftendatei, um den Server zu sichern. Konfigurieren Sie diese Eigenschaftendatei mit der Administrationskonsole von WebSphere Application Server. Klicken Sie auf **WebSphere-Anwendungsserver** > *Servername* > **Java- und Prozessverwaltung** > **Prozessdefinition** > **Java Virtual Machine**. Fügen Sie das folgende generische JVM-Argument hinzu:

   ```
   -Dobjectgrid.server.props=<Name der Servereigenschaftendatei>
   ```

Nächste Schritte

„Clientanforderungen in WebSphere Application Server authentifizieren" auf Seite 213

Anforderungen von Clients an Server authentifizieren

Ihre Clientanwendungen müssen sichere Anforderungen über das Netz senden.

Nächste Schritte

„Zugriff auf das Datengrid autorisieren" auf Seite 215

Clientanforderungen in eigenständigen Umgebungen authentifizieren

Sofern Client nicht authentifiziert werden, ist der Zugriff auf Griddaten und JMX-Verwaltungsoperationen, die das Grid steuern, ungeschützt. Dies gilt auch, wenn SSL aktiviert ist.

Informationen zu diesem Vorgang

Das Authentifizierungsverhalten, dass eXtreme-Scale-Server von eXtreme-Scale-Clients erfordern, wird durch die Einstellung **`credentialAuthentication=required`** in der Datei `server.properties` bestimmt.

Wenn "credentialAuthentication" auf `Required` oder `Supported` gesetzt ist, ist eine weitergehende Konfiguration erforderlich, die in den folgenden Schritten beschrieben ist. Diese Schritte sind ausführlicher mit Beispielen zu den Änderungen in den Konfigurationsdateien im Abschnitt Lernprogramm zur Java-SE-Sicherheit - Schritt 3 beschrieben.

Vorgehensweise

- Verweisen Sie auf eine XML-Sicherheitsdeskriptordatei in jedem Katalogserver.

  Wenn der Katalogserver in einer eigenständigen Umgebung gestartet wird, können Sie mit dem Parameter `-clusterSecurityFile` des Befehls **`startXsServer`** oder **`startOgServer`** auf diese Datei verweisen.

Zum Aktivieren der Sicherheit muss diese Datei `securityEnabled="true"` im Element "security" enthalten. Die XML-Sicherheitsdeskriptordatei muss außerdem einen Deskriptor des Authentifikators enthalten, den Sie verwenden möchten. WebSphere eXtreme Scale enthält die Authentifikatoren `LDAPAuthenticator`, `KeyStoreLoginAuthenticator` und `WSTokenAuthenticator`. Der Authentifikator `WSTokenAuthenticator` kann in den eigenständigen Umgebungen nicht verwendet werden. Sie können diesen Authentifikator nur verwenden, wenn sowohl eXtreme-Scale-Clients als auch eXtreme-Scale-Server mit WebSphere Application Server ausgeführt werden. Alternativ können Sie angepasste Authentifikatoren und Anmeldemodule entsprechend den Schnittstellen entwickeln, die in der API-Dokumentation beschrieben sind.

- Verweisen Sie mit dem JVM-Argument `-Djava.security.auth.login.config="path_name"` auf eine JAAS-Konfigurationsdatei in jedem Katalogserver und jedem Container-Server. Informationen zum Erstellen dieser Dateien und zum Konfigurieren der Verwendung dieser Dateien in eXtreme-Scale-Servern finden Sie im Lernprogramm Lernprogramm: Sicherheit von Java SE konfigurieren. Die JAAS-Konfigurationsdatei gibt ein LoginModule an. Sie können das KeyStoreLoginModule mit dem KeyStoreLoginAuthenticator verwenden. Verwenden Sie SimpleLDAPLoginModule für den LDAPAuthenticator. Weitere Informationen finden Sie unter LDAP-Authentifizierung in Katalog- und Container-Servern von eXtreme Scale aktivieren für eXtreme-Scale-Container und -Katalogserver bzw. unter Keystore-Authentifizierung in Container- und Katalogservern von eXtreme Scale aktivieren.
- Konfigurieren Sie den Client so, dass die für die Authentifizierung erforderlichen Berechtigungsnachweise übergeben werden. Gewöhnlich werden dazu Werte in einer Clienteigenschaftendatei angegeben. Weitere Informationen zum Aktivieren der LDAP-Authentifizierung in eXtreme-Scale-Clients finden Sie unter LDAP-Authentifizierung in Katalog- und Container-Servern von eXtreme Scale aktivieren. Weitere Informationen zum Aktivieren der Keystore-Authentifizierung in eXtreme-Scale-Clients finden Sie unter Keystore-Authentifizierung in Container- und Katalogservern von eXtreme Scale aktivieren.

Nächste Schritte

„Zugriff auf das Datengrid in eigenständigen Umgebungen autorisieren" auf Seite 215

Clientanforderungen im Liberty-Profil authentifizieren

Sofern Client nicht authentifiziert werden, ist der Zugriff auf Griddaten und JMX-Verwaltungsoperationen, die das Grid steuern, ungeschützt. Dies gilt auch, wenn SSL im Liberty-Profil aktiviert ist.

Informationen zu diesem Vorgang

Das von eXtreme-Scale-Clients geforderte Authentifizierungsverhalten wird durch die Einstellung **`credentialAuthentication=required`** in der Datei `server.properties`, die Einstellung **`KeyStoreLogin`** in der JAAS-Konfigurationsdatei `og_jaas.config` und die Einstellung **`KeyStoreLoginAuthenticator`** in der Datei `security.xml` bestimmt.

Die Servereigenschaftendatei wird geladen, indem sie in der Datei `server.xml` wie in „Authentifizierung von eXtreme-Scale-Serververbindungen im Liberty-Profil" auf Seite 206 beschrieben referenziert wird. Aus Gründen der Sicherheit muss diese Datei wie in eigenständigen Umgebungen `credentialAuthentication=Required` enthalten.

Jede der Konfigurationsdateien wird von jedem Katalogserver geladen. Container-Server verwenden nur die JAAS-Konfigurationsdatei und die Implementierungsdeskriptordateien für die Sicherheit.

Verwenden Sie eine der folgenden Methoden, um Clients zu authentifizieren.

Vorgehensweise

- Verweisen Sie auf eine XML-Sicherheitsdeskriptordatei in jedem Katalogserver.

  Wenn der Katalogserver das Liberty-Profil ist, können Sie mit dem Attribut `clusterSecurityURL=` in der Datei `server.xml` auf diese Datei verweisen. Sehen Sie sich das folgende Beispiel an, in dem die Datei `objectGridSecurity.xml` die XML-Sicherheitsdeskriptordatei ist:

  ```
  <server description="new server">
  <!-- Features aktivieren -->
  <featureManager>
  <feature>eXtremeScale.server-1.1</feature>
  </featureManager>

  <xsServer
  isCatalog="true"
  serverProps="server.xs.props"
  clusterSecurityURL="file://C:/wlp/usr/servers/objectGridSecurity.xml"
  />
  </server>
  ```

  Zum Aktivieren der Sicherheit muss diese Datei `securityEnabled="true"` im Element "security" enthalten. Die XML-Sicherheitsdeskriptordatei muss außerdem einen Deskriptor des Authentifikators enthalten, den Sie verwenden möchten. WebSphere eXtreme Scale enthält die Authentifikatoren `LDAPAuthenticator`, `KeyStoreLoginAuthenticator` und `WSTokenAuthenticator`.

- Verweisen Sie mit dem JVM-Argument `-Djava.security.auth.login.config="path_name"` in der Datei `jvm.options` auf eine JAAS-Konfigurationsdatei in jedem Katalogserver und jedem Container-Server.

  Editieren oder erstellen Sie die Datei `jvm.options` im Verzeichnis `WLP-Installationsverzeichnis/usr/servers/<server_name>`.

  Anmerkung: Wenn Sie eine Datei `jvm.options` auf Serverkonfigurationsebene erstellen müssen, müssen Sie die Version in die Datei `WLP-Installationsstammverzeichnis/etc/jvm.options` kopieren. Die Datei `jvm.options` enthält einige Optionen, die eXtreme Scale für die Ausführung im Liberty-Profil benötigt.

  Wenn Sie eine Datei `jvm.options` auf Serverebene erstellen und das JVM-Argument zur Referenzierung der JAAS-Konfigurationsdatei eingeben, sehen Ihre Dateien `jvm.options` wie folgt aus:

  ```
  C:\wlp\usr\servers\simpCatalog>cat jvm.options
  -Dorg.osgi.framework.bootdelegation=com.ibm.wsspi.runtime
  -Djava.endorsed.dirs=C:\wlp\wxs\lib\endorsed
  -Djava.security.auth.login.config=C:\wlp\usr\servers\ogjaas.config
  ```

  Informationen zum Erstellen dieser Dateien und zum Konfigurieren der Verwendung dieser Dateien in eXtreme-Scale-Servern finden Sie im Lernprogramm Lernprogramm: Sicherheit von Java SE konfigurieren. In der JAAS-Konfigurationsdatei ist ein LoginModule angegeben. Sie können das Modul "KeyStoreLoginModule" mit dem Authentifikator "KeyStoreLoginAuthenticator" verwenden. Verwenden Sie das Modul "SimpleLDAPLoginModule" für den Authentifikator "LDAPAuthenticator". Weitere Informationen finden Sie unter LDAP-Authentifi-

zierung in Katalog- und Container-Servern von eXtreme Scale aktivieren für eXtreme-Scale-Container und -Katalogserver bzw. unter Keystore-Authentifizierung in Container- und Katalogservern von eXtreme Scale aktivieren.

- Konfigurieren Sie den Client so, dass die für die Authentifizierung erforderlichen Berechtigungsnachweise übergeben werden. Gewöhnlich werden dazu Werte in einer Clienteigenschaftendatei angegeben. Weitere Informationen zum Aktivieren der LDAP-Authentifizierung in eXtreme-Scale-Clients finden Sie unter LDAP-Authentifizierung in Katalog- und Container-Servern von eXtreme Scale aktivieren. Weitere Informationen zum Aktivieren der Keystore-Authentifizierung in eXtreme-Scale-Clients finden Sie unter Keystore-Authentifizierung in Container- und Katalogservern von eXtreme Scale aktivieren.

Nächste Schritte

„Zugriff auf das Datengrid im Liberty-Profil autorisieren" auf Seite 216

Clientanforderungen im OSGi-Framework authentifizieren

Sofern Client nicht authentifiziert werden, ist der Zugriff auf Griddaten und JMX-Verwaltungsoperationen, die das Grid steuern, ungeschützt. Dies gilt auch, wenn SSL im OSGi-Framework aktiviert ist.

Vorbereitende Schritte

Sie müssen das OSGi-Framework installieren, bevor Sie das Datengrid sichern. Weitere Informationen finden Sie unter „Eclipse-Equinox-OSGi-Framework mit Eclipse Gemini für Clients und Server installieren" auf Seite 232.

Informationen zu diesem Vorgang

Das von eXtreme-Scale-Clients geforderte Authentifizierungsverhalten wird durch die Einstellung **credentialAuthentication=required** in der Datei `server.properties`, die Einstellung **KeyStoreLogin** in der JAAS-Konfigurationsdatei `og_jaas.config` und die Einstellung **KeyStoreLoginAuthenticator** in der Datei `security.xml` bestimmt.

Verwenden Sie eine der folgenden Methoden, um Clients zu authentifizieren.

Vorgehensweise

- Verweisen Sie mit dem JVM-Argument "-DclusterSecurityFile="path_name"" auf eine XML-Sicherheitsdeskriptordatei in jedem Katalogserver.

  Verwenden Sie dieses JVM-Argument in der OSGi-Befehlszeile, wenn Sie den Katalogserver starten.

  Zum Aktivieren der Sicherheit muss diese Datei `securityEnabled="true"` im Element "security" enthalten. Die XML-Sicherheitsdeskriptordatei muss außerdem einen Deskriptor des Authentifikators enthalten, den Sie verwenden möchten. WebSphere eXtreme Scale enthält die Authentifikatoren `LDAPAuthenticator`, `KeyStoreLoginAuthenticator` und `WSTokenAuthenticator`. Der Authentifikator `WSTokenAuthenticator` kann in den eigenständigen Umgebungen nicht verwendet werden. Sie können diesen Authentifikator nur verwenden, wenn sowohl eXtreme-Scale-Clients als auch eXtreme-Scale-Server mit WebSphere Application Server ausgeführt werden. Alternativ können Sie angepasste Authentifikatoren und Anmeldemodule entsprechend den Schnittstellen entwickeln, die in der API-Dokumentation beschrieben sind.
- Verweisen Sie mit dem JVM-Argument `-Djava.security.auth.login.config="path_name"` auf eine JAAS-Konfigurations-

datei in jedem Katalogserver und jedem Container-Server. Informationen zum Erstellen dieser Dateien und zum Konfigurieren der Verwendung dieser Dateien in eXtreme-Scale-Servern finden Sie im Lernprogramm Lernprogramm: Sicherheit von Java SE konfigurieren. Die JAAS-Konfigurationsdatei gibt ein LoginModule an. Sie können das KeyStoreLoginModule mit dem KeyStoreLoginAuthenticator verwenden. Verwenden Sie SimpleLDAPLoginModule für den LDAPAuthenticator. Weitere Informationen finden Sie unter LDAP-Authentifizierung in Katalog- und Container-Servern von eXtreme Scale aktivieren für eXtreme-Scale-Container und -Katalogserver bzw. unter Keystore-Authentifizierung in Container- und Katalogservern von eXtreme Scale aktivieren.

- Konfigurieren Sie den Client so, dass die für die Authentifizierung erforderlichen Berechtigungsnachweise übergeben werden. Gewöhnlich werden dazu Werte in einer Clienteigenschaftendatei angegeben. Weitere Informationen zum Aktivieren der LDAP-Authentifizierung in eXtreme-Scale-Clients finden Sie unter LDAP-Authentifizierung in Katalog- und Container-Servern von eXtreme Scale aktivieren. Weitere Informationen zum Aktivieren der Keystore-Authentifizierung in eXtreme-Scale-Clients finden Sie unter Keystore-Authentifizierung in Container- und Katalogservern von eXtreme Scale aktivieren.

Nächste Schritte

„Zugriff auf das Datengrid im OSGi-Framework autorisieren" auf Seite 217

Clientanforderungen in WebSphere Application Server authentifizieren

Anforderungen, die WebSphere Application Server vom eXtreme-Scale-Datengrid empfängt, müssen authentifiziert werden.

Vorbereitende Schritte

Die Authentifizierungsanforderungen für eXtreme-Scale-Client werden durch die Einstellungen in der Servereigenschaftendatei bestimmt. Im Verzeichnis *WAS-Stammverzeichnis*/optionalLibraries/ObjectGrid/properties/sampleServer.properties wird eine Beispielservereigenschaftendatei bereitgestellt.

Informationen zu diesem Vorgang

Sie müssen die Authentifizierung für eXtreme-Scale-Server, die unter WebSphere Application Server ausgeführt werden, wie folgt konfigurieren.

Vorgehensweise

1. Erstellen Sie die Servereigenschaftendatei. Erstellen Sie mithilfe dieser Beispielservereigenschaftendatei eine Servereigenschaftendatei, die die folgenden Zeilen enthält:

   ```
   securityEnabled=true
   credentialAuthentication=Required
   ```

   Sofern die Eigenschaft "credentialAuthentication=Required" nicht vorhanden ist, ist das Grid nicht sicher, und nicht authentifizierte Benutzer können Gridoperationen ausführen.

   Einschränkung: Die Eigenschaft "credentialAuthentication=Required" kann für den dynamischen Cache-Provider nicht angegeben werden.

2. Erstellen Sie die XML-Sicherheitsdeskriptordatei. Wenn die Eigenschaft "credentialAuthentication" auf "Required" oder "Supported" gesetzt ist, müssen Sie eine XML-Sicherheitsdeskriptordatei angeben. Sehen Sie sich das folgende Beispiel an:

```
<securityConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ibm.com/ws/objectgrid/config/security ../objectGridSecurity.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config/security">

 <security securityEnabled="true">
<authenticator className ="com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenAuthenti
  </authenticator>
 </security>
</securityConfig>
```

   Die XML-Sicherheitsdeskriptordatei gibt den zu verwendenden Authentifikator an. Wenn alle eXtreme-Scale-Clients und -Server unter WebSphere Application Server ausgeführt werden, können Sie den Authentifikator "WSTokenAuthenticator" verwenden. Mit eXtreme Scale werden noch zwei weitere Authentifikatoren bereitgestellt: KeyStoreLoginAuthenticator und LDAPLoginAuthenticator. Weitere Informationen zum Konfigurieren der LDAP-Authentifizierung für eXtreme Scale finden Sie unter LDAP-Authentifizierung in Katalog- und Container-Servern von eXtreme Scale aktivieren. Wenn Sie die Keystore- und Anmeldeauthentifikatoren mit eXtreme Scale unter WebSphere Application Server verwenden möchten, ist eine JAAS-Konfiguration erforderlich. Weitere Informationen zum Konfigurieren der Keystore-Authentifizierung für eXtreme Scale finden Sie unter Keystore-Authentifizierung in Container- und Katalogservern von eXtreme Scale aktivieren.
3. Erstellen Sie die JAAS-Konfiguration, sofern Sie den Authentifikator "WSTokenAuthenticator" nicht verwenden.
4. Verweisen Sie in jedem Katalogserver mit den folgenden JVM-Argumenten auf die Servereigenschaftendatei. Konfigurieren Sie diese Eigenschaften über die Administrationskonsole von WebSphere Application Server. Klicken Sie auf **Server** > **Alle Server** > **Servername** > **Prozessdefinition** > **Generische JVM-Argumente**. Die folgenden Argumente sind erforderlich:

   ```
   -Dobjectgrid.server.props=<Name der Servereigenschaftendatei>
   -Dobjectgrid.cluster.security.xml.url=file://<XML-Sicherheitsdeskriptordatei>
   ```
5. Verweisen Sie in jedem Container-Server mit dem folgenden JVM-Argument auf die Servereigenschaftendatei:

   ```
   -Dobjectgrid.server.props=<Name der Servereigenschaftendatei>
   ```

Nächste Schritte

Die Server von WebSphere eXtreme Scale müssen so konfiguriert werden, dass entsprechende Berechtigungsnachweise übergeben werden. Führen Sie diese Konfiguration über die Clienteigenschaftendatei durch. Sehen Sie sich den folgenden Beispielauthentifikator "WSTokenAuthenticator" an:

```
securityEnabled=true
credentialAuthentication=supported
credentialGeneratorClass=com.ibm.websphere.security.plugins.builtins.WSTokenCredentialGenerator
```

Die Verwendung dieser Datei muss im Client konfiguriert werden. Wenn der Client unter WebSphere Application Server ausgeführt wird, konfigurieren Sie den Client mit dem folgenden JVM-Argument:

```
-Dobjectgrid.client.props=<Clienteigenschaftendatei>
```

Zum Sichern der Gridimplementierung definieren Sie die Anwendungssicherheit und die Java-2-Sicherheit für die Server von WebSphere Application Server, die eX-

treme-Scale-Server hosten. Verwenden Sie Anzeige für die Sicherheitskonfiguration in der Administrationskonsole von WebSphere Application Server, um diese Einstellungen zu aktivieren.

Jetzt können Sie mit dem nächsten Schritt, „Zugriff auf das Datengrid in WebSphere Application Server autorisieren" auf Seite 218, fortfahren.

Zugriff auf das Datengrid autorisieren

Sie können eine Zugriffssteuerung erzwingen, sodass authentifizierte IDs nur Operationen ausführen können, für die sie eigens autorisiert wurden.

Nächste Schritte

„Zugriff für spezielle Verwaltungsoperationen autorisieren" auf Seite 219

Zugriff auf das Datengrid in eigenständigen Umgebungen autorisieren

Mit der Richtliniendatei können Sie können steuern, welche Benutzer spezielle Berechtigungen für den Zugriff auf das Datengrid erhalten.

Informationen zu diesem Vorgang

Die Authentifizierung eines Clients reicht möglicherweise nicht aus, um den Zugriff auf das Datengrid zu schützen. Wenn Sie KeyStoreLoginAuthenticator verwenden, definieren Sie gewöhnlich nur wenige IDs, und alle IDs können vollständigen Zugriff auf das Datengrid haben. In diesem Fall ist eine Autorisierung möglicherweise nicht erforderlich. Bei der Verwendung der LDAP-Authentifizierung können jedoch viele IDs im LDAP-Server vorhanden sein, denen kein Zugriff auf die Griddaten oder Operationen erteilt werden darf.

Vorgehensweise

1. Aktivieren Sie die Zugriffssteuerung für das Datengrid. Geben Sie `securityEnabled="true"` in der Datei `ObjectGrid.xml` für das implementierte Datengrid an.

   Geben Sie diese Einstellung für jedes Grid an, das Sie definieren. Nach der Konfiguration dieser Einstellung werden Lese- und Schreiboperationen für Datengrideinträge nur noch für die IDs ausgeführt, denen in einer Richtliniendatei entsprechende Berechtigungen erteilt wurden.
2. Erstellen Sie eine Richtliniendatei. Sehen Sie sich die folgende Beispielrichtliniendatei an:

   ```
   grant codebase "http://www.ibm.com/com/ibm/ws/objectgrid/security/PrivilegedAction"
       principal javax.security.auth.x500.X500Principal "CN=cashier,O=acme,OU=OGSample" {
       permission com.ibm.websphere.objectgrid.security.MapPermission "accounting.*", "read ";
   };

   grant codebase "http://www.ibm.com/com/ibm/ws/objectgrid/security/PrivilegedAction"
       principal javax.security.auth.x500.X500Principal "CN=manager,O=acme,OU=OGSample" {
       permission com.ibm.websphere.objectgrid.security.MapPermission "accounting.*", "all";
   };
   ```

   Mithilfe von Richtliniendateien können je nach Autorisierung des Benutzers verschiedene Berechtigungen erteilt werden. Weitere Informationen zum Erstellen dieser Datei finden Sie unter Lernprogramm zur Java-SE-Sicherheit - Schritt 5.

3. Konfigurieren Sie jeden Container-Server so, dass diese Richtliniendatei geladen wird. Sie können diese Konfiguration durchführen, indem Sie den Container mit dem folgenden JVM-Argument starten:

   ```
   -Djava.security.policy=<Richtliniendatei>
   ```

   Tipp: Diese Richtliniendatei wird auch für die Steuerung des Verwaltungszugriffs auf die Daten-Grid-Server verwendet. Wenn Sie diese Richtliniendatei zum Steuern des Verwaltungszugriffs verwenden, muss die Richtliniendatei MBeanPermission-Einträge enthalten und von den Katalogservern und Container-Servern geladen werden.

Nächste Schritte

„Zugriff für Verwaltungsoperationen in eigenständigen Umgebungen autorisieren" auf Seite 219

Zugriff auf das Datengrid im Liberty-Profil autorisieren

Mit der Richtliniendatei können Sie können steuern, welche Benutzer spezielle Berechtigungen für den Zugriff auf das Datengrid im Liberty-Profil erhalten.

Informationen zu diesem Vorgang

Die Authentifizierung eines Clients reicht möglicherweise nicht aus, um den Zugriff auf das Datengrid zu schützen. Wenn Sie die Eigenschaft "KeyStoreLoginAuthenticator" verwenden, definieren Sie gewöhnlich nur wenige IDs, und alle IDs können vollständigen Zugriff auf das Grid haben. In diesem Fall ist eine Autorisierung möglicherweise nicht erforderlich. Bei der Verwendung der LDAP-Authentifizierung können jedoch viele IDs im LDAP-Server vorhanden sein, denen kein Zugriff auf die Griddaten oder Operationen erteilt werden sollte.

Vorgehensweise

1. Aktivieren Sie die Zugriffssteuerung für das Datengrid. Geben Sie `securityEnabled="true"` in der Datei `ObjectGrid.xml` für das implementierte Datengrid an.

   Geben Sie diese Einstellung für jedes Grid an, das Sie definieren. Nach der Konfiguration dieser Einstellung werden Lese- und Schreiboperationen für Datengrideinträge nur noch für die IDs ausgeführt, denen in einer Richtliniendatei entsprechende Berechtigungen erteilt wurden.
2. Erstellen Sie eine Richtliniendatei. Sehen Sie sich die folgende Beispielrichtliniendatei an:

   ```
   grant codebase "http://www.ibm.com/com/ibm/ws/objectgrid/security/PrivilegedAction"
       principal javax.security.auth.x500.X500Principal "CN=cashier,O=acme,OU=OGSample" {
       permission com.ibm.websphere.objectgrid.security.MapPermission "accounting.*", "read ";
   };

   grant codebase "http://www.ibm.com/com/ibm/ws/objectgrid/security/PrivilegedAction"
       principal javax.security.auth.x500.X500Principal "CN=manager,O=acme,OU=OGSample" {
       permission com.ibm.websphere.objectgrid.security.MapPermission "accounting.*", "all";
   };
   ```

   Mithilfe von Richtliniendateien können je nach Autorisierung des Benutzers verschiedene Berechtigungen erteilt werden. Weitere Informationen zum Erstellen dieser Datei finden Sie unter Lernprogramm zur Java-SE-Sicherheit - Schritt 5.
3. Konfigurieren Sie jeden Container-Server so, dass diese Richtliniendatei geladen wird. Sie können diese Konfiguration durchführen, indem Sie der Datei

jvm.options im Verzeichnis WLP-Installationsverzeichnis/usr/servers/<server_name> das folgende JVM-Argument hinzufügen:

```
-Djava.security.policy=<Richtliniendatei>
```

Tipp: Diese Richtliniendatei wird auch für die Steuerung des Verwaltungszugriffs auf die Daten-Grid-Server verwendet. Wenn Sie diese Richtliniendatei zum Steuern des Verwaltungszugriffs verwenden, muss die Richtliniendatei MBeanPermission-Einträge enthalten und von den Katalogservern und Container-Servern geladen werden.

Wenn Sie eine Datei jvm.options auf Serverkonfigurationsebene erstellen müssen, müssen Sie die Version in die Datei WLP-Installationsstammverzeichnis/etc/jvm.options kopieren.

Nächste Schritte

„Zugriff für Verwaltungsoperationen im Liberty-Profil autorisieren" auf Seite 220

Zugriff auf das Datengrid im OSGi-Framework autorisieren

Mit der Richtliniendatei können Sie können steuern, welche Benutzer spezielle Berechtigungen für den Zugriff auf das Datengrid im OSGi-Framework erhalten.

Vorbereitende Schritte

Sie müssen das OSGi-Framework installieren, bevor Sie das Datengrid sichern. Weitere Informationen finden Sie unter „Eclipse-Equinox-OSGi-Framework mit Eclipse Gemini für Clients und Server installieren" auf Seite 232.

Informationen zu diesem Vorgang

Die Authentifizierung eines Clients reicht möglicherweise nicht aus, um den Zugriff auf das Datengrid zu schützen. Wenn Sie die Eigenschaft "KeyStoreLoginAuthenticator" verwenden, definieren Sie gewöhnlich nur wenige IDs, und alle IDs können vollständigen Zugriff auf das Grid haben. In diesem Fall ist eine Autorisierung möglicherweise nicht erforderlich. Bei der Verwendung der LDAP-Authentifizierung können jedoch viele IDs im LDAP-Server vorhanden sein, denen kein Zugriff auf die Griddaten oder Operationen erteilt werden sollte.

Vorgehensweise

1. Aktivieren Sie die Zugriffssteuerung für das Datengrid. Geben Sie securityEnabled="true" in der Datei ObjectGrid.xml für das implementierte Datengrid an.

   Geben Sie diese Einstellung für jedes Grid an, das Sie definieren. Nach der Konfiguration dieser Einstellung werden Lese- und Schreiboperationen für Datengrideinträge nur noch für die IDs ausgeführt, denen in einer Richtliniendatei entsprechende Berechtigungen erteilt wurden.
2. Erstellen Sie eine Richtliniendatei. Fügen Sie der Sicherheitsrichtliniendatei die folgenden Codezeilen hinzu, um der Datei osgi.jar die Berechtigung "AllPermission" für das implementierte Datengrid zu erteilen.

   ```
   grant codeBase "file:/opt/OSGI2/plugins/org.eclipse.osgi_3.7.1.R37x_v20110808-1106.jar" {
     permission java.security.AllPermission;
   };
   ```

   Geben Sie diesen Code für jedes Grid an, das Sie definieren. Nach der Konfiguration dieser Einstellung werden Lese- und Schreiboperationen für Datengrideinträge nur noch für die IDs ausgeführt, denen in einer Richtliniendatei

die entsprechenden Berechtigungen erteilt wurden. Mithilfe von Richtliniendateien können je nach Autorisierung des Benutzers verschiedene Berechtigungen erteilt werden. Weitere Informationen zum Erstellen dieser Datei finden Sie unter Lernprogramm zur Java-SE-Sicherheit - Schritt 5.

Die Richtliniendatei gleicht dem folgenden Beispiel:

Hinweis: Die Richtliniendatei enthält gewöhnlich auch MapPermission-Einträge, die im Abschnitt Lernprogramm zur Java-SE-Sicherheit - Schritt 5 dokumentiert sind.

```
grant codeBase "file:${objectgrid.home}/lib/*" {
permission java.security.AllPermission;
};

grant principal javax.security.auth.x500.X500Principal "CN=manager,O=acme,OU=OGSample" {
permission javax.management.MBeanPermission "*", "getAttribute,setAttribute,invoke,queryNames";
};
```

3. Konfigurieren Sie jeden Container-Server so, dass diese Richtliniendatei geladen wird. Sie können diese Konfiguration durchführen, indem Sie den Container mit dem folgenden JVM-Argument starten:

   ```
   -Djava.security.policy=<Richtliniendatei>
   ```

   Tipp: Diese Richtliniendatei wird auch für die Steuerung des Verwaltungszugriffs auf die Daten-Grid-Server verwendet. Wenn Sie diese Richtliniendatei zum Steuern des Verwaltungszugriffs verwenden, muss die Richtliniendatei MBeanPermission-Einträge enthalten und von den Katalogservern und Container-Servern geladen werden.

Nächste Schritte

„Zugriff für Verwaltungsoperationen im OSGi-Framework autorisieren" auf Seite 221

Zugriff auf das Datengrid in WebSphere Application Server autorisieren

Welche Benutzer spezielle Berechtigungen für den Zugriff auf das Datengrid in Implementierungen von WebSphere Application Server haben, können Sie auf dieselbe Weise steuern wie den Zugriff auf das Datengrid in eigenständigen Implementierungen.

Informationen zu diesem Vorgang

Die Authentifizierung eines Clients reicht möglicherweise nicht aus, um den Zugriff auf das Datengrid zu schützen. Wenn Sie KeyStoreLoginAuthenticator verwenden, definieren Sie gewöhnlich nur wenige IDs, und alle IDs können vollständigen Zugriff auf das Datengrid haben. In diesem Fall ist eine Autorisierung möglicherweise nicht erforderlich. Bei der Verwendung der LDAP-Authentifizierung können jedoch viele IDs im LDAP-Server vorhanden sein, denen kein Zugriff auf die Griddaten oder Operationen erteilt werden darf.

Achtung: Es ist nicht erforderlich, MBeanPermissions für WebSphere-Application-Server-Implementierungen von eXtreme-Scale-Servern anzugeben, weil der JMX-Zugriff von WebSphere Application Server selbst gesteuert wird.

Vorgehensweise

1. Aktivieren Sie die Zugriffssteuerung für das Datengrid. Geben Sie `securityEnabled="true"` in der Datei `ObjectGrid.xml` für das implementierte Datengrid an.

Geben Sie diese Einstellung für jedes Grid an, das Sie definieren. Nach der Konfiguration dieser Einstellung werden Lese- und Schreiboperationen für Datengrideinträge nur noch für die IDs ausgeführt, denen in einer Richtliniendatei entsprechende Berechtigungen erteilt wurden.

2. Erstellen Sie eine Richtliniendatei. Mithilfe von Richtliniendateien können je nach Autorisierung des Benutzers verschiedene Berechtigungen erteilt werden. Weitere Informationen zum Erstellen dieser Datei finden Sie unter Lerneinheit 4.2: Benutzerbasierte Berechtigung aktivieren.
3. Konfigurieren Sie jeden Container-Server so, dass diese Richtliniendatei geladen wird. Sie können die Richtliniendatei in den generischen JVM-Argumenten des Anwendungsservers angeben, in dem der Container ausgeführt wird. Weitere Informationen zum Festlegen der Servereigenschaften mit JVM-Eigenschaften finden Sie unter Lerneinheit 2.2: Sicherheit des Katalogservers konfigurieren.

   ```
   -Djava.security.policy=<Richtliniendatei>
   ```

Nächste Schritte

„Zugriff für Verwaltungsoperationen in WebSphere Application Server autorisieren" auf Seite 222

Zugriff für spezielle Verwaltungsoperationen autorisieren

Es sind eine spezielle Autorisierung erforderlich, damit Benutzer Verwaltungsoperationen im Datengrid ausführen können.

Nächste Schritte

„Daten, die zwischen eXtreme-Scale-Clients und -Servern übertragen werden, mit SSL-Verschlüsselung sichern" auf Seite 222

Zugriff für Verwaltungsoperationen in eigenständigen Umgebungen autorisieren

Die meisten Datengridimplementierer beschränken den Verwaltungszugriff auf einen Teil der Benutzer, die auf das Datengrid zugreifen können.

Vorgehensweise

Sie müssen die Katalogserver und Container-Server mit dem Java-Sicherheitsmanager ausführen, der eine Richtliniendatei erfordert.
Die Richtliniendatei wird durch Übergabe des JVM-Arguments `-Djava.security.policy=<Richtliniendatei>` angegeben.
Der Java-Sicherheitsmanager wird gestartet, indem das JVM-Argument "-Djava.security.manager" beim Starten des eXtreme-Scale-Servers angegeben wird. Geben Sie dieses Argument für Container- und Katalogserver an.
Die Richtliniendatei gleicht dem folgenden Beispiel:

Hinweis: Die Richtliniendatei enthält gewöhnlich auch MapPermission-Einträge, die im Abschnitt Lernprogramm zur Java-SE-Sicherheit - Schritt 5 dokumentiert sind.

```
grant codeBase "file:${objectgrid.home}/lib/*" {
permission java.security.AllPermission;
};

grant principal javax.security.auth.x500.X500Principal "CN=manager,O=acme,OU=OGSample" {
permission javax.management.MBeanPermission "*", "getAttribute,setAttribute,invoke,queryNames";
};
```

In diesem Beispiel wird nur der Manager-Principal für Verwaltungsoperationen mit dem Befehl **xscmd** autorisiert. Sie können bei Bedarf weitere Zeilen hinzufügen, um weiteren Principals MBean-Berechtigungen zu erteilen.

Geben Sie den folgenden Befehl ein: UNIX Linux

```
startOgServer.sh <arguments> -jvmargs -Djava.security.auth.login.config=jaas.config
-Djava.security.manager -Djava.security.policy="auth.policy" -Dobjectgrid.home=$OBJECTGRID_HOME
```

UNIX Linux 8.6+

```
startXsServer.sh <Argumente> -jvmargs -Djava.security.auth.login.config=jaas.config
-Djava.security.manager -Djava.security.policy="auth.policy" -Dobjectgrid.home=$OBJECTGRID_HOME
```

Windows

```
startOgServer.bat <arguments> -jvmargs -Djava.security.auth.login.config=jaas.config
-Djava.security.manager -Djava.security.policy="auth.policy" -Dobjectgrid.home=%OBJECTGRID_HOME%
```

Windows 8.6+

```
startXsServer.bat <Argumente> -jvmargs -Djava.security.auth.login.config=jaas.config
 -Djava.security.manager -Djava.security.policy="auth.policy" -Dobjectgrid.home=%OBJECTGRID_HOME%
```

Nächste Schritte

„Daten, die zwischen eXtreme-Scale-Servern in eigenständigen Umgebungen übertragen werden, mit SSL-Verschlüsselung sichern” auf Seite 222

Zugriff für Verwaltungsoperationen im Liberty-Profil autorisieren

Über die Verwaltungssicherheit können Sie Benutzer für den Zugriff auf das Datengrid im Liberty-Profil berechtigen.

Informationen zu diesem Vorgang

Die meisten Datengridimplementierer beschränken den Verwaltungszugriff auf einen Teil der Benutzer, die auf das Datengrid zugreifen können.

Vorgehensweise

- Führen Sie den Java-Sicherheitsmanager aus, und geben Sie eine Richtliniendatei an, in der die Berechtigung "MBeanPermissions" erteilt wird, um den Verwaltungszugriff einzuschränken, wenn eXtreme-Scale-Server im Liberty-Profil ausgeführt werden. Dieser Ansatz ist derselbe wie in eigenständigen Umgebungen. Geben Sie die folgenden Zeilen in der Datei `jvm.options` für jeden Liberty-Profilserver ein, in dem ein Katalog- oder Container-Server von eXtreme Scale ausgeführt wird.

  ```
  -Djava.security.manager
  -Djava.security.policy="policy file"
  ```

- Konfigurieren Sie die Richtliniendatei so, dass dem Code von Liberty-Profil und dem Code von eXtreme Scale alle Berechtigungen erteilt werden. Diese Konfiguration ermöglicht dem Liberty-Profil und eXtreme Scale die Arbeit mit dem Sicherheitsmanager. Fügen Sie der Datei `jvm.options` auf Serverebene die folgenden Zeilen hinzu:

  ```
  grant codeBase "file:${objectgrid.home}/lib/*" {
  permission java.security.AllPermission;
  };
  ```

Nächste Schritte

„Daten, die zwischen eXtreme Scale und dem Liberty-Profil übertragen werden, mit SSL-Verschlüsselung sichern” auf Seite 224

Zugriff für Verwaltungsoperationen im OSGi-Framework autorisieren

Über die Verwaltungssicherheit können Sie Benutzer für den Zugriff auf das Datengrid im OSGi-Framework berechtigen.

Vorbereitende Schritte

Sie müssen das OSGi-Framework installieren, bevor Sie das Datengrid sichern. Weitere Informationen finden Sie unter „Eclipse-Equinox-OSGi-Framework mit Eclipse Gemini für Clients und Server installieren” auf Seite 232.

Informationen zu diesem Vorgang

Die meisten Datengridimplementierer beschränken den Verwaltungszugriff auf einen Teil der Benutzer, die auf das Datengrid zugreifen können.

Vorgehensweise

- Sie müssen die Katalogserver und Container-Server mit dem Java-Sicherheitsmanager ausführen, der eine Richtliniendatei erfordert.

  Die Richtliniendatei wird durch Übergabe des JVM-Arguments `-Djava.security.policy=<Richtliniendatei>` angegeben.

  Der Java-Sicherheitsmanager wird gestartet, indem das JVM-Argument "-Djava.security.manager" beim Starten des eXtreme-Scale-Servers angegeben wird. Geben Sie dieses Argument für Container- und Katalogserver an.

  Die Richtliniendatei gleicht dem folgenden Beispiel:

  Hinweis: Die Richtliniendatei enthält gewöhnlich auch MapPermission-Einträge, die im Abschnitt Lernprogramm zur Java-SE-Sicherheit - Schritt 5 dokumentiert sind.

```
grant codeBase "file:${objectgrid.home}/lib/*" {
permission java.security.AllPermission;
};

grant principal javax.security.auth.x500.X500Principal "CN=manager,O=acme,OU=OGSample" {
permission javax.management.MBeanPermission "*", "getAttribute,setAttribute,invoke,queryNames";
};
```

  In diesem Beispiel wird nur der Manager-Principal für Verwaltungsoperationen mit dem Befehl **xscmd** autorisiert. Sie können bei Bedarf weitere Zeilen hinzufügen, um weiteren Principals MBean-Berechtigungen zu erteilen.

- Starten Sie die Katalog- und Server-Container, indem Sie die vorherigen JVM-Argumente in der Befehlszeile angeben. Beispiel:

  ```
  /opt/XS86/java/jre/bin/java -DclusterSecurityFile=/og/security/secFiles_SA/objectGridSecurity.x
  ```

Nächste Schritte

„Daten, die zwischen eXtreme Scale und dem OSGi-Framework übertragen werden, mit SSL-Verschlüsselung sichern” auf Seite 225

Zugriff für Verwaltungsoperationen in WebSphere Application Server autorisieren

Über die Verwaltungssicherheit können nur Administratoren von WebSphere Application Server eXtreme-Scale-Verwaltungsoperationen ausführen.

Informationen zu diesem Vorgang

Die Autorisierung für den Verwaltungszugriff funktioniert in Implementierungen von WebSphere Application Server anders als in eigenständigen Umgebungen. eXtreme-Scale-Verwaltungsoperationen können nur von Benutzern von WebSphere Application Server ausgeführt werden, die WebSphere-Application-Server-Administratoren sind. Sie müssen in der Richtliniendatei keine MbeanPermissions angeben.

Vorgehensweise

Aktivieren Sie die Verwaltungssicherheit in WebSphere Application Server. Klicken Sie in der Administrationskonsole auf **Sicherheit** > **Globale Sicherheit**. Klicken Sie auf **Verwaltungssicherheit aktivieren**, und wählen Sie **Java-2-Sicherheit** aus, um den Anwendungszugriff auf lokale Ressourcen zu beschränken.

Nächste Schritte

„Daten, die zwischen eXtreme Scale und WebSphere Application Server übertragen werden, mit SSL-Verschlüsselung sichern" auf Seite 227

Daten, die zwischen eXtreme-Scale-Clients und -Servern übertragen werden, mit SSL-Verschlüsselung sichern

Sie können die Kommunikation zwischen eXtreme-Scale-Clients und -Servern mit SSL-Verschlüsselung schützen.

Nächste Schritte

„Sicherheitsartefakte für autorisierte Benutzer speichern" auf Seite 228

Daten, die zwischen eXtreme-Scale-Servern in eigenständigen Umgebungen übertragen werden, mit SSL-Verschlüsselung sichern

Konfigurieren Sie SSL-Eigenschaften und JMX-Ports, um sensible Daten zu schützen, die zwischen Servern über das Netz übertragen werden.

Informationen zu diesem Vorgang

Wenn ein Datengrid implementiert wird, werden die in diesem Grid enthaltenen sensiblen Daten über das Netz übertragen. Auch die Berechtigungsnachweise, die Datengrids für die Authentifizierung beim Datengrid verwenden, werden über das Netz übertragen. Verwenden Sie zum Schutz der Daten und der Berechtigungsnachweise während der Übertragung die Verschlüsselung auf Transportebene mit SSL, um Implementieren zu sichern.

Die Sicherheit von SSL ist vom Zugriffsschutz der Keystores und Truststores abhängig, sodass nur berechtigte Benutzer Zugriff auf die Keystores und Truststores haben. Nachdem Sie die SSL-Verschlüsselung aktiviert haben, müssen Sie einen JMXConnectorPort- und einen JMXServicePort-Wert in der Servereigenschaftendatei angeben, damit der SSL-Zugriffsschutz für JMX-Datenverkehr verwendet wird.

Der Transport zwischen dem JMX-Client und dem JMX-Server kann mit TLS (Transport Layer Security) oder SSL gesichert werden. Wenn das Attribut "transportType" des Katalogserver oder Container-Servers auf "SSL_Required" oder "SSL_Supported" gesetzt ist, müssen Sie SSL verwenden, um die Verbindung zum JMX-Server herzustellen.

Vorgehensweise

1. Geben Sie SSL in der Servereigenschaftendatei an. Setzen Sie die Eigenschaft "transportType" auf `SSL-Required`, z. B.:

   ```
   transportType=SSL-Required
   ```

2. Geben Sie SSL-Eigenschaften in der Servereigenschaftendatei an.

   ```
   alias=serverprivate
   contextProvider=IBMJSSE2
   protocol=SSL
   keyStoreType=JKS
   keyStore=etc/test/security/key.jks
   keyStorePassword=serverpw
   trustStoreType=JKS
   trustStore=etc/test/security/trust.jks
   trustStorePassword=public
   clientAuthentication=false
   ```

   Konfigurieren Sie den Truststore, den Truststore-Typ und das Truststore-Kennwort. Es ist nicht erforderlich, einen Keystore, einen Keystore-Typ und ein Keystore-Kennwort für den Client anzugeben. Der Alias, der Keystore, das Keystore-Kennwort und der Keystore-Typ sind im Client nur dann erforderlich, wenn die SSL-Eigenschaften des Servers `clientAuthentication=true` enthalten. Dieser Wert wird jedoch nur selten verwendet.

   Der Client-Truststore muss das Serverzertifikat anerkennen. Wenn das Serverzertifikat (wie im Lernprogramm) selbst signiert ist, muss das Zertifikat in den Truststore des Clients importiert werden. Wenn das Serverzertifikat von einer lokalen Zertifizierungsstelle ausgegeben wird, muss das Unterzeichnerzertifikat für diese Zertifizierungsstelle in den Client-Truststore importiert werden. Weitere Informationen zum Erstellen von Keystore- und Truststore-Dateien finden Sie unter Lernprogramm zur Java-SE-Sicherheit - Schritt 6.

3. Geben Sie SSL in der Clienteigenschaftendatei an, wenn SSL erforderlich ist. Setzen Sie die Eigenschaft "transportType" auf `SSL-Required` oder `SSL-Supported`, z. B.:

   ```
   transportType=SSL-Required
   ```

4. Geben Sie SSL-Eigenschaften in der Clienteigenschaftendatei an. Sie können beispielsweise die folgenden Eigenschaften angeben:

   ```
   alias=clientprivate
   contextProvider=IBMJSSE2
   protocol=SSL
   keyStoreType=JKS
   keyStore=etc/test/security/client.private
   keyStorePassword={xor}PDM2OjErLyg\=
   trustStoreType=JKS
   trustStore=etc/test/security/server.public
   trustStorePassword={xor}Lyo9MzY8
   ```

5. Setzen Sie den JMX-Service-Port. Verwenden Sie die Option **-JMXServicePort** im Script **startOgServer** oder **startXsServer**.

   Der Standardwert für den JMX-Service-Port in Katalogservern ist `1099`. Sie müssen für jede JVM in Ihrer Konfiguration eine andere Portnummer verwenden.

Wenn Sie JMX/RMI verwenden, geben Sie die Option **-JMXServicePort** und die Portnummer explizit an, selbst wenn Sie den Standardportwert verwenden möchten.

6. Setzen Sie den JMX-Connector-Port.

   Verwenden Sie die Option **-JMXConnectorPort** im Script **startOgServer** oder **startXsServer**.

   Die Definition des JMX-Service-Ports ist erforderlich, wenn Sie Informationen zum Container-Server aus dem Katalogserver anzeigen möchten. Der Port ist beispielsweise erforderlich, wenn Sie den Befehl **xscmd -c showMapSizes** verwenden. Setzen Sie den JMX-Connector-Port, um die Erstellung ephemerer Ports zu verhindern.

Nächste Schritte

„Sicherheitsartefakte in eigenständigen Umgebungen speichern" auf Seite 228

Daten, die zwischen eXtreme Scale und dem Liberty-Profil übertragen werden, mit SSL-Verschlüsselung sichern

Konfigurieren Sie SSL-Eigenschaften und JMX-Ports, um sensible Daten zu schützen, die zwischen WebSphere eXtreme Scale und dem Liberty-Profil übertragen werden.

Informationen zu diesem Vorgang

Wenn ein Datengrid implementiert wird, werden die in diesem Grid enthaltenen sensiblen Daten über das Netz übertragen. Auch die Berechtigungsnachweise, die Datengrids für die Authentifizierung beim Datengrid verwenden, werden über das Netz übertragen. Verwenden Sie zum Schutz der Daten und der Berechtigungsnachweise während der Übertragung die Verschlüsselung auf Transportebene mit SSL, um Implementieren zu sichern.

Die Sicherheit von SSL ist vom Zugriffsschutz der Keystores und Truststores abhängig, sodass nur berechtigte Benutzer Zugriff auf die Keystores und Truststores haben. Nachdem Sie die SSL-Verschlüsselung aktiviert haben, müssen Sie einen JMXConnectorPort- und einen JMXServicePort-Wert in der Servereigenschaftendatei angeben, damit der SSL-Zugriffsschutz für JMX-Datenverkehr verwendet wird.

Der Transport zwischen dem JMX-Client und dem JMX-Server kann mit TLS (Transport Layer Security) oder SSL gesichert werden. Wenn das Attribut "transportType" des Katalogserver oder Container-Servers auf "SSL_Required" oder "SSL-_Supported" gesetzt ist, müssen Sie SSL verwenden, um die Verbindung zum JMX-Server herzustellen.

Vorgehensweise

1. Geben Sie SSL in der Servereigenschaftendatei an. Setzen Sie die Eigenschaft "transportType" auf `SSL-Required`, z. B.:

   ```
   transportType=SSL-Required
   ```

2. Geben Sie SSL-Eigenschaften in der Servereigenschaftendatei an.

   ```
   alias=serverprivate
   contextProvider=IBMJSSE2
   protocol=SSL
   keyStoreType=JKS
   keyStore=etc/test/security/key.jks
   keyStorePassword=serverpw
   ```

```
trustStoreType=JKS
trustStore=etc/test/security/trust.jks
trustStorePassword=public
clientAuthentication=false
```

Konfigurieren Sie den Truststore, den Truststore-Typ und das Truststore-Kennwort. Es ist nicht erforderlich, einen Keystore, einen Keystore-Typ und ein Keystore-Kennwort für den Client anzugeben. Der Alias, der Keystore, das Keystore-Kennwort und der Keystore-Typ sind im Client nur dann erforderlich, wenn die SSL-Eigenschaften des Servers `clientAuthentication=true` enthalten. Dieser Wert wird jedoch nur selten verwendet.

Der Client-Truststore muss das Serverzertifikat anerkennen. Wenn das Serverzertifikat (wie im Lernprogramm) selbst signiert ist, muss das Zertifikat in den Truststore des Clients importiert werden. Wenn das Serverzertifikat von einer lokalen Zertifizierungsstelle ausgegeben wird, muss das Unterzeichnerzertifikat für diese Zertifizierungsstelle in den Client-Truststore importiert werden. Weitere Informationen zum Erstellen von Keystore- und Truststore-Dateien finden Sie unter Lernprogramm zur Java-SE-Sicherheit - Schritt 6.

3. Geben Sie SSL in der Clienteigenschaftendatei an, wenn SSL erforderlich ist. Setzen Sie die Eigenschaft "transportType" auf `SSL-Required` oder `SSL-Supported`, z. B.:

   ```
   transportType=SSL-Required
   ```

4. Geben Sie SSL-Eigenschaften in der Clienteigenschaftendatei an. Sie können beispielsweise die folgenden Eigenschaften angeben:

   ```
   alias=clientprivate
   contextProvider=IBMJSSE2
   protocol=SSL
   keyStoreType=JKS
   keyStore=etc/test/security/client.private
   keyStorePassword={xor}PDM2OjErLyg\=
   trustStoreType=JKS
   trustStore=etc/test/security/server.public
   trustStorePassword={xor}Lyo9MzY8
   ```

5. Setzen Sie den JMX-Service-Port in der Servereigenschaftendatei.

   Der Standardwert für den JMX-Service-Port in Katalogservern ist `1099`. Sie müssen für jede JVM in Ihrer Konfiguration eine andere Portnummer verwenden. Wenn Sie JMX/RMI verwenden möchten, geben Sie die Option **JMXServicePort** und die Portnummer explizit an, selbst wenn Sie den Standardportwert verwenden möchten.

6. Setzen Sie den JMX-Connector-Port in der Servereigenschaftendatei.

   Die Definition des JMX-Service-Ports ist erforderlich, wenn Sie Informationen zum Container-Server aus dem Katalogserver anzeigen möchten. Der Port ist beispielsweise erforderlich, wenn Sie den Befehl **xscmd -c showMapSizes** verwenden. Setzen Sie den JMX-Connector-Port, um die Erstellung ephemerer Ports zu verhindern.

Nächste Schritte

„Sicherheitsartefakte im Liberty-Profil speichern” auf Seite 228

Daten, die zwischen eXtreme Scale und dem OSGi-Framework übertragen werden, mit SSL-Verschlüsselung sichern

Konfigurieren Sie SSL-Eigenschaften und JMX-Ports, um sensible Daten zu schützen, die zwischen WebSphere eXtreme Scale und dem OSGi-Framework übertragen werden.

Vorbereitende Schritte

Sie müssen das OSGi-Framework installieren, bevor Sie das Datengrid sichern. Weitere Informationen finden Sie unter „Eclipse-Equinox-OSGi-Framework mit Eclipse Gemini für Clients und Server installieren” auf Seite 232.

Informationen zu diesem Vorgang

Wenn ein Datengrid implementiert wird, werden die in diesem Grid enthaltenen sensiblen Daten über das Netz übertragen. Auch die Berechtigungsnachweise, die Datengrids für die Authentifizierung beim Datengrid verwenden, werden über das Netz übertragen. Verwenden Sie zum Schutz der Daten und der Berechtigungsnachweise während der Übertragung die Verschlüsselung auf Transportebene mit SSL, um Implementieren zu sichern.

Die Sicherheit von SSL ist vom Zugriffsschutz der Keystores und Truststores abhängig, sodass nur berechtigte Benutzer Zugriff auf die Keystores und Truststores haben. Nachdem Sie die SSL-Verschlüsselung aktiviert haben, müssen Sie einen JMXConnectorPort- und einen JMXServicePort-Wert in der Servereigenschaftendatei angeben, damit der SSL-Zugriffsschutz für JMX-Datenverkehr verwendet wird.

Der Transport zwischen dem JMX-Client und dem JMX-Server kann mit TLS (Transport Layer Security) oder SSL gesichert werden. Wenn das Attribut "transportType" des Katalogserver oder Container-Servers auf "SSL_Required" oder "SSL-_Supported" gesetzt ist, müssen Sie SSL verwenden, um die Verbindung zum JMX-Server herzustellen.

Vorgehensweise

1. Geben Sie SSL in der Servereigenschaftendatei an. Setzen Sie die Eigenschaft "transportType" auf `SSL-Required`, z. B.:
   ```
   transportType=SSL-Required
   ```
2. Zur Verwendung von SSL müssen Sie den Truststore, den Truststore-Typ und das Truststore-Kennwort im MBean-Client mit Systemeigenschaften konfigurieren, die mit "-D" beginnen, z. B.:
   ```
   -Djavax.net.ssl.trustStore=TRUST_STORE_LOCATION-Djavax.net.ssl.trustStorePassword=TRUST_STORE_PAS
   ```

   Wenn Sie "com.ibm.websphere.ssl.protocol.SSLSocketFactory" als SSL-SocketFactory in der Datei *Java-Ausgangsverzeichnis*`/jre/lib/security/java.security` verwenden, definieren Sie die folgenden Eigenschaften:
   ```
   -Dcom.ibm.ssl.trustStore=TRUST_STORE_LOCATION-Dcom.ibm.ssl.trustStorePassword=TRUST_STORE_PASSWOR
   ```
3. Setzen Sie den JMX-Service-Port in der Servereigenschaftendatei.

   Der Standardwert für den JMX-Service-Port in Katalogservern ist `1099`. Sie müssen für jede JVM in Ihrer Konfiguration eine andere Portnummer verwenden. Wenn Sie JMX/RMI verwenden, geben Sie die Option **`JMXServicePort`** und die Portnummer explizit an, selbst wenn Sie den Standardportwert verwenden möchten.
4. Setzen Sie den JMX-Connector-Port in der Servereigenschaftendatei.

   Die Definition des JMX-Service-Ports ist erforderlich, wenn Sie Informationen zum Container-Server aus dem Katalogserver anzeigen möchten. Der Port ist beispielsweise erforderlich, wenn Sie den Befehl **`xscmd c showMapSizes`** verwenden. Setzen Sie den JMX-Connector-Port, um die Erstellung ephemerer Ports zu verhindern.
5. Geben Sie den SSL-Port in der Befehlszeile des OSGi-Frameworks mit dem folgenden JVM-Argument an:

```
-Dcom.ibm.CSI.SSL.Port=7602
```

Nächste Schritte

„Sicherheitsartefakte im OSGi-Framework speichern" auf Seite 229

Daten, die zwischen eXtreme Scale und WebSphere Application Server übertragen werden, mit SSL-Verschlüsselung sichern

WebSphere eXtreme Scale verwendet die SSL-Konfiguration (Secure Sockets Layer) in WebSphere Application Server.

Informationen zu diesem Vorgang

Um sicherzustellen, dass der SSL-Zugriffsschutz für den gesamten Datengridverkehr im Netz verwendet wird, konfigurieren Sie die globale Sicherheit, die Sicherheit für eingehenden und abgehenden CSIv2-Datenverkehr in der Administrationskonsole von WebSphere Application Server und die Verwaltung von SSL-Zertifikaten und -Schlüsseln.

Vorgehensweise

1. Konfigurieren Sie die globale Sicherheit in WebSphere Application Server. Weitere Informationen zum Konfigurieren der globalen Sicherheit finden Sie unter Globale Sicherheitseinstellungen konfigurieren.
2. Konfigurieren Sie die Sicherheit für eingehenden CSIv2-Datenverkehr. Klicken Sie in der Administrationskonsole von WebSphere Application Server auf **Sicherheit** > **Globale Sicherheit** > **RMI/IIOP-Sicherheit** > **Eingehende CSIv2-Kommunikation**. Klicken Sie auf **SSL erforderlich**.
3. Konfigurieren Sie die Sicherheit für den abgehenden CSIv2-Datenverkehr. Klicken Sie in der Administrationskonsole von WebSphere Application Server auf **Sicherheit** > **Globale Sicherheit** > **RMI/IIOP-Sicherheit** > **Abgehende CSIv2-Kommunikation**. Wählen Sie **SSL unterstützt** oder **SSL erforderlich** für die abgehende CSIv2-Kommunikation aus.
4. Konfigurieren Sie die Verwaltung von SSL-Zertifikaten und -Schlüsseln in WebSphere Application Server. Wenn in einer Instanz von WebSphere Application Server nur ein eXtreme-Scale-Client ausgeführt wird und die eXtreme-Scale-Daten-Grid-Server eigenständig sind, müssen Sie sicherstellen, dass die Informationen zu den Keystore- und Truststore-Zertifikaten in den Keystore- und Truststore-Dateien enthalten sind, die in der Servereigenschaftendatei angegeben sind, die zum Starten der eigenständigen Katalog- und Container-Server verwendet wird.

   Wenn der Client, die Katalogserver und die Container-Server alle in Prozessen von WebSphere Application Server ausgeführt werden, verwenden Sie die Sicherheitskonfiguration von WebSphere Application Server für die Client-Server-Kommunikation.

   Sind jedoch mehrere Katalogserver konfiguriert, die in einem Prozess von WebSphere Application Server ausgeführt werden, werden für die Kommunikation zwischen den Katalogen eigene proprietäre Transportpfade verwendet, die nicht mit den CSIv2-Transporteinstellungen (Common Secure Interoperability Protocol Version 2) von WebSphere Application Server verwaltet werden können. Deshalb müssen Sie die SSL-Eigenschaften in der Servereigenschaftendatei für jeden Katalogserver konfigurieren. Weitere Informationen finden Sie unter Lerneinheit 3.2: SSL-Eigenschaften der Eigenschaftendatei des Katalogservers hinzufügen.

Nächste Schritte

„Sicherheitsartefakte in WebSphere Application Server speichern” auf Seite 229

Sicherheitsartefakte für autorisierte Benutzer speichern

Keystores, Kennwörter, Shared Secrets und Eigenschaftendateien müssen in einem Verzeichnis gespeichert werden, auf das nur autorisierte Benutzer zugreifen können.

Nächste Schritte

Sichere Server starten und stoppen

Sicherheitsartefakte in eigenständigen Umgebungen speichern

Sie können sichere Kennwörter schützen, um den Zugriff nicht berechtigter Benutzer zu verhindern.

Informationen zu diesem Vorgang

Mit dem in WebSphere eXtreme Scale Client enthaltenen Dienstprogramm "FilePasswordEncoder" können Kennwörter in eXtreme-Scale-Konfigurationsdateien codiert werden. Das Dienstprogramm "FilePasswordEncoder" codiert Kennwörter. Es ist jedoch möglich, die Kennwörter wiederherzustellen, die für den Zugriff auf die Datei verwendet werden. Deshalb müssen Sie das Dateisystem, in dem die Clienteigenschaften, die Servereigenschaften sowie die Keystores und Truststores aufbewahrt werden, zu schützen, sodass nur autorisierte Benutzer Zugriff haben.

Vorgehensweise

Führen Sie den Befehl Befehl **`FilePasswordEncoder.bat|sh`** aus, um diese Eigenschaft mit einem xor-Algorithmus (`exclusive or`, exklusives Oder) zu codieren, um den Zugriffsschutz für Kennwörter zu gewährleisten.
Führen Sie das Dienstprogramm "FilePasswordEncoder" für die Datei `client.properties` und die Datei `server.properties` aus, z. B.:

```
./FilePasswordEncoder.sh <Servereigenschaftendatei>
./FilePasswordEncoder.sh <Clienteigenschaftendatei>
```

Ein erfahrener Benutzer kann codierte Kennwörter wiederherstellen. Diese Kennwörter sind nicht verschlüsselt, weil der eXtreme-Scale-Code in der Lage sein muss, die Kennwörter wiederherzustellen, um ausgeführt werden zu können. Stellen Sie deshalb sicher, dass nur autorisierte Personen auf die Dateien zugreifen können, in denen diese Kennwörter gespeichert sind.

Nächste Schritte

Sichere Server in einer eigenständigen Umgebung starten

Sicherheitsartefakte im Liberty-Profil speichern

Sie können sichere Kennwörter schützen, um den Zugriff nicht berechtigter eXtreme-Scale-Benutzer im Liberty-Profil zu verhindern.

Informationen zu diesem Vorgang

Mit dem in WebSphere eXtreme Scale Client enthaltenen Dienstprogramm "FilePasswordEncoder" können Kennwörter in eXtreme-Scale-Konfigurationsdateien codiert werden.

Vorgehensweise

1. Führen Sie den Befehl **securityUtility.bat|sh** des Liberty-Profils aus, um diese Eigenschaft mit einem xor-Algorithmus (exclusive or, exklusives Oder) zu codieren, um den Zugriffsschutz für Kennwörter zu gewährleisten. Beachten Sie, dass ein erfahrener Benutzer codierte Kennwörter wiederherstellen kann. Diese Kennwörter sind nicht verschlüsselt, weil der eXtreme-Scale-Code in der Lage sein muss, die Kennwörter wiederherzustellen, um ausgeführt werden zu können. Stellen Sie deshalb sicher, dass nur autorisierte Personen auf die Dateien zugreifen können, in denen diese Kennwörter gespeichert sind.
2. Beschränken Sie den Zugriff auf Keystore-Dateien und Truststore-Dateien, indem Sie den Zugriff auf das Dateisystem schützen, in dem die Dateien gespeichert sind.

Nächste Schritte

Sichere Server im Liberty-Profil starten und stoppen

Sicherheitsartefakte im OSGi-Framework speichern

Sie können sichere Kennwörter schützen, um den Zugriff nicht berechtigter Benutzer im OSGi-Framework zu verhindern.

Vorbereitende Schritte

Sie müssen das OSGi-Framework installieren, bevor Sie das Datengrid sichern. Weitere Informationen finden Sie unter „Eclipse-Equinox-OSGi-Framework mit Eclipse Gemini für Clients und Server installieren” auf Seite 232.

Informationen zu diesem Vorgang

Mit dem in WebSphere eXtreme Scale Client enthaltenen Dienstprogramm "FilePasswordEncoder" können Kennwörter in eXtreme-Scale-Konfigurationsdateien codiert werden.

Vorgehensweise

1. Führen Sie den Befehl Befehl **FilePasswordEncoder.bat|sh** aus, um diese Eigenschaft mit einem xor-Algorithmus (exclusive or, exklusives Oder) zu codieren, um den Zugriffsschutz für Kennwörter zu gewährleisten. Beachten Sie, dass ein erfahrener Benutzer codierte Kennwörter wiederherstellen kann. Diese Kennwörter sind nicht verschlüsselt, weil der eXtreme-Scale-Code in der Lage sein muss, die Kennwörter wiederherzustellen, um ausgeführt werden zu können. Stellen Sie deshalb sicher, dass nur autorisierte Personen auf die Dateien zugreifen können, in denen diese Kennwörter gespeichert sind.
2. Beschränken Sie den Zugriff auf Keystore-Dateien und Truststore-Dateien, indem Sie den Zugriff auf das Dateisystem schützen, in dem die Dateien gespeichert sind.

Nächste Schritte

Sichere Server im OSGi-Framework starten und stoppen

Sicherheitsartefakte in WebSphere Application Server speichern

Sie können sichere Kennwörter schützen, um den Zugriff nicht berechtigter Benutzer in Implementierungen von WebSphere Application Server zu verhindern.

Informationen zu diesem Vorgang

Kennwörter und das authenticationSecret in den Servereigenschaften- und Clienteigenschaftendateien müssen codiert werden.

Vorgehensweise

Rufen Sie das Dienstprogramm "PropFilePasswordEncoder" auf, um Kennwörter und das authenticationSecret zu codieren. Führen Sie den Befehl *WAS-Stammverzeichnis*/bin/PropFilePasswordEncoder.sh bzw. unter Windows den Befehl *WAS-Stammverzeichnis*\bin\PropFilePasswordEncoder.bat aus, z. B.:

```
./PropFilePasswordEncoder <Eigenschaftendatei> <zu_codierende_Eigenschaft>
```

Zu den Eigenschaften, die codiert werden müssen, gehören die Eigenschaften **keyStorePassword**, **trustStorePassword**, **credentialGeneratorProps** und **authenticationSecret**. Selbst wenn diese Eigenschaften codiert sind, können die ursprünglichen Werte wiederhergestellt werden. Das Dateisystem, in dem die Eigenschaftendateien, Keystores und Truststores aufbewahrt werden, muss geschützt werden, sodass nur autorisierte Benutzer auf sie zugreifen kann.
Weitere Informationen finden Sie in der Dokumentation zu WebSphere Application Server.

Nächste Schritte

Sichere Server in WebSphere Application Server starten

Szenario: OSGi-Umgebung für die Entwicklung und Ausführung von eXtreme-Scale-Plug-ins verwenden

Verwenden Sie diese Szenarien, um allgemeine Aufgaben in einer OSGi-Umgebung auszuführen. Das OSGi-Framework eignet sich beispielsweise ideal für das Starten von Servern und Clients in einem OSGi-Container, was Ihnen ermöglicht, Plug-ins von WebSphere eXtreme Scale dynamisch in der Laufzeitumgebung hinzuzufügen und zu aktualisieren.

Vorbereitende Schritte

Weitere Informationen zur OSGi-Unterstützung und deren Vorteilen finden Sie im Abschnitt „Übersicht über das OSGi-Framework” auf Seite 36.

Informationen zu diesem Vorgang

In den folgenden Szenarien geht es um das Erstellen und Ausführen dynamischer Plug-ins, was Ihnen ermöglicht, Plug-ins dynamisch zu installieren, zu starten, zu stoppen, zu ändern und zu deinstallieren. Sie können auch ein anderes wahrscheinliches Szenario verwenden, in dem Sie das OSGi-Framework ohne die dynamische Funktionalität verwenden können. Sie können Ihre Anwendungen trotzdem als Bundles packen, die über Services definiert und übertragen werden. Diese servicebasierten Bundles bieten mehrere Vorteile, zu denen effizientere Entwicklungs- und Implementierungsfunktionen gehören.

Szenarioziele

Nach dem Ausführen dieses Szenarios wissen Sie, wie die folgenden Ziele erreicht werden:

- Dynamische Plug-ins von eXtreme Scale für die Verwendung in einer OSGi-Umgebung erstellen
- eXtreme-Scale-Container in einer OSGi-Umgebung ohne dynamische Funktionalität ausführen

Übersicht über das OSGi-Framework

OSGi definiert ein dynamisches Modulsystem für Java. Die OSGi-Serviceplattform hat eine Schichtenarchitektur und ist für die Ausführung in verschiedenen Java-Standardprofilen bestimmt. Sie können Server und Client von WebSphere eXtreme Scale in einem OSGi-Container starten.

Vorteile der Ausführung von Anwendungen im OSGi-Container

Die OSGi-Unterstützung von WebSphere eXtreme Scale ermöglicht Ihnen, das Produkt im Eclipse-Equinox-OSGi-Framework zu implementieren. Zum Aktualisieren der von eXtreme Scale verwendeten Plug-ins mussten Sie früher die Java Virtual Machine (JVM) erneut starten, um die neuen Versionen der Plug-ins anzuwenden. Mit der Funktionalität für dynamische Aktualisierungen, die das OSGi-Framework bereitstellt, können Sie die Plug-in-Klassen jetzt ohne Neustart der JVM aktualisieren. Diese Plug-ins werden über Benutzerbundles als Services exportiert. WebSphere eXtreme Scale greift über eine Suche in der OSGi-Registry auf die Services zu.

eXtreme-Scale-Container lassen Sich durch Konfiguration mit dem OSGi-Konfigurationsverwaltungssservice oder mit OSGi Blueprint einfacher und dynamischer starten. Wenn Sie ein neues Datengrid mit der zugehörigen Verteilungsstrategie implementieren möchten, können Sie eine OSGi-Konfiguration erstellen oder ein Bundle mit den eXtreme-Scale-XML-Deskriptordateien implementieren. Mit der OSGi-Unterstützung können Anwendungsbundles, die die Konfigurationsdaten von eXtreme Scale enthalten, installiert, gestartet, gestoppt, aktualisiert und deinstalliert werden, ohne das gesamte System erneut starten zu müssen. Mit dieser Funktionalität können Sie ein Upgrade der Anwendung ohne Unterbrechung des Datengrids durchführen.

Plug-in-Beans und -Services können mit angepassten Shard-Geltungsbereichen konfiguriert werden. Dies bietet Ihnen differenzierte Integrationsmöglichkeiten mit anderen Services, die im Datengrid ausgeführt werden. Jedes Plug-in kann OSGi-Blueprint-Rankings verwenden, um sicherzustellen, dass jede Instanz des Plug-ins mit der richtigen Version aktiviert wird. Mit der bereitgestellten OSGi-Managed-Bean (MBean) und dem bereitgestellten Dienstprogramm **xscmd** können Sie die OSGi-Services des eXtreme-Plug-ins und deren Rankings abfragen.

Auf diese Weise können Administratoren potenzielle Konfigurations- und Verwaltungsfehler schnell erkennen und die Plug-in-Service-Rankings, die von eXtreme Scale verwendet werden, aktualisieren.

OSGi-Bundles

Für die Interaktion mit und die Implementierung von Plug-ins im OSGi-Framework müssen Sie *Bundles* verwenden. In der OSGi-Serviceplattform ist ein Bundle eine JAR-Datei (Java-Archiv), das Java-Code, Ressourcen und ein Manifest enthält, das das Bundle und dessen Abhängigkeiten beschreibt. Das Bundle ist die Implementierungseinheit für eine Anwendung. Das Produkt eXtreme Scale unterstützt die folgenden Bundletypen:

Server-Bundle
Das Server-Bundle ist die Datei `objectgrid.jar`, die mit der eigenständigen eXtreme-Scale-Serverinstallation installiert wird und die für die Ausführung von eXtreme-Scale-Servern erforderlich ist. Das Server-Bundle kann auch für die Ausführung von eXtreme-Scale-Clients oder lokalen speicherinternen Caches verwendet werden. Die Bundle-ID für die Datei `objectgrid.jar` ist "com.ibm.websphere.xs.server_<Version>", wobei Version das folgende Format hat: `<Version>.<Release>.<Modifikation>`. Das Server-Bundle für eXtreme Scale Version 7.1.1 ist beispielsweise com.ibm-.websphere.xs.server_7.1.1.

Client-Bundle
Das Client-Bundle ist die Datei `ogclient.jar` und wird mit eigenständigen und Clientinstallationen von eXtreme Scale installiert. Es wird verwendet, um eXtreme-Scale-Clients oder lokale Speichercaches auszuführen. Die Bundle-ID für die Datei `ogclient.jar` ist "com.ibm.websphere.xs.client-_Version", wobei Version das folgende Format hat: `<Version>.<Release>.<Modifikation>`. Das Client-Bundle für eXtreme Scale Version 7.1.1 ist beispielsweise "com.ibm.websphere.xs.client_7.1.1".

Einschränkungen

Sie können das eXtreme-Scale-Bundle nicht erneut starten, weil Sie den Object Request Broker (ORB) oder eXtremeIO (XIO) nicht erneut starten können. Zum erneuten Starten des eXtreme-Scale-Servers müssen Sie das OSGi-Framework erneut starten.

Eclipse-Equinox-OSGi-Framework mit Eclipse Gemini für Clients und Server installieren

Java

Wenn Sie WebSphere eXtreme Scale im OSGi-Framework implementieren möchten, müssen Sie die Eclipse-Equinox-Umgebung einrichten.

Informationen zu diesem Vorgang

Diese Aufgabe erfordert, dass Sie das Blueprint-Framework herunterladen und installieren, das Ihnen ermöglicht, JavaBeans zu einem späteren Zeitpunkt zu konfigurieren und als Services bereitzustellen. Die Verwendung von Services ist wichtig, weil Sie Plug-ins als OSGi-Services bereitstellen können, damit diese von der Laufzeitumgebung von eXtreme Scale verwendet werden können. Das Produkt unterstützt zwei Blueprint-Container im Eclipse-Equinox-Basis-OSGi-Framework: Eclipse Gemini und Apache Aries. Verwenden Sie die folgende Prozedur, um den Eclipse-Gemini-Container zu konfigurieren.

Vorgehensweise

1. Laden Sie Eclipse Equinox SDK Version 3.6.1 oder höher von der Eclipse-Website herunter. Erstellen Sie ein Verzeichnis für das Equinox-Framework, z. B. `/opt/equinox`. In den folgenden Anweisungen wird das Verzeichnis `equinox_root` verwendet. Entpacken Sie die komprimierte Datei im Verzeichnis `equinox_root`.
2. Laden Sie die komprimierte Datei für gemini-blueprint incubation 1.0.0 von der Eclipse-Website herunter. Extrahieren Sie den Dateiinhalt in ein temporäres Verzeichnis, und kopieren Sie die folgenden extrahierten Dateien in das Verzeichnis `equinox_root/plugins`:

```
dist/gemini-blueprint-core-1.0.0.jar
dist/gemini-blueprint-extender-1.0.0.jar
dist/gemini-blueprint-io-1.0.0.jar
```

Achtung: Je nach Position, an die Sie die komprimierte Blueprint-Datei heruntergeladen haben, können die entpackten Dateien entweder die Erweiterung `RELEASE.jar` haben, ähnlich wie die JAR-Dateien des Spring-Frameworks im nächsten Schritt. Sie müssen sicherstellen, dass die Dateinamen den Dateireferenzen in der Datei `config.ini` entsprechen.

3. Laden Sie Spring Framework Version 3.0.5 von der folgenden Webseite mit dem Spring-Quellcode herunter: http://www.springsource.com/download/community. Entpacken Sie die Datei in einem temporären Verzeichnis, und kopieren Sie die folgenden extrahierten Dateien in das Verzeichnis `equinox_root/plugins`:

   ```
   org.springframework.aop-3.0.5.RELEASE.jar
   org.springframework.asm-3.0.5.RELEASE.jar
   org.springframework.beans-3.0.5.RELEASE.jar
   org.springframework.context-3.0.5.RELEASE.jar
   org.springframework.core-3.0.5.RELEASE.jar
   org.springframework.expression-3.0.5.RELEASE.jar
   ```

4. Laden Sie die JAR-Datei (Java-Archiv) von AOP Alliance von der Webseite SpringSource herunter. Kopieren Sie die Datei "com.springsource.org.aopalliance-1.0.0.jar" in das Verzeichnis `equinox_root/plugins`.
5. Laden Sie JAR-Datei von Apache Commons Logging 1.1.1 von der Webseite SpringSource herunter. Kopieren Sie die Datei `com.springsource.org.apache.commons.logging-1.1.1.jar` in das Verzeichnis `equinox_root/plugins`.
6. Laden Sie den Befehlszeilenclient "Luminis OSGi Configuration Admin" herunter. Verwenden Sie dieses JAR-Dateibundle für die Verwaltung von OSGi-Verwaltungskonfigurationen. Kopieren Sie die Datei `net.luminis.cmc-0.2.5.jar` in das Verzeichnis `equinox_root/plugins`.
7. Laden Sie das Bundle für Apache Felix File Installation Version 3.0.2 von der folgenden Webseite herunter: http://felix.apache.org/site/index.html. Kopieren Sie die Datei `org.apache.felix.fileinstall-3.0.2.jar` in das Verzeichnis `equinox_root/plugins`.
8. Erstellen Sie ein Konfigurationsverzeichnis im Verzeichnis `equinox_root/plugins`, z. B.:

   ```
   mkdir equinox_root/plugins/configuration
   ```

9. Erstellen Sie die folgende Datei `config.ini` im Verzeichnis `equinox_root/plugins/configuration`, und ersetzen Sie *equinox_root* durch den absoluten Pfad zu Ihrem Verzeichnis, das Sie anstelle von `equinox_root` verwenden, und entfernen Sie alle abschließenden Leerzeichen hinter dem Backslash in der jeder Zeile. Sie müssen am Ende der Datei eine Leerzeile einfügen, z. B.:

   ```
   osgi.noShutdown=true
   osgi.java.profile.bootdelegation=none
   org.osgi.framework.bootdelegation=none
   eclipse.ignoreApp=true
   osgi.bundles=\
   org.eclipse.osgi.services_3.2.100.v20100503.jar@1:start, \
   org.eclipse.osgi.util_3.2.100.v20100503.jar@1:start, \
   org.eclipse.equinox.cm_1.0.200.v20100520.jar@1:start, \
   com.springsource.org.apache.commons.logging-1.1.1.jar@1:start, \
   com.springsource.org.aopalliance-1.0.0.jar@1:start, \
   org.springframework.aop-3.0.5.RELEASE.jar@1:start, \
   org.springframework.asm-3.0.5.RELEASE.jar@1:start, \
   org.springframework.beans-3.0.5.RELEASE.jar@1:start, \
   org.springframework.context-3.0.5.RELEASE.jar@1:start, \
   org.springframework.core-3.0.5.RELEASE.jar@1:start, \
   org.springframework.expression-3.0.5.RELEASE.jar@1:start, \
   org.apache.felix.fileinstall-3.0.2.jar@1:start, \
   ```

```
net.luminis.cmc-0.2.5.jar@1:start, \
gemini-blueprint-core-1.0.0.jar@1:start, \
gemini-blueprint-extender-1.0.0.jar@1:start, \
gemini-blueprint-io-1.0.0.jar@1:start
```

Wenn Sie die Umgebung bereits eingerichtet haben, können Sie das Equinox-Plug-in-Repository bereinigen, indem Sie das folgende Verzeichnis entfernen: `equinox_root\plugins\configuration\org.eclipse.osgi`.

10. Führen Sie die folgenden Befehle aus, um die Equinox-Konsole zu starten.

    Wenn Sie eine andere Version von Equinox verwenden, ist der Name Ihrer JAR-Datei anders als der Name, der im folgenden Beispiel verwendet wird:

    ```
    java -jar plugins\org.eclipse.osgi_3.6.1.R36x_v20100806.jar -console
    ```

eXtreme-Scale-Bundles installieren

Java

WebSphere eXtreme Scale enthält Bundles, die in einem Eclipse-Equinox-OSGi-Framework installiert werden können. Diese Bundles sind erforderlich, um eXtreme-Scale-Server zu starten oder um eXtreme-Scale-Clients in OSGi zu verwenden. Sie können die Bundles von eXtreme Scale mit der Equinox-Konsole oder mithilfe der Konfigurationsdatei "config.ini" installieren.

Vorbereitende Schritte

In dieser Task wird davon ausgegangen, dass Sie die folgenden Produkte installiert haben:

- Eclipse-Equinox-OSGi-Framework
- Eigenständiger eXtreme-Scale-Client oder -Server

Informationen zu diesem Vorgang

eXtreme Scale enthält zwei Bundles. In einem OSGi-Framework ist nur eines der folgenden Bundles erforderlich:

objectgrid.jar
: Das Server-Bundle ist die Datei `objectgrid.jar`, die mit der eigenständigen eXtreme-Scale-Serverinstallation installiert wird und die für die Ausführung von eXtreme-Scale-Servern erforderlich ist. Das Server-Bundle kann auch für die Ausführung von eXtreme-Scale-Clients oder lokalen speicherinternen Caches verwendet werden. Die Bundle-ID für die Datei `objectgrid.jar` ist "com.ibm.websphere.xs.server_<Version>", wobei Version das folgende Format hat: `<Version>.<Release>.<Modifikation>`. Das Server-Bundle für dieses Release ist beispielsweise `com.ibm.websphere.xs.server_8.5.0`.

ogclient.jar
: Das Bundle `ogclient.jar` wird mit eigenständigen Installationen und Clientinstallationen von eXtreme Scale installiert und wird verwendet, um eXtreme-Scale-Clients oder lokale speicherinterne Caches auszuführen. eXtreme Scale Die Bundle-ID für die Datei `ogclient.jar` ist "com.ibm.websphere.xs.client_<Version>", wobei die Version das folgende Format hat: `<Version>_<Release>_<Modifikation>`. Das Client-Bundle für dieses Release ist beispielsweise `com.ibm.websphere.xs.server_8.5.0`.

Weitere Informationen zum Entwickeln von eXtreme-Scale-Plug-ins finden Sie im Abschnitt System-APIs und Plug-ins.

Client- oder Server-Bundle von eXtreme Scale über die Equinox-Konsole im Eclipse-Equinox-OSGi-Framework installieren:
Vorgehensweise

1. Starten Sie das Eclipse-Equinox-Framework mit aktivierter Konsole, z. B.:

   ```
   Java-Ausgangsverzeichnis/bin/java -jar <equinox-Stammverzeichnis>/
   plugins/org.eclipse.osgi_3.6.1.R36x_v20100806.jar -console
   ```

2. Installieren Sie das Client- oder Server-Bundle von eXtreme Scale in der Equinox-Konsole:

   ```
   osgi> install file:///<Pfad_zum_Bundle>
   ```

3. Equinox zeigt die Bundle-ID für das neu installierte Bundle an:

   ```
   Bundle id is 25
   ```

4. Starten Sie das Bundle in der Equinox-Konsole, wobei <ID> für die Bundle-ID steht, die dem Bundle bei der Installation zugeordnet wurde:

   ```
   osgi> start <ID>
   ```

5. Rufen Sie den Servicestatus in der Equinox-Konsole ab, um sicherzustellen, dass das Bundle gestartet wurde, z. B.:

   ```
   osgi> ss
   ```

   Wenn das Bundle erfolgreich gestartet wird, wird der Status ACTIVE für das Bundle angezeigt, z. B.:

   ```
   25      ACTIVE      com.ibm.websphere.xs.server_8.5.0
   ```

Client- oder Server-Bundle von eXtreme Scale mit der Datei config.ini im Eclipse-Equinox-OSGi-Framework installieren:
Vorgehensweise

1. Kopieren Sie das eXtreme-Scale-Client- oder Server-Bundle (objectgrid.jar oder ogclient.jar) aus dem Verzeichnis <WXS-Installationsstammverzeichnis>/ObjectGrid/lib in das Eclipse-Equinox-Plug-in-Verzeichnis, z. B.: <equinox-Stammverzeichnis>/plugins

2. Bearbeiten Sie die Eclipse-Equinox-Konfigurationsdatei config.ini, und fügen Sie das Bundle der Eigenschaft "osgi.bundles" hinzu, z. B.:

   ```
   osgi.bundles=\
   org.eclipse.osgi.services_3.2.100.v20100503.jar@1:start, \
   org.eclipse.osgi.util_3.2.100.v20100503.jar@1:start, \
   org.eclipse.equinox.cm_1.0.200.v20100520.jar@1:start, \
   objectgrid.jar@1:start
   ```

   Wichtig: Vergewissern Sie sich, dass dem letzten Bundlenamen eine leere Zeile folgt. Jedes Bundle wird durch ein Komma abgetrennt.

3. Starten Sie das Eclipse-Equinox-Framework mit aktivierter Konsole, z. B.:

   ```
   Java-Ausgangsverzeichnis/bin/java -jar <equinox-Stammverzeichnis>/
   plugins/org.eclipse.osgi_3.6.1.R36x_v20100806.jar -console
   ```

4. Rufen Sie den Servicestatus in der Equinox-Konsole ab, um sicherzustellen, dass das Bundle gestartet wurde:

   ```
   osgi> ss
   ```

   Wenn das Bundle erfolgreich gestartet wird, wird der Status ACTIVE für das Bundle angezeigt, z. B.:

   ```
   25      ACTIVE      com.ibm.websphere.xs.server_8.5.0
   ```

Ergebnisse

Sie haben das Server- oder Client-Bundle von eXtreme Scale im Eclipse-Equinox-OSGi-Framework installiert und gestartet.

eXtreme-Scale-Container mit nicht dynamischen Plug-ins in einer OSGi-Umgebung ausführen

Wenn Sie die dynamischen Funktionen einer OSGi-Umgebung nicht verwenden müssen, können Sie trotzdem die starre Verbindung, das deklarative Packen und die Serviceabhängigkeiten nutzen, die das OSGi-Framework bietet.

Vorbereitende Schritte

1. Entwickeln Sie Ihre Anwendung mit den APIs und Plug-ins von WebSphere eXtreme Scale.
2. Packen Sie die Anwendung mit den entsprechenden Import- bzw. Exportabhängigkeiten, die in einem oder mehreren Bundlemanifesten deklariert sind, in ein oder mehrere OSGi-Bundles. Stellen Sie sicher, dass alle Klassen und Pakete, die für die Plug-ins, Agenten, Datenobjekte usw. erforderlich sind, exportiert werden.

Informationen zu diesem Vorgang

Mit dynamischen Plug-ins können Sie ein Upgrade Ihrer Plug-ins durchführen, ohne das Grid zu stoppen. Zur Verwendung dieser Funktionalität müssen die ursprünglichen und die neuen Plug-ins kompatibel sein. Wenn Sie keine Plug-ins aktualisieren müssen oder es sich leisten können, das Grid zu stoppen, um die Plug-ins zu aktualisieren, benötigen Sie die Komplexität dynamischer Plug-ins möglicherweise nicht. Es gibt trotzdem gute Gründe für die Ausführung Ihrer Anwendung von eXtreme Scale in einer OSGi-Umgebung. Zu diesen Gründen gehören die starre Verbindung, das deklarative Packen, die Serviceabhängigkeiten usw.

Eine Problemstellung beim Hosten des Grids oder Clients in einer OSGi-Umgebung ohne dynamische Plug-ins (d. h. ohne Deklaration der Plug-ins mit OSGi-Services) ist die Art und Weise, in der das Bundle von eXtreme Scale die Plug-in-Klassen lädt. Das Bundle von eXtreme Scale stützt sich beim Laden von Plug-in-Klassen auf OSGi-Services, sodass das Bundle Objektmethoden in Klassen anderer Bundles aufrufen kann, ohne die Pakete dieser Klassen direkt zu importieren.

Wenn die Plug-ins nicht über OSGi-Services bereitgestellt werden, muss das Bundle von eXtreme Scale in der Lage sein, die Plug-in-Klassen direkt zu importieren. Anstatt das Manifest des Bundles von eXtreme Scale für den Import der Benutzerklassen und Pakete zu ändern, erstellen Sie ein Bundlefragment, das die erforderlichen Paketimporte hinzufügt. Das Fragment kann auch die Klassen für andere Nicht-Plug-in-Benutzerklassen wie Datenobjekte und Agentenklassen importieren.

Vorgehensweise

1. Erstellen Sie ein OSGi-Fragment, das das Bundle von eXtreme Scale (Client oder Server, je nach geplanter Implementierungsumgebung) als Host verwendet. Das Fragment deklariert die Abhängigkeiten (Import-Package) in allen Paketen, die von einem oder mehreren Plug-ins geladen werden müssen. Wenn Sie beispielsweise ein Plug-in für eine Serialisierungsmethode installieren, dessen Klassen sich im Paket `com.mycompany.myapp.serializers` befinden und das von Klassen im Paket `com.mycompany.myapp.common` abhängig ist, gleicht die Datei META-INF/MANIFEST.MF für Ihr Fragment dem folgenden Beispiel:

```
Bundle-ManifestVersion: 2
Bundle-Name: Plug-in fragment for XS serializers
Bundle-SymbolicName: com.mycompany.myapp.myfragment; singleton:=true
Bundle-Version: 1.0.0
Fragment-Host: com.ibm.websphere.xs.server; bundle-version=7.1.1
Manifest-Version: 1.0
Import-Package: com.mycompany.myapp.serializers,
 com.mycompany.myapp.common
...
```

Dieses Manifest muss in eine Fragment-JAR-Datei gepackt werden, die in diesem Beispiel den Namen `com.mycompany.myapp.myfragment_1.0.0.jar` hat.

2. Implementieren Sie das neu erstellte Fragment, das Bundle von eXtreme Scale und die Anwendungsbundles in Ihrer OSGi-Umgebung. Starten Sie jetzt die Bundles.

Ergebnisse

Jetzt können Sie Ihre Anwendung in der OSGi-Umgebung testen und ausführen, ohne OSGi-Services wie Plug-ins und Agenten zum Laden der Benutzerklassen zu verwenden.

eXtreme-Scale-Server und -Anwendungen in einer OSGi-Umgebung verwalten

Verwenden Sie diesen Abschnitt, um das Bundle für den Server von WebSphere eXtreme Scale zu installieren. Dieses Bundle ist ein optionales Fragment, das das Laden der Anwendungsbundles und nicht dynamischer Benutzerklassen wie Plug-ins, Agenten, Datenobjekte usw. ermöglicht.

Vorbereitende Schritte

1. Installieren und starten Sie ein unterstütztes OSGi-Framework. Momentan ist Equinox die einzige unterstützte OSGi-Implementierung. Wenn Ihre Anwendung Blueprint verwendet, stellen Sie sicher, dass eine unterstützte Blueprint-Implementierung installiert und gestartet wird. Apache Aries und Eclipse Gemini werden unterstützt.
2. Öffnen Sie die OSGi-Konsole

Vorgehensweise

1. Installieren Sie das Bundle für den Server von eXtreme Scale. Sie müssen den Datei-URL der Bundle-JAR-Datei kennen. Beispiel:

```
osgi> install file:///home/user1/myOsgiEnv/plugins/objectgrid.jar
Bundle id is 41

osgi>
```

Das Bundle von eXtreme Scale ist jetzt installiert, aber noch nicht aufgelöst.

2. Wenn der Server von eXtreme Scale Benutzerklassen direkt laden muss, anstatt dynamische Plug-ins zu verwenden, die über OSGi-Services bereitgestellt werden, müssen Sie außerdem ein manuell entwickeltes Fragment installieren, das diese Klassen bereitstellt oder diese importiert. Wenn Sie dynamische Plug-ins und keeine Agenten verwenden, können Sie diesen Schritt überspringen. Im Folgenden sehen Sie ein Beispiel für die Installation eines angepassten Fragments:

```
osgi> install file:///home/user1/myOsgiEnv/plugins/myFragment.jar
Bundle id is 42
```

```
osgi> ss

Framework is launched.

id State       Bundle
...
41 INSTALLED   com.ibm.websphere.xs.server_7.1.1
42 INSTALLED   com.mycompany.myfragment_1.0.0

osgi>
```

Jetzt sind das Bundle für den Server von eXtreme Scale und das angepasste Fragment, das dem Bundle zugeordnet ist, installiert.

3. Starten Sie das Bundle für den Server von eXtreme Scale, z. B.:

```
osgi> start 41

osgi> ss

Framework is launched.

id State       Bundle
...
41 ACTIVE      com.ibm.websphere.xs.server_7.1.1
               Fragments=42
42 RESOLVED    com.mycompany.myfragment_1.0.0
               Master=41

osgi>
```

4. Jetzt installieren und starten Sie alle Benutzeranwendungsbundle mit denselben zuvor beschriebenen Befehlen. Zum Starten eines Grids in diesem Server, müssen die Server- und Containerdefinitionen mit Blueprint deklariert werden, oder die Anwendung muss den Server und den Container programmgesteuert über einen Bundleaktivator oder einen anderen Mechanismus gestartet werden.

Ergebnisse

Das Bundle für den Server von eXtreme Scale und die Anwendung sind implementiert, gestartet und für die Annahme von Anforderungen bereit.

Dynamische eXtreme-Scale-Plug-ins für die Verwendung in einer OSGi-Umgebung erstellen und ausführen

Alle Plug-ins von eXtreme Scale können für eine OSGi-Umgebung konfiguriert werden. Der Hauptvorteil dynamischer Plug-ins ist die Möglichkeit, die Plug-ins aktualisieren zu können, ohne das Grid beenden zu müssen. Dies ermöglicht die Weiterentwicklung einer Anwendung ohne Neustart der Grid-Container-Prozesse.

Informationen zu diesem Vorgang

Die OSGi-Unterstützung von WebSphere eXtreme Scale ermöglicht Ihnen, das Produkt in einem OSGi-Framework wie Eclipse Equinox zu implementieren. Zum Aktualisieren der von eXtreme Scale verwendeten Plug-ins mussten Sie früher die Java Virtual Machine (JVM) erneut starten, um die neuen Versionen der Plug-ins anzuwenden. Mit der Unterstützung dynamischer Plug-ins in eXtreme Scale und der Möglichkeit, die vom OSGi-Framework bereitgestellten Bundles aktualisieren zu können, können Sie die Plug-in-Klassen jetzt ohne Neustart der JVM aktualisieren. Diese Plug-ins werden von *Bundles* als Services exportiert. WebSphere eXtreme Scale greift auf den Service zu, indem dieser in der OSGi-Registry gesucht wird. In der OSGi-Serviceplattform ist ein Bundle eine JAR-Datei (Java-Archiv), das Java-

Code, Ressourcen und ein Manifest enthält, das das Bundle und dessen Abhängigkeiten beschreibt. Das Bundle ist die Implementierungseinheit für eine Anwendung.

Vorgehensweise

1. Dynamische eXtreme-Scale-Plug-ins erstellen
2. eXtreme-Scale-Plug-ins mit OSGi Blueprint konfigurieren
3. OSGi-fähige Plug-ins installieren und starten

Dynamische eXtreme-Scale-Plug-ins erstellen

Java

WebSphere eXtreme Scale enthält ObjectGrid- und BackingMap-Plug-ins. Diese Plug-ins werden in Java implementiert und mithilfe der ObjectGrid-XML-Deskriptordatei konfiguriert. Wenn Sie ein dynamisches Plug-in erstellen möchten, das dynamisch aktualisiert werden kann, müssen die Plug-ins ObjectGrid- und BackingMap-Lebenszyklusereignisse erkennen, weil sie während einer Aktualisierung unter Umständen einige Aktionen ausführen müssen. Die Erweiterung eines Plug-in-Bundles mit Callback-Methoden und/oder Ereignislistenern für den Lebenszyklus ermöglicht dem Plug-in, diese Aktionen zu den entsprechenden Zeiten auszuführen.

Vorbereitende Schritte

In diesem Abschnitt wird angenommen, dass Sie das entsprechende Plug-in erstellt haben. Weitere Informationen zum Entwickeln von eXtreme-Scale-Plug-ins finden Sie unter System-APIs und Plug-ins.

Informationen zu diesem Vorgang

Alle Plug-ins von eXtreme Scale gelten entweder für eine BackingMap- oder ObjectGrid-Instanz. Viele Plug-ins interagieren auch mit anderen Plug-ins. Ein Loader- und ein TransactionCallback-Plug-in arbeiten beispielsweise zusammen, um ordnungsgemäß mit einer Datenbanktransaktion und den verschiedenen Datenbank-JDBC-Aufrufen zu interagieren. Einige Plug-ins müssen unter Umständen auch Konfigurationsdaten aus anderen Plug-ins zwischenspeichern, um die Leistung zu verbessern.

Die Plug-ins BackingMapLifecycleListener und ObjectGridLifecycleListener stellen Lebenszyklusoperationen für die entsprechenden BackingMap- und ObjectGrid-Instanzen bereit. Dieser Prozess ermöglicht die Benachrichtigung von Plug-ins, wenn die übergeordnete BackingMap- oder ObjectGrid-Instanz und die entsprechenden Plug-ins geändert werden. BackingMap-Plug-ins implementieren die Schnittstelle "BackingMapLifecyleListener", und ObjectGrid-Plug-ins implementieren die Schnittstelle "ObjectGridLifecycleListener". Diese Plug-ins werden automatisch aufgerufen, wenn sich der Lebenszyklus der übergeordneten BackingMap- oder ObjectGrid-Instanz ändert. Weitere Informationen zu Lebenszyklus-Plug-ins finden Sie im Abschnitt Plug-in-Lebenszyklen verwalten.

Sie können Bundles mit Lebenszyklusmethoden oder Ereignislistenern in den folgenden allgemeinen Aufgaben erweitern:

- Ressourcen wie Threads oder Messaging-Subskribenten starten und stoppen
- Festlegen, dass eine Benachrichtigung gesendet wird, wenn Peer-Plug-ins aktualisiert wurden, um somit den direkten Zugriff auf die Plug-ins und die Erkennung von Änderungen zu ermöglichen.

Wenn Sie direkt auf ein anderes Plug-in zugreifen, greifen Sie über den OSGi-Container auf dieses Plug-in zu, um sicherzustellen, dass alle Teile des Systems auf das richtige Plug-in verweisen. Wenn beispielsweise eine Komponente in der Anwendung direkt auf eine Instanz eines Plug-ins zugreift oder diese zwischenspeichert, verwaltet sie ihre Referenz auf diese Version des Plug-ins selbst nach einer dynamischen Aktualisierung des Plug-ins. Dieses Verhalten kann zu anwendungsbezogenen Problemen und zu Speicherverlusten führen. Schreiben Sie deshalb Code, der von dynamischen Plug-ins abhängig ist, die ihre Referenzen mit der OSGi-Semantik getService() abrufen. Wenn die Anwendung Plug-ins zwischenspeichern muss, empfängt sie über die Schnittstellen "ObjectGridLifecycleListener" und "BackingMapLifecycleListener" Lebenszyklusereignisse. Die Anwendung muss auch in der Lage sein, bei Bedarf ihren Cache threadsicher zu aktualisieren.

Alle Plug-ins von eXtreme Scale, die mit OSGi verwendet werden, müssen auch die entsprechenden BackingMapPlugin- bzw. ObjectGridPlugin-Schnittstellen implementieren. Neue Plug-ins wie die Schnittstelle "MapSerializerPlugin" setzen dieses Verfahren um. Diese Schnittstellen stellen der eXtreme-Scale-Laufzeitumgebung und OSGi eine konsistente Schnittstelle für die Injektion von Statusinformationen in das Plug-in und die Steuerung des Lebenszyklus bereit.

Verwenden Sie diese Aufgabe, um festzulegen, dass eine Benachrichtigung gesendet wird, wenn Peer-Plug-ins aktualisiert werden. Sie können eine Listener-Factory erstellen, die eine Listenerinstanz erzeugt.

Vorgehensweise

- Aktualisieren Sie die ObjectGrid-Plug-in-Klasse, um die Schnittstelle "ObjectGridPlugin" zu implementieren. Diese Schnittstelle enthält Methoden, mit denen eXtreme Scale initialisiert, die ObjectGrid-Instanz definiert und das Plug-in gelöscht werden kann. Sehen Sie sich das folgende Codebeispiel an:

```
package com.mycompany;
import com.ibm.websphere.objectgrid.plugins.ObjectGridPlugin;
...

public class MyTranCallback implements TransactionCallback, ObjectGridPlugin {

    private ObjectGrid og = null;

    private enum State {
        NEW, INITIALIZED, DESTROYED
    }

    private State state = State.NEW;

    public void setObjectGrid(ObjectGrid grid) {
        this.og = grid;
    }

    public ObjectGrid getObjectGrid() {
        return this.og;
    }
    void initialize() {
        // Plug-in-Initialisierung hier behandeln. Wird von
        // eXtreme Scale und nicht vom OSGi-Bean-Manager aufgerufen.
        state = State.INITIALIZED;
    }
    boolean isInitialized() {
        return state == State.INITIALIZED;
    }

    public void destroy() {
        // Löscht das Plug-in und gibt alle Ressourcen frei. Kann
        // vom OSGi-Bean-Manager oder von eXtreme Scale aufgerufen werden.
        state = State.DESTROYED;
    }

    public boolean isDestroyed() {
        return state == State.DESTROYED;
    }
}
```

- ObjectGrid-Plug-in-Klasse aktualisieren, um die Schnittstelle "ObjectGridLifecycleListener" zu implementieren. Sehen Sie sich das folgende Codebeispiel an:

```
package com.mycompany;
import com.ibm.websphere.objectgrid.plugins.ObjectGridLifecycleListener;
      import com.ibm.websphere.objectgrid.plugins.ObjectGridLifecycleListener.LifecycleEvent;
      ...

      public class MyTranCallback implements TransactionCallback, ObjectGridPlugin, ObjectGridLifecycleListener{
          public void objectGridStateChanged(LifecycleEvent event) {
              switch(event.getState()) {
              case NEW:
              case DESTROYED:
              case DESTROYING:
              case INITIALIZING:
                  break;
              case INITIALIZED:
                  // Loader oder MapSerializerPlugin mit OSGi
                  // oder direkt über die ObjectGrid-Instanz suchen.
                  lookupOtherPlugins()
                  break;
              case STARTING:
              case PRELOAD:
                  break;
              case ONLINE:
                  if (event.isWritable()) {
                      startupProcessingForPrimary();
                  } else {
                      startupProcessingForReplica();
                  }
                  break;
              case QUIESCE:
                  if (event.isWritable()) {
                      quiesceProcessingForPrimary();
                  } else {
                      quiesceProcessingForReplica();
                  }
                  break;
              case OFFLINE:
                  shutdownShardComponents();
                  break;
              }
          }
          ...
      }
```

- BackingMap-Plug-in aktualisieren. BackingMap-Plug-in-Klasse aktualisieren, um die Plug-in-Schnittstelle BackingMap zu implementieren. Diese Schnittstelle enthält Methoden, mit denen eXtreme Scale initialisiert, die BackingMap-Instanz definiert und das Plug-in gelöscht werden kann. Sehen Sie sich das folgende Codebeispiel an:

```
package com.mycompany;
import com.ibm.websphere.objectgrid.plugins.BackingMapPlugin;
...

public class MyLoader implements Loader, BackingMapPlugin {

    private BackingMap bmap = null;

    private enum State {
        NEW, INITIALIZED, DESTROYED
    }

    private State state = State.NEW;

    public void setBackingMap(BackingMap map) {
        this.bmap = map;
    }

    public BackingMap getBackingMap() {
        return this.bmap;
    }
    void initialize() {
        // Plug-in-Initialisierung hier behandeln. Wird von
        // eXtreme Scale und nicht vom OSGi-Bean-Manager aufgerufen.
        state = State.INITIALIZED;
    }
    boolean isInitialized() {
        return state == State.INITIALIZED;
    }

    public void destroy() {
        // Löscht das Plug-in und gibt alle Ressourcen frei. Kann
        // vom OSGi-Bean-Manager oder von eXtreme Scale aufgerufen werden.
        state = State.DESTROYED;
    }

    public boolean isDestroyed() {
        return state == State.DESTROYED;
    }
}
```

- Aktualisieren Sie die BackingMap-Plug-in-Klasse, um die Schnittstelle "BackingMapLifecycleListener" zu implementieren. Sehen Sie sich das folgende Codebeispiel an:

```
package com.mycompany;

      import com.ibm.websphere.objectgrid.plugins.BackingMapLifecycleListener;
      import com.ibm.websphere.objectgrid.plugins.BackingMapLifecycleListener.LifecycleEvent;
      ...

      public class MyLoader implements Loader, ObjectGridPlugin, ObjectGridLifecycleListener{
          ...
          public void backingMapStateChanged(LifecycleEvent event) {
              switch(event.getState()) {
              case NEW:
              case DESTROYED:
              case DESTROYING:
              case INITIALIZING:
                  break;
              case INITIALIZED:
                  // MapSerializerPlugin mit OSGi oder direkt
                  // über die ObjectGrid-Instanz suchen.
                  lookupOtherPlugins()
                  break;
              case STARTING:
              case PRELOAD:
                  break;
              case ONLINE:
                  if (event.isWritable()) {
                      startupProcessingForPrimary();
                  } else {
                      startupProcessingForReplica();
                  }
                  break;
              case QUIESCE:
                  if (event.isWritable()) {
                      quiesceProcessingForPrimary();
                  } else {
                      quiesceProcessingForReplica();
                  }
                  break;
              case OFFLINE:
                  shutdownShardComponents();
                  break;
              }
          }
          ...
      }
```

Ergebnisse

Durch die Implementierung der Schnittstelle ObjectGridPlugin oder BackingMapPlugin kann eXtreme Scale den Lebenszyklus Ihres Plug-ins zu den richtigen Zeiten aktualisieren.

Durch die Implementierung der Schnittstelle ObjectGridLifecycleListener oder BackingMapLifecycleListener wird das Plug-in automatisch als Listener der zugeordneten ObjectGrid- oder BackingMap-Lebenszyklusereignisse registriert. Das Ereignis INITIALIZING wird verwendet, um zu signalisieren, dass alle ObjectGrid- und BackingMap-Plug-ins initialisiert wurden und für Suchoperationen und Verwendung verfügbar sind. Das Ereignis ONLINE wird verwendet, um zu signalisieren, dass das ObjectGrid online und für die Verarbeitung von Ereignissen bereit ist.

eXtreme-Scale-Plug-ins mit OSGi Blueprint konfigurieren

Java

Alle ObjectGrid- und BackingMap-Plug-ins von eXtreme Scale können mit dem OSGi-Blueprint-Service, der mit Eclipse Gemini und Apache Aries bereitgestellt wird, als OSGi-Beans und -Services definiert werden.

Vorbereitende Schritte

Bevor Sie Ihre Plug-ins als OSGi-Services konfigurieren können, müssen Sie Ihre Plug-ins in ein OSGi-Bundle packen und sich mit den grundlegenden Prinzipien

der erforderlichen Plug-ins vertraut machen. Das Bundle muss die Server- bzw. Clientpakete von WebSphere eXtreme Scale sowie weitere abhängige Pakete, die von den Plug-ins benötigt werden, importieren oder eine Bundleabhängigkeit in den Server- bzw. Client-Bundles von eXtreme Scale erstellen. In diesem Abschnitt wird beschrieben, wie Sie die Blueprint-XML konfigurieren, um Plug-in-Beans zu erstellen und diese als OSGi-Services für eXtreme Scale bereitzustellen.

Informationen zu diesem Vorgang

Beans und Services werden in einer Blueprint-XML-Datei definiert, und der Blueprint-Container erkennt, erstellt und verbindet die Beans miteinander und stellt diese dann als Services bereit. Durch diesen Prozess werden die Beans anderen OSGi-Bundles, einschließlich den Server- und Client-Bundles von eXtreme Scale, zur Verfügung gestellt.

Wenn Sie angepasste Plug-in-Services für eXtreme Scale erstellen, muss das Bundle, in dem die Plug-ins gehostet werden sollen, für die Verwendung von Blueprint konfiguriert werden. Außerdem muss eine Blueprint-XML-Datei erstellt und im Bundle gespeichert werden. Informationen zum allgemeinen Verständnis der Spezifikation finden Sie unter Building OSGi applications with the Blueprint Container specification.

Vorgehensweise

1. Erstellen Sie eine Blueprint-XML-Datei. Sie können die Datei beliebig nennen. Sie müssen jedoch den Blueprint-Namespace einschließen.

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">
   ...
   </blueprint>
   ```

2. Erstellen Sie Bean-Definitionen in der Blueprint-XML-Datei für jedes eXtreme-Scale-Plug-in.

   Beans werden mit dem Element `<bean>` definiert, können mit anderen Bean-Referenzen verbunden werden und können Initialisierungsparameter enthalten.

   Wichtig: Beim Definieren einer Bean müssen Sie den richtigen Geltungsbereich verwenden. Blueprint unterstützt die Geltungsbereiche "Singleton" und "Prototyp". eXtreme Scale unterstützt auch einen angepassten Shard-Geltungsbereich.

   Definieren Sie die meisten eXtreme-Scale-Plug-ins als Beans mit dem Geltungsbereich "Prototyp" oder "Shard", weil alle Beans für jedes ObjectGrid-Shard bzw. jede BackingMap-Instanz, dem bzw. der sie zugeordnet ist, eindeutig sein muss. Beans mit dem Geltungsbereich "Shard" können hilfreich sein, wenn die Beans in anderen Kontexten verwendet werden, damit die richtige Instanz abgerufen wird.

   Zum Definieren einer Bean mit dem Geltungsbereich "Prototyp" verwenden Sie das Attribut `scope="prototype"` in der Bean:

   ```
   <bean id="myPluginBean" class="com.mycompany.MyBean" scope="prototype">
   ...
   </bean>
   ```

   Zum Definieren einer Bean mit dem Geltungsbereich "Shard" müssen Sie dem XML-Schema den Namespace `objectgrid` hinzufügen und das Attribut `scope="objectgrid:shard"` in der Bean verwenden:

   ```
   <?xml version="1.0" encoding="UTF-8"?>

   <blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
              xmlns:objectgrid="http://www.ibm.com/schema/objectgrid"
   ```

```
    xsi:schemaLocation="http://www.ibm.com/schema/objectgrid
                http://www.ibm.com/schema/objectgrid/objectgrid.xsd">

    <bean id="myPluginBean" class="com.mycompany.MyBean"
   scope="objectgrid:shard">
    ...
    </bean>

...
```

3. Erstellen Sie PluginServiceFactory-Bean-Definitionen für jede Plug-in-Bean. Alle eXtreme-Scale-Beans müssen eine definierte PluginServiceFactory-Bean haben, damit der richtige Bean-Geltungsbereich angewendet werden kann. eXtreme Scale enthält eine BlueprintServiceFactory, die Sie verwenden können. Sie enthält zwei Eigenschaften, die definiert werden müssen. Sie müssen die Eigenschaft blueprintContainer auf die `blueprintContainer`-Referenz und die Eigenschaft beanId auf den Bean-ID-Namen gesetzt werden. Wenn eXtreme Scale den Service für die Instanziierung der entsprechenden Beans sucht, sucht der Server die Bean-Komponenteninstanz mithilfe des Blueprint-Containers.

   ```
   bean id="myPluginBeanFactory"
       class="com.ibm.websphere.objectgrid.plugins.osgi.BluePrintServiceFactory">
         <property name="blueprintContainer" ref="blueprintContainer"/>
   <property name="beanId" value="myPluginBean" />
   </bean>
   ```

4. Erstellen Sie einen Servicemanager für jede PluginServiceFactory-Bean. Jeder Servicemanager stellt die PluginServiceFactory-Bean mithilfe des Elements `<service>` bereit. Das Element "service" gibt den Namen an, unter dem die Bean OSGi bereitgestellt wird, die Referenz auf die PluginServiceFactory-Bean, die bereitzustellende Schnittstelle und das Ranking des Service. eXtreme Scale verwendet das Service-Manager-Ranking, um Service-Upgrades durchzuführen, wenn das eXtreme-Scale-Grid aktiv ist. Wenn das Ranking nicht angegeben wird, nimmt das OSGi-Framework das Ranking 0 an. Weitere Informationen finden Sie im Abschnitt zum Aktualisieren von Service-Rankings.

   Blueprint enthält mehrere Optionen für die Konfiguration von Service-Managern. Zum Definieren eines einfachen Service-Managers für eine PluginServiceFactory-Bean erstellen Sie ein Element `<service>` für jede PluginServiceFactory-Bean:

   ```
   <service ref="myPluginBeanFactory"
       interface="com.ibm.websphere.objectgrid.plugins.osgi.PluginServiceFactory"
       ranking="1">
   </service>
   ```

5. Speichern Sie die Blueprint-XML-Datei im Plug-in-Bundle. Die Blueprint-XML-Datei muss im Verzeichnis `OSGI-INF/blueprint` gespeichert werden, damit sie vom Blueprint-Container erkannt wird.

   Wenn Sie die Blueprint-XML-Datei in einem anderen Verzeichnis speichern möchten, müssen Sie den folgenden Bundle-Blueprint-Manifestheader angeben:

   ```
   Bundle-Blueprint: OSGI-INF/blueprint.xml
   ```

Ergebnisse

Die eXtreme-Scale-Plug-ins sind jetzt für die Bereitstellung in einem OSGi-Blueprint-Container konfiguriert. Außerdem wurde die ObjectGrid-XML-Deskriptordatei so konfiguriert, dass sie auf die Plug-ins über den OSGi-Blueprint-Service verweist.

OSGi-fähige Plug-ins installieren und starten

In dieser Aufgabe installieren Sie das Bundle mit den dynamischen Plug-ins im OSGi-Framework. Anschließend starten Sie das Plug-in.

Vorbereitende Schritte

In diesem Abschnitt wird davon ausgegangen, dass die folgenden Aufgaben ausgeführt wurden:

- Sie haben das Server- oder Client-Bundle von eXtreme Scale im Eclipse-Equinox-OSGi-Framework installiert. Weitere Informationen finden Sie im Abschnitt „eXtreme-Scale-Bundles installieren" auf Seite 234.
- Sie haben mindestens ein dynamisches BackingMap- oder ObjectGrid-Plug-in implementiert. Weitere Informationen finden Sie im Abschnitt „Dynamische eXtreme-Scale-Plug-ins erstellen" auf Seite 239.
- Sie haben die dynamischen Plug-ins als OSGi-Services in OSGi-Bundles gepackt.

Informationen zu diesem Vorgang

In dieser Aufgabe wird beschrieben, wie Sie das Bundle über die Eclipse-Equinox-Konsole installieren. Das Bundle kann mit verschiedenen Methoden installiert werden, z. B. durch Ändern der Konfigurationsdatei `config.ini`. Produkte, in denen Eclipse Equinox integriert ist, haben alternative Methoden für die Verwaltung von Bundles. Informationen zum Hinzufügen von Bundles in der Datei `config.ini` in Eclipse Equinox finden Sie unter Eclipse runtime options.

OSGi ermöglcht das Starten von Bundles, die doppelte Services haben. WebSphere eXtreme Scale verwendet das aktuellste Service-Ranking. Wenn Sie mehrere OSGi-Frameworks in einem eXtreme-Scale-Datengrid starten, müssen Sie sicherstellen, dass die richtigen Service-Rankings in jedem Server gestartet werden. Wenn Sie dies nicht tun, wird das Grid mit verschiedenen Versionen gestartet.

Um festzustellen, welche Versionen vom Datengrid verwendet werden, überprüfen Sie mit dem Dienstprogramm "xscmd" die aktuellen und verfügbaren Rankings. Weitere Informationen zu den verfügbaren Service-Rankings finden Sie unter OSGi-Services für eXtreme-Scale-Plug-ins mit **xscmd** aktualisieren.

Vorgehensweise

Plug-in-Bundle über die OSGi-Konsole im Eclipse-Equinox-OSGi-Framework installieren.

1. Starten Sie das Eclipse-Equinox-Framework mit aktivierter Konsole, z. B.:
   ```
   <java_home>/bin/java -jar <equinox_root>/plugins/org.eclipse.osgi_3.6.1.R36x_v20100806.jar -console
   ```
2. Installieren Sie das Plug-in-Bundle in der Equinox-Konsole.
   ```
   osgi> install file:///<Pfad_zum_Bundle>
   ```
   Equinox zeigt die Bundle-ID für das neu installierte Bundle an:
   ```
   Bundle id is 17
   ```
3. Geben Sie die folgende Zeile ein, um das Bundle in der Equinox-Konsole zu starten, wobei <ID> für die Bundle-ID steht, die dem Bundle bei der Installation zugeordnet wurde:
   ```
   osgi> start <ID>
   ```
4. Rufen Sie den Servicestatus in der Equinox-Konsole ab, um sicherzustellen, dass das Bundle gestartet wurde:

```
osgi> ss
```

Wenn das Bundle erfolgreich gestartet wurde, wird der Status ACTIVE für das Bundle angezeigt, z. B.:

```
17      ACTIVE      com.mycompany.plugin.bundle_VRM
```

Plug-in-Bundle über die Datei "config.ini" im Eclipse-Equonix-OSGi-Framework installieren.

5. Kopieren Sie das Plug-in-Bundle in das Eclipse-Equinox-Plug-in-Verzeichnis, z. B.:

   ```
   <equinox_root>/plugins
   ```

6. Bearbeiten Sie die Eclipse-Equinox-Konfigurationsdatei `config.ini`, und fügen Sie das Bundle der Eigenschaft "osgi.bundles" hinzu, z. B.:

   ```
   osgi.bundles=\
   org.eclipse.osgi.services_3.2.100.v20100503.jar@1:start, \
   org.eclipse.osgi.util_3.2.100.v20100503.jar@1:start, \
   org.eclipse.equinox.cm_1.0.200.v20100520.jar@1:start, \
   com.mycompany.plugin.bundle_VRM.jar@1:start
   ```

   Wichtig: Vergewissern Sie sich, dass dem letzten Bundlenamen eine leere Zeile folgt. Jedes Bundle wird durch ein Komma abgetrennt.

7. Starten Sie das Eclipse-Equinox-Framework mit aktivierter Konsole, z. B.:

   ```
   <java_home>/bin/java -jar <equinox_root>/plugins/org.eclipse.osgi_3.6.1.R36x_v20100806.jar -console
   ```

8. Rufen Sie den Servicestatus in der Equinox-Konsole ab, um sicherzustellen, dass das Bundle gestartet wurde, z. B.:

   ```
   osgi> ss
   ```

   Wenn das Bundle erfolgreich gestartet wurde, wird der Status ACTIVE für das Bundle angezeigt, z. B.:

   ```
   17      ACTIVE      com.mycompany.plugin.bundle_VRM
   ```

Ergebnisse

Das Plug-in-Bundle ist jetzt installiert und gestartet. Der Container oder Client von eXtreme Scale kann jetzt gestartet werden. Weitere Informationen zum Entwickeln von eXtreme-Scale-Plug-ins finden Sie im Abschnitt System-APIs und Plug-ins.

eXtreme-Scale-Container mit dynamischen Plug-ins in einer OSGi-Umgebung ausführen

Wenn Ihre Anwendung im Eclipse-Equinox-OSGi-Framework mit Eclipse Gemini oder Apache Aries ausgeführt wird, können Sie diese Task verwenden, um Ihre Anwendung von WebSphere eXtreme Scale in OSGi zu installieren und zu konfigurieren.

Vorbereitende Schritte

Bevor Sie mit dieser Aufgabe beginnen, müssen Sie die folgenden Aufgaben ausgeführt haben:

- Eclipse-Equinox-OSGi-Framework mit Eclipse Gemini installieren
- Dynamische Plug-ins von eXtreme Scale in einer OSGi-Umgebung erstellen und ausführen

Informationen zu diesem Vorgang

Mit dynamischen Plug-ins können Sie das Plug-in dynamisch aktualisieren, während das Grid noch aktiv ist. Dies ermöglicht die Aktualisierung einer Anwendung ohne Neustart der Grid-Container-Prozesse. Weitere Informationen zum Entwickeln von eXtreme-Scale-Plug-ins finden Sie unter System-APIs und Plug-ins.

Vorgehensweise

1. OSGi-fähige Plug-ins mit der ObjectGrid-XML-Deskriptordatei konfigurieren
2. eXtreme-Scale-Container-Server mit dem Eclipse-Equinox-OSGi-Framework starten
3. OSGi-Services für eXtreme-Scale-Plug-ins mit dem Dienstprogramm xscmd verwalten
4. Server mit OSGi Blueprint konfigurieren

OSGi-fähige Plug-ins mit der ObjectGrid-XML-Deskriptordatei konfigurieren

Java

In dieser Aufgabe fügen Sie einer XML-Deskriptordatei OSGi-Services hinzu, sodass Container von WebSphere eXtreme Scale die OSGi-fähigen Plug-ins erkennen und ordnungsgemäß laden können.

Vorbereitende Schritte

Zum Konfigurieren Ihrer Plug-ins müssen Sie folgende Aktionen ausführen:

- Sie müssen Ihr Paket erstellen und dynamische Plug-ins für die OSGi-Implementierung aktivieren.
- Sie müssen die Namen der OSGi-Services kennen, die Ihre verfügbaren Plug-ins darstellen.

Informationen zu diesem Vorgang

Sie haben einen OSGi-Service für den Einschluss Ihres Plug-ins erstellt. Jetzt müssen diese Services in der Datei `objectgrid.xml` definiert werden, damit die Container von eXtreme Scale die Plug-ins erfolgreich laden und konfigurieren können.

Vorgehensweise

1. Alle gridspezifischen Plug-ins wie TransactionCallback müssen im Element objectGrid angegeben werden. Sehen Sie sich das folgende Beispiel aus der Datei `objectgrid.xml` an:

```
<?xml version="1.0" encoding="UTF-8"?>

<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
    xmlns="http://ibm.com/ws/objectgrid/config">

    <objectGrids>
        <objectGrid name="MyGrid" txTimeout="60">
            <bean id="myTranCallback" osgiService="myTranCallbackFactory"/>
            ...
        </objectGrid>
        ...
    </objectGrids>
    ...
/objectGridConfig>
```

Wichtig: Der Wert des Attributs `osgiService` muss mit dem Wert des Attributs `ref` übereinstimmen, der in der `Blueprint`XML-Datei bei der Definition des Service für myTranCallback PluginServiceFactory angegeben wurde.

2. Alle mapspezifischen Plug-ins wie beispielsweise Loader und Serializer müssen im Element backingMapPluginCollections angegeben und über das Element backingMap referenziert werden. Sehen Sie sich das folgende Beispiel aus der Datei `objectgrid.xml` an:

```
<?xml version="1.0" encoding="UTF-8"?>

objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
    xmlns="http://ibm.com/ws/objectgrid/config">
    <objectGrids>
        <objectGrid name="MyGrid" txTimeout="60">
            <backingMap name="MyMap1" lockStrategy="PESSIMISTIC"
                copyMode="COPY_TO_BYTES" nullValuesSupported="false"
                pluginCollectionRef="myPluginCollectionRef1"/>
            <backingMap name="MyMap2" lockStrategy="PESSIMISTIC"
                copyMode="COPY_TO_BYTES" nullValuesSupported="false"
                pluginCollectionRef="myPluginCollectionRef2"/>
            ...
        </objectGrid>
        ...
    </objectGrids>
    ...
    <backingMapPluginCollections>
        <backingMapPluginCollection id="myPluginCollectionRef1">
            <bean id="MapSerializerPlugin" osgiService="mySerializerFactory"/>
        </backingMapPluginCollection>
        <backingMapPluginCollection id="myPluginCollectionRef2">
            <bean id="MapSerializerPlugin" osgiService="myOtherSerializerFactory"/>
            <bean id="Loader" osgiService="myLoader"/>
        </backingMapPluginCollection>
        ...
    </backingMapPluginCollections>
    ...
</objectGridConfig>
```

Ergebnisse

Die Datei `objectgrid.xml` in diesem Beispiel weist eXtreme Scale an, ein Grid mit dem Namen `MyGrid` mit zwei Maps, `MyMap1` und `MyMap2`, zu erstellen. Die Map `MyMap1` verwendet den Serializer, der in den OSGi-Service eingeschlossen ist, mySerializerFactory. Die Map `MyMap2` verwendet einen Serializer aus dem OSGi-Service, myOtherSerializerFactory, und einen Loader aus dem OSGi-Service, myLoader.

Server von eXtreme Scale mit dem Eclipse-Equinox-OSGi-Framework starten

Container-Server von WebSphere eXtreme Scale können mit verschiedenen Methoden in einem Eclipse-Equinox-OSGi-Framework gestartet werden.

Vorbereitende Schritte

Bevor Sie einen eXtreme-Scale-Container starten können, müssen Sie die folgenden Aufgaben ausgeführt haben:

1. Sie haben das Server-Bundle von WebSphere eXtreme Scale in Eclipse Equinox installiert.
2. Sie haben Ihre Anwendung als OSGi-Bundle gepackt.
3. Sie haben Ihre Plug-ins von WebSphere eXtreme Scale (sofern vorhanden) als OSGi-Bundle gepackt. Die Plug-ins können in dasselbe Bundle wie die Anwendung oder als separate Bundles gepackt werden.
4. Wenn Ihre Container-Server IBM eXtremeMemory verwenden, müssen Sie zuerst die nativen Bibliotheken konfigurieren. Weitere Informationen finden Sie unter IBM eXtremeMemory konfigurieren.

Informationen zu diesem Vorgang

In dieser Aufgabe wird beschrieben, wie Sie einen eXtreme-Scale-Container-Server in einem Eclipse-Equinox-OSGi-Framework starten. Sie können jede der folgenden Methoden verwenden, um Container-Server mit der Eclipse-Equinox-Implementierung zu starten:

- OSGi-Blueprint-Service

  Sie können alle Konfigurations- und Metadaten in ein OSGi-Bundle einschließen. Sehen Sie sich die folgende Abbildung an, um sich mit dem Eclipse-Equinox-Prozess für diese Methode vertraut zu machen:

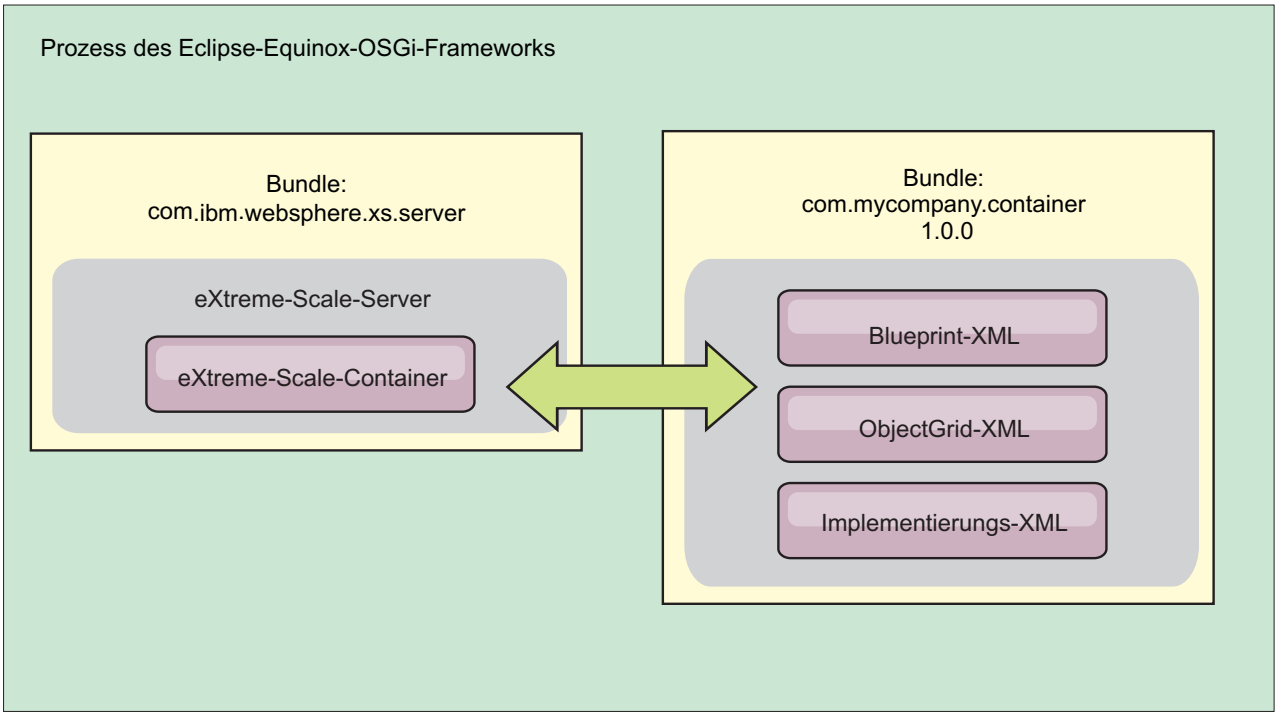


Abbildung 63. Eclipse-Equinox-Prozess für den Einschluss aller Konfigurations- und Metadaten in ein OSGi-Bundle

- Verwaltungsservice für OSGi-Konfiguration

  Sie können Konfigurations- und Metadaten außerhalb eines OSGi-Bundles angeben. Sehen Sie sich die folgende Abbildung an, um sich mit dem Eclipse-Equinox-Prozess für diese Methode vertraut zu machen:

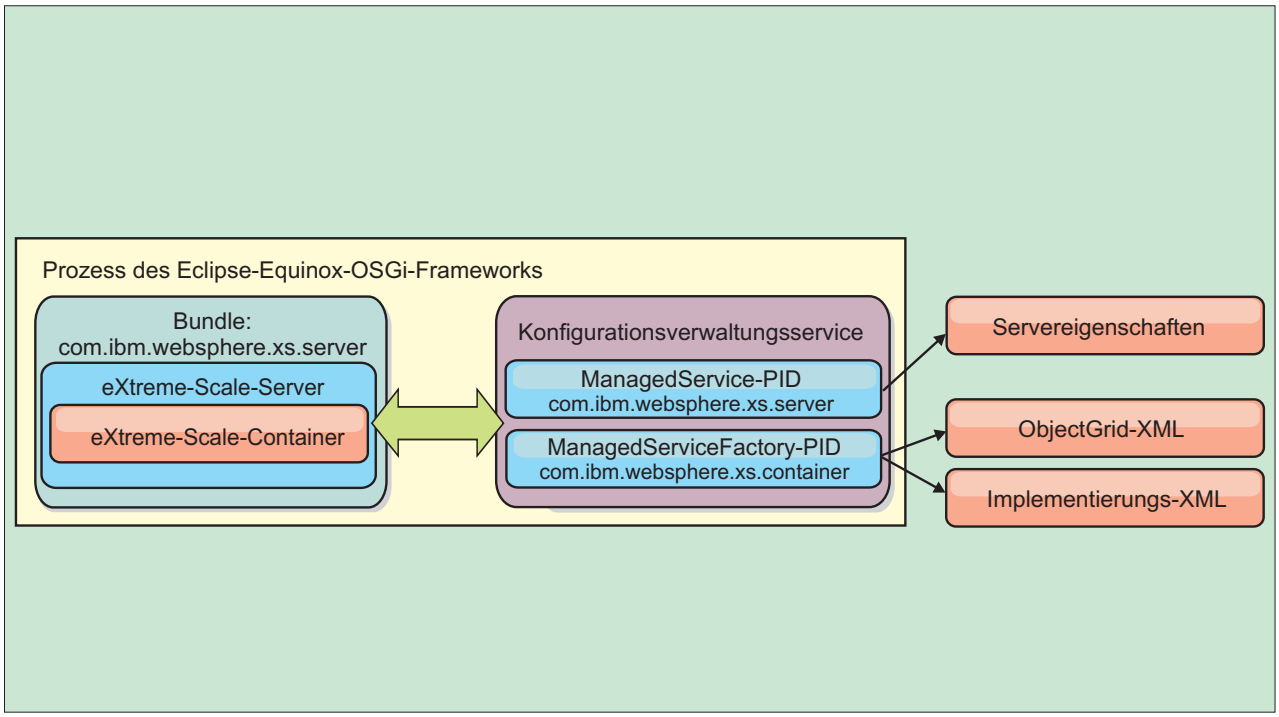


Abbildung 64. Eclipse-Equinox-Prozess für die Angabe von Konfigurations- und Metadaten außerhalb eines OSGi-Bundles

- Über das Programm
  Unterstützt angepasste Konfigurationslösungen.

In jedem Fall werden ein Server-Singleton von eXtreme Scale und mindestens ein Container konfiguriert.

Das Server-Bundle von eXtreme Scale, `objectgrid.jar`, enthält alle erforderlichen Bibliotheken zum Starten und Ausführen eines Grid-Containers von eXtreme Scale in einem OSGi-Framework. Die Laufzeitumgebung des Servers kommuniziert mit benutzerdefinierten Plug-ins und Datenobjekten über den OSGi-Service-Manager.

Wichtig: Nach dem Start eines Server-Bundles von eXtreme Scale und der Initialisierung des Servers von eXtreme Scale kann der Server nicht erneut gestartet werden. Zum erneuten Starten des Servers von eXtreme Scale muss der Eclipse-Equinox-Prozess erneut gestartet werden.

Sie können die Unterstützung für Spring-Namespaces von eXtreme Scale verwenden, um Container-Server von eXtreme Scale in einer Blueprint-XML-Datei zu konfigurieren. Wenn die XML-Elemente für Server und Container der Blueprint-XML-Datei hinzugefügt werden, startet der Namespace-Handler von eXtreme Scale automatisch einen Container-Server, wobei er die Parameter verwendet, die in der Blueprint-XML-Datei beim Start des Bundles definiert sind. Der Handler stoppt den Container, wenn das Bundle gestoppt wird.

Gehen Sie zum Konfigurieren von eXtreme-Scale-Container-Server mit der Blueprint-XML wie folgt vor:

Vorgehensweise

- Container-Server von eXtreme Scale mit OSGi-Blueprint starten.
  1. Erstellen Sie ein Container-Bundle.
  2. Installieren Sie das Container-Bundle im Eclipse-Equinox-OSGi-Framework. Weitere Informationen hierzu finden Sie im Abschnitt „OSGi-fähige Plug-ins installieren und starten” auf Seite 245.
  3. Starten Sie das Container-Bundle.
- Container-Server von eXtreme Scale mit der OSGi-Konfigurationsverwaltung starten.
  1. Konfigurieren Sie den Server und den Container mit der Konfigurationsverwaltung.
  2. Wenn das Server-Bundle von eXtreme Scale gestartet wird oder die persistenten IDs mit der Konfigurationsverwaltung erstellt werden, werden Server und Container automatisch gestartet.
- Container-Server von eXtreme Scale mit der API "ServerFactory" starten. Weitere Informationen hierzu finden Sie in der Dokumentation der Server-APIs.
  1. Erstellen Sie eine Aktivatorklasse für OSGi-Bundles, und verwenden Sie die API "ServerFactory" von eXtreme Scale, um einen Server zu starten.

OSGi-fähige Services mit dem Dienstprogramm `xscmd` verwalten

Sie können das Dienstprogramm **xscmd** verwenden, um Verwaltungsaufgaben wie das Anzeigen von Services und Rankings, die von jedem Container verwendet werden, und die Aktualisierung der Laufzeitumgebung zur Verwendung neuer Versionen der Bundles auszuführen.

Informationen zu diesem Vorgang

Mit dem Eclipse-Equinox-OSGi-Framework können Sie mehrere Versionen desselben Bundles installieren und diese Bundles zur Laufzeit aktualisieren. WebSphere eXtreme Scale ist eine verteilte Umgebung, in der die Container-Server in vielen OSGi-Framework-Instanzen ausgeführt werden.

Administratoren sind für das manuelle Kopieren, Installieren und Starten von Bundles im OSGi-Framework verantwortlich. eXtreme Scale enthält eine OSGi-Schnittstelle ServiceTrackerCustomizer, um Services zu überwachen, die als Plug-ins von eXtreme Scale in der ObjectGrid-XML-Deskriptordatei angegeben wurden. Verwenden Sie das Dienstprogramm **xscmd**, um festzustellen, welche Version des Plug-ins verwendet wird und welche Versionen verwendet werden können, und um Bundle-Upgrades durchzuführen.

eXtreme Scale verwendet die Service-Ranking-Nummer, um die Version jedes Service anzugeben. Wenn zwei oder mehr Services mit derselben Referenz geladen werden, verwendet eXtreme Scale automatisch den Service mit dem höchsten Ranking.

Vorgehensweise

- Führen Sie den Befehl **osgiCurrent** aus, und vergewissern Sie sich, dass jeder Server von eXtreme Scale das richtige Plug-in-Service-Ranking verwendet.

  Da eXtreme Scale automatisch die Servicereferenz mit dem höchsten Ranking auswählt, ist es möglich, dass das Datengrid mit mehreren Rankings eines Plug-in-Service gestartet wird.

Wenn der Befehl eine Diskrepanz bei den Rankings feststellt oder einen Service nicht findet, wird eine Fehlerkategorie ungleich null gesetzt. Wird der Befehl erfolgreich ausgeführt, wird die Fehlerkategorie auf 0 gesetzt.

Im folgenden Beispiel sehen Sie die Ausgabe des Befehls **osgiCurrent**, wenn zwei Plug-ins in demselben Grid mit vier Servern installiert sind. Das Plug-in loaderPlugin verwendet Ranking 1, und txCallbackPlugin verwendet Ranking 2.

```
OSGi Service Name Current Ranking ObjectGrid Name MapSet Name Server Name
----------------- --------------- --------------- ----------- -----------
loaderPlugin      1               MyGrid          MapSetA     server1
loaderPlugin      1               MyGrid          MapSetA     server2
loaderPlugin      1               MyGrid          MapSetA     server3
loaderPlugin      1               MyGrid          MapSetA     server4
txCallbackPlugin  2               MyGrid          MapSetA     server1
txCallbackPlugin  2               MyGrid          MapSetA     server2
txCallbackPlugin  2               MyGrid          MapSetA     server3
txCallbackPlugin  2               MyGrid          MapSetA     server4
```

Im folgenden Beispiel sehen Sie die Ausgabe des Befehls **osgiCurrent**, wenn server2 mit einem neueren Ranking von loaderPlugin gestartet wurde:

```
OSGi Service Name Current Ranking ObjectGrid Name MapSet Name Server Name
----------------- --------------- --------------- ----------- -----------
loaderPlugin      1               MyGrid          MapSetA     server1
loaderPlugin      2               MyGrid          MapSetA     server2
loaderPlugin      1               MyGrid          MapSetA     server3
loaderPlugin      1               MyGrid          MapSetA     server4
txCallbackPlugin  2               MyGrid          MapSetA     server1
txCallbackPlugin  2               MyGrid          MapSetA     server2
txCallbackPlugin  2               MyGrid          MapSetA     server3
txCallbackPlugin  2               MyGrid          MapSetA     server4
```

- Führen Sie den Befehl **osgiAll** aus, um sicherzustellen, dass die Plug-in-Services in jedem Container-Server von eXtreme Scale ordnungsgemäß gestartet wurden.

  Wenn Bundles gestartet werden, die Services enthalten, auf die eine ObjectGrid-Konfiguration verweist, überwacht die eXtreme-Scale-Laufzeitumgebung das Plug-in automatisch, verwendet es aber nicht sofort. Der Befehl **osgiAll** zeigt an, welche Plug-ins für jeden Server verfügbar sind.

  Wenn Sie den Befehl ohne Parameter ausführen, werden alle Services für alle Grids und Server angezeigt. Es können zusätzliche Filter, einschließlich des Filters "**-serviceName** <Servicename>, angegeben werden, um die Ausgabe auf einen einzigen Service oder einen Teil des Datengrids zu beschränken.

  Im folgenden Beispiel sehen Sie die Ausgabe des Befehls **osgiAll**, wenn zwei Plug-ins in zwei Servern gestartet werden. Im Plug-in loaderPlugin sind beide Rankings (1 und 2) gestartet, und im Plug-in txCallbackPlugin ist nur Ranking 1 gestartet. Die Übersichtsnachricht am Ende der Ausgabe bestätigt, dass beide Server dieselben Service-Rankings sehen:

```
Server: server1
   OSGi Service Name   Available Rankings
   -----------------   ------------------
   loaderPlugin        1, 2
   txCallbackPlugin    1

Server: server2
   OSGi Service Name   Available Rankings
   -----------------   ------------------
   loaderPlugin        1, 2
   txCallbackPlugin    1

Summary - All servers have the same service rankings.
```

Im folgenden Beispiel sehen Sie die Ausgabe des Befehls **osgiAll**, wenn das Bundle, das loaderPlugin mit Ranking 1 1 enthält, in server1 gestoppt wird. Die Übersichtsnachricht am Ende der Ausgabe bestätigt, dass loaderPlugin mit Ranking 1 jetzt in server1 fehlt.

```
Server: server1
   OSGi Service Name   Available Rankings
   -----------------   ------------------
   loaderPlugin        2
   txCallbackPlugin    1

Server: server2
   OSGi Service Name   Available Rankings
   -----------------   ------------------
   loaderPlugin        1, 2
   txCallbackPlugin    1

Summary - The following servers are missing service rankings:
   Server  OSGi Service Name Missing Rankings
   ------  ----------------- ----------------
   server1 loaderPlugin      1
```

Im folgenden Beispiel sehen Sie die Ausgabe des Befehls, wenn der Servicename mit dem Argument **-sn** angegeben wird, aber der Service nicht vorhanden ist:

```
Server: server2
   OSGi Service Name Available Rankings
   ----------------- ------------------
   invalidPlugin     No service found

Server: server1
   OSGi Service Name Available Rankings
   ----------------- ------------------
   invalidPlugin     No service found

Summary - All servers have the same service rankings.
```

- Führen Sie den Befehl **osgiCheck** aus, um zu prüfen, ob Gruppen von Plug-in-Services und -Rankings verfügbar sind.

  Der Befehl **osgiCheck** akzeptiert eine oder mehrere Gruppen von Service-Rankings im folgenden Format: `-serviceRankings <Servicename>;<Ranking>[,<Servicename>;<Ranking>]`

  Wenn alle Rankings verfügbar sind, kehrt die Methode mit der Fehlerkategorie 0 zurück. Ist mindestens ein Ranking nicht verfügbar, wird eine Fehlerkategorie ungleich 0 gesetzt. Außerdem wird eine Tabelle mit allen Servern angezeigt, die die angegebenen Service-Rankings nicht enthalten. Es können zusätzliche Filter verwendet werden, um die Serviceprüfung auf einen Teil der verfügbaren Server in der eXtreme-Scale-Domäne zu beschränken.

  Wenn beispielsweise das angegebene Ranking oder der angegebene Service fehlt, wird die folgende Nachricht angezeigt:

  ```
  Server  OSGi Service Unavailable Rankings
  ------  ------------ --------------------
  server1 loaderPlugin 3
  server2 loaderPlugin 3
  ```

- Führen Sie den Befehl **osgiUpdate** aus, um das Ranking eines oder mehrerer Plug-ins für alle Server in einem einzelnen ObjectGrid und MapSet in einer einzigen Operation zu aktualisieren.

  Der Befehl akzeptiert eine oder mehrere Gruppen von Service-Rankings im folgenden Format: `-serviceRankings <Servicename>;<Ranking>[,<Servicename>;<Ranking>] -g <Gridname> -ms <MapSet-Name>`

  Mit diesem Befehl können Sie die folgenden Operationen ausführen:

– Vergewissern Sie sich, dass die angegebenen Services zur Aktualisierung in allen Servern verfügbar sind.
– Ändern Sie den Status des Grids mit der Schnittstelle StateManager in "offline". Weitere Informationen finden Sie unter ObjectGrid-Verfügbarkeit verwalten. Dieser Prozess legt das Grid still und wartet, bis alle aktiven Transaktionen abgeschlossen sind, und verhindert, dass neue Transaktionen gestartet werden. Dieser Prozess weist außerdem alle ObjectGridLifecycleListener- und BackingMapLifecycleListener-Plug-ins an, alle transaktionsorientierten Aktivitäten einzustellen. Informationen zu Ereignislistener-Plug-ins finden Sie unter Plug-ins für die Bereitstellung von Ereignislistenern.
– Aktualisieren Sie alle Container von eXtreme Scale, die in einem OSGi-Framework ausgeführt werden, so, dass sie die neuen Serviceversionen verwenden.
– Ändern Sie den Status des Grids in "online", damit Transaktionen fortgesetzt werden können.

Der Aktualisierungsprozess ist insofern idempotent, dass er in dem Fall, dass ein Client eine Task nicht ausführen kann, bewirkt, dass die Operation rückgängig gemacht wird. Wenn ein Client das Rollback nicht durchführen kann oder während des Aktualisierungsprozesses unterbrochen wird, kann derselbe Befehl erneut abgesetzt und beim entsprechenden Schritt fortgesetzt werden.

Wenn der Client seine Aktivitäten nicht fortsetzen kann und der Prozess über einen anderen Client erneut gestartet wird, verwenden Sie die Option `-force`, um dem Client die Durchführung der Aktualisierung zu ermöglichen. Der Befehl **`osgiUpdate`** verhindert, dass mehrere Clients dasselbe MapSet gleichzeitig aktualisieren. Weitere Einzelheiten zum Befehl **`osgiUpdate`** finden Sie unter OSGi-Services für eXtreme-Scale-Plug-ins mit **`xscmd`** aktualisieren.

Server mit OSGi Blueprint konfigurieren

Java

Sie können Container-Server von WebSphere eXtreme Scale mit einer OSGi-Blueprint-XML-Datei konfigurieren, was das Packen und die Entwicklung eigenständiger Server-Bundles vereinfacht.

Vorbereitende Schritte

In diesem Abschnitt wird davon ausgegangen, dass die folgenden Aufgaben ausgeführt wurden:

- Das Eclipse-Equinox-OSGi-Framework wurde installiert und mit dem Eclipse-Gemini- oder Apache-Aries-Blueprint-Container gestartet.
- Das eXtreme-Scale-Server-Bundle wurde installiert und gestartet.
- Das Bundle mit den dynamischen eXtreme-Scale-Plug-ins wurde erstellt.
- Die ObjectGrid-XML-Deskriptordatei und die XML-Implementierungsrichtliniendatei von eXtreme Scale wurden erstellt.

Informationen zu diesem Vorgang

In dieser Aufgabe wird beschrieben, wie Sie einen eXtreme-Scale-Server mit einem Container über eine Blueprint-XML-Datei konfigurieren. Das Ergebnis dieser Prozedur ist ein Container-Bundle. Wenn das Container-Bundle gestartet wird, überwacht das eXtreme-Scale-Server-Bundle das Bundle, parst die Server-XML und startet Server und Container.

Ein Container-Bundle kann optional mit der Anwendung und den eXtreme-Scale-Plug-ins kombiniert werden, wenn dynamische Plug-in-Aktualisierungen nicht er-

forderlich sind oder die Plug-ins keine dynamische Aktualisierung unterstützen.

Vorgehensweise

1. Blueprint-XML-Datei mit eingeschlossenem `objectgrid`-Namespace erstellen. Sie können die Datei beliebig nennen. Sie muss jedoch den Blueprint-Namespace enthalten.

```
<?xml version="1.0" encoding="UTF-8"?>

<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
           xmlns:objectgrid="http://www.ibm.com/schema/objectgrid"
           xsi:schemaLocation="http://www.ibm.com/schema/objectgrid
                http://www.ibm.com/schema/objectgrid/objectgrid.xsd">
...
</blueprint>
```

2. XML-Definition für den eXtreme-Scale-Server mit den entsprechenden Servereigenschaften hinzufügen. Einzelheiten zu allen verfügbaren Konfigurationseigenschaften finden Sie in der Spring-XML-Deskriptordatei. Sehen Sie sich das folgende XML-Definitionsbeispiel an:

```
<objectgrid:server id="xsServer" tracespec="ObjectGridOSGi=all=enabled"
tracefile="logs/osgi/wxsserver/trace.log" jmxport="1199" listenerPort="2909">
<objectgrid:catalog host="catserver1.mycompany.com" port="2809" />
<objectgrid:catalog host="catserver2.mycompany.com" port="2809" />
</objectgrid:server>
```

3. XML-Definition für den eXtreme-Scale-Container mit der Referenz auf die Serverdefinition sowie die im Bundle integrierten ObjectGrid-XML-Deskriptor- und ObjectGrid-XML-Implementierungsdateien hinzufügen, z. B.:

```
<objectgrid:container id="container"
    objectgridxml="/META-INF/objectGrid.xml"
    deploymentxml="/META-INF/objectGridDeployment.xml"
    server="xsServer" />
```

4. Blueprint-XML-Datei im Container-Bundle speichern. Die Blueprint-XML-Datei muss im Verzeichnis `OSGI-INF/blueprint` gespeichert werden, damit der Blueprint-Container gefunden wird.

   Wenn Sie die Blueprint-XML-Datei in einem anderen Verzeichnis speichern möchten, müssen Sie den Manifestheader "Bundle-Blueprint" angeben, z. B.:

```
Bundle-Blueprint: OSGI-INF/blueprint.xml
```

5. Dateien in eine einzige Bundle-JAR-Datei packen. Sehen Sie sich das folgende Beispiel für eine Bundleverzeichnishierarchie an:

```
MyBundle.jar
    /META-INF/manifest.mf
    /META-INF/objectGrid.xml
    /META-INF/objectGridDeployment.xml
    /OSGI-INF/blueprint/blueprint.xml
```

Ergebnisse

Es wurde ein eXtreme-Scale-Container-Bundle erstellt, das in Eclipse Equinox installiert werden kann. Wenn das Container-Bundle gestartet wird, startet die Laufzeitumgebung des eXtreme-Scale-Servers automatisch den eXtreme-Scale-Singleton-Server mit den im Bundle definierten Parametern und startet einen Container-Server. Das Bundle kann gestoppt und gestartet werden, was dazu führt, dass der Container gestoppt bzw. gestartet wird. Der Server ist ein Singleton und wird nicht gestoppt, wenn das Bundle zum ersten Mal gestartet wird.

Szenario: JCA für die Verbindung transaktionsorientierter Anwendungen mit Clients von eXtreme Scale verwenden

Das folgende Szenario veranschaulicht die Herstellung von Verbindungen zwischen Clients und Anwendungen, die an Transaktionen teilnehmen.

Vorbereitende Schritte

Lesen Sie den Abschnitt Übersicht über die Transaktionsverarbeitung in Java-EE-Anwendungen, um mehr über die Transaktionsunterstützung zu erfahren.

Informationen zu diesem Vorgang

Java EE Connector Architecture (JCA) stellt Unterstützung für Clients bereit, die Java Transaction API (JTA) verwenden. Wenn Sie JTA verwenden, ist das Clientmanagement einfacher und wird über Java Platform, Enterprise Edition (Java EE) durchgeführt. Die JCA-Spezifikation unterstützt auch Ressourcenadapter, die Sie verwenden können, um Anwendungen mit Clients von eXtreme Scale zu verbinden. Ein Ressourcenadapter ist ein Softwaretreiber auf Systemebene, den eine Java-Anwendung für die Verbindung zu einem unternehmensweiten Informationssystem (EIS, Enterprise Information System, EIS) nutzt. Ein Ressourcenadapter integriert sich in einen Anwendungsserver und stellt die Konnektivität zwischen dem unternehmensweiten Informationssystem, Anwendungsserver und Unternehmensanwendung bereit. WebSphere eXtreme Scale stellt einen eigenen Ressourcenadapter bereit, den Sie ohne Konfiguration installieren können.

Wie in den früheren Versionen des Produkts können Sie Transaktionen verwenden, um eine einzelne Arbeitseinheit für das Datengrid zu verarbeiten. Wenn Sie diese Transaktionen festschreiben, können Sie mit der Unterstützung von JCA Ressourcen für diese Transaktion in einer einphasigen Festschreibung registrieren, was die folgende Vorteile hat:

- Vereinfachte Anwendungsentwicklung in eXtreme Scale Früher haben Entwickler die Transaktionen von eXtreme Scale mit Ressourcen wie Enterprise-Beans, Servlets und Web-Containern koordiniert. Da kein Rollbackmechanismus vorhanden war, besaßen die Entwickler keine einfache Methode für die Wiederherstellung nach einem Fehler.
- Es besteht eine nahtlose Integration mit WebSphere Application Server, die die Unterstützung des letzten Teilnehmers für die Registrierung bei globalen Transaktionen beinhaltet, sofern erforderlich.

Szenarioziele

Nach dem Ausführen dieses Szenarios wissen Sie, wie die folgenden Ziele erreicht werden:

- Verwendung der JTA-Unterstützung (Java Transaction API) für die Entwicklung von Anwendungskomponenten, die Transaktionen verwenden
- Verbinden der Anwendungen mit Clients von eXtreme Scale

Transaktionsverarbeitung in Java-EE-Anwendungen

WebSphere eXtreme Scale stellt einen eigenen Ressourcenadapter bereit, den Sie verwenden können, um Anwendungen mit dem Datengrid zu verbinden und lokale Transaktionen zu verarbeiten.

Mit der Unterstützung des Ressourcenadapters von eXtreme Scale können Java-EE-Anwendungen (Java Platform, Enterprise Edition) Clientverbindungen von eXtreme Scale suchen und lokale Transaktionen mithilfe lokaler Java-EE-Transaktionen oder mithilfe der APIs von eXtreme Scale abgrenzen. Wenn der Ressourcenadapter konfiguriert ist, können Sie die folgenden Aktionen mit Ihren Java-EE-Anwendungen ausführen:

- Verbindungsfactory des Ressourcenadapters von eXtreme Scale suchen oder in eine Java-EE-Anwendungskomponente injizieren
- Standardverbindungshandles zum Client von eXtreme Scale anfordern und unter Einhaltung der Java-EE-Konventionen in Anwendungskomponenten gemeinsam nutzen
- eXtreme-Scale-Transaktionen mithilfe der API javax.resource.cci.LocalTransaction oder der Schnittstelle com.ibm.websphere.objectgrid.Session abgrenzen
- Gesamte Client-API von eXtreme Scale verwenden, wie z. B. ObjectMap oder EntityManager

Die folgenden zusätzlichen Funktionen stehen mit WebSphere Application Server zur Verfügung:

- Registrieren von Verbindungen von eXtreme Scale bei einer globalen Transaktion als letzter Teilnehmer mit anderen Ressourcen für zweiphasige Festschreibung. Der Ressourcenadapter von eXtreme Scale bietet Unterstützung für lokale Transaktionen mit einer Ressource für einphasige Festschreibung. WebSphere Application Server ermöglicht Anwendungen, über die Unterstützung des letzten Teilnehmers eine einzige Ressource für einphasige Festschreibung bei einer globalen Transaktion zu registrieren.
- Automatische Installation des Ressourcenadapters bei Erweiterung des Profils
- Automatische Weitergabe des Sicherheitsprincipals

Zuständigkeiten des Administrators

Der Ressourcenadapter von eXtreme Scale wird im Java-EE-Anwendungsserver installiert oder mit der Anwendung eingebettet. Nach der Installation des Ressourcenadapters erstellt der Administrator eine oder mehrere Verbindungsfactory des Ressourcenadapters für jede Katalogservicedomäne und optional jede Datengridinstanz. Die Verbindungsfactory identifiziert die Eigenschaften, die erforderlich sind, um mit dem Datengrid zu kommunizieren.

Anwendungen referenzieren die Verbindungsfactory, woraufhin die Verbindung zum fernen Datengrid aufgebaut wird. Jede Verbindungsfactory hostet eine einzelne Clientverbindung von eXtreme Scale, die für alle Anwendungskomponenten wiederverwendet wird.

Wichtig: Da die Clientverbindung von eXtreme Scale einen nahen Cache enthalten kann, dürfen Anwendungen Verbindungen nicht gemeinsam nutzen. Es muss eine Verbindungsfactory für jede Anwendungsinstanz vorhanden sein, um Probleme mit der gemeinsamen Nutzung von Objekten durch mehrere Anwendungen zu vermeiden.

Die Verbindungsfactory hostet eine Clientverbindung von eXtreme Scale, die von allen referenzierenden Anwendungskomponenten gemeinsam genutzt wird. Sie können eine Managed Bean (MBean) verwenden, um auf Informationen zur Clientverbindung zuzugreifen oder um die Verbindung zurückzusetzen, wenn sie nicht mehr benötigt wird.

Zuständigkeiten des Anwendungsentwicklers

Ein Anwendungsentwickler erstellt Ressourcenreferenzen für verwaltete Verbindungsfactorys im Anwendungsimplementierungsdeskriptor oder mit Annotationen. Jede Ressourcenreferenz enthält eine lokale Referenz für die Verbindungsfactory von eXtreme Scale sowie den Geltungsbereich für die gemeinsame Nutzung von Ressourcen.

Wichtig: Die Aktivierung der gemeinsamen Nutzung von Ressourcen ist wichtig, weil sie die gemeinsame Nutzung lokaler Transaktionen durch mehrere Anwendungskomponenten ermöglicht.

Anwendungen können die Verbindungsfactory in die Java-EE-Anwendungskomponente injizieren, oder sie können sie mit JNDI suchen. Die Verbindungsfactory wird verwendet, um Verbindungshandles zur Clientverbindung von eXtreme Scale abzurufen. Die Clientverbindung von eXtreme Scale wird unabhängig von der Ressourcenadapterverbindung verwaltet und bei erstmaliger Verwendung aufgebaut und dann für alle nachfolgenden Verbindungen wiederverwendet.

Nachdem die Verbindung gefunden wurde, ruft die Anwendung eine Referenz auf die Sitzung von eXtreme Scale ab. Mit der Referenz auf die Sitzung von eXtreme Scale ist die Anwendung in der Lage, alle Client-APIs und Features von eXtreme Scale zu nutzen.

Sie können Transaktionen auf eine der folgenden Arten abgrenzen:

- Verwendung der Transaktionsdemarkationsmethoden von com.ibm.websphere.objectgrid.Session
- Verwendung der lokalen Transaktionen von javax.resource.cci.LocalTransaction
- Verwendung einer globalen Transaktion, wenn WebSphere Application Server mit aktivierter Unterstützung des letzten Teilnehmers verwendet wird. Wenn Sie diesen Ansatz für die Demarkation wählen, müssen Sie
  - eine anwendungsverwaltete globale Transaktion mit javax.transaction.UserTransaction verwenden,
  - eine containerverwaltete Transaktion verwenden.

Zuständigkeiten des Anwendungsimplementierers

Der Anwendungsimplementierer bindet die lokale Referenz der Verbindungsfactory des Ressourcenadapters, die der Anwendungsentwickler definiert, an die Verbindungsfactorys des Ressourcenadapters, die der Administrator definiert. Der Anwendungsimplementierer muss der Anwendung den richtigen Verbindungsfactorytyp und -geltungsbereich zuweisen und sicherstellen, dass die Verbindungsfactory nicht von mehreren Anwendungen gemeinsam genutzt wird, um die gemeinsame Nutzung von Java-Objekten zu verhindern. Der Anwendungsimplementierer ist auch für die Konfiguration und Zuordnung anderer Konfigurationsinformationen zuständig, die für alle Verbindungsfactorys gleichermaßen gelten.

Ressourcenadapter von eXtreme Scale installieren

Der Ressourcenadapter von WebSphere eXtreme Scale ist mit Java Connector Architecture (JCA) 1.5 kompatibel und kann in Java 2 Platform, Enterprise Edition (J2EE) 1.5 1.6 oder höher oder in einem Anwendungsserver wie WebSphere Application Server installiert werden.

Vorbereitende Schritte

Der Ressourcenadapter ist in der RAR-Datei (Ressourcenadapterarchiv) `wxsra.rar` enthalten, die in allen Installationen von eXtreme Scale verfügbar ist. Die RAR-Datei befindet sich in den folgenden Verzeichnissen:

- Für Installationen in WebSphere Application Server: *WXS-Installationsstammverzeichnis*`/optionalLibraries/ObjectGrid`
- Für eigenständige Installationen: *WXS-Installationsstammverzeichnis*`/ObjectGrid/lib`

Der Ressourcenadapter ist mit der Laufzeitumgebung von eXtreme Scale gekoppelt. Er setzt die JAR-Dateien der Laufzeitumgebung von eXtreme Scale im richtigen Klassenpfad voraus. Im Allgemeinen können Sie ein Upgrade der Laufzeitumgebung von eXtreme Scale durchführen, ohne den Ressourcenadapter zu aktualisieren. Bei einem Upgrade der Laufzeitumgebung von eXtreme Scale wird auch ein Upgrade der Laufzeitumgebung des Ressourcenadapters durchgeführt. Der Ressourcenadapter unterstützt Version 8.5 und bis zu zwei neuere Versionen der Laufzeitumgebung von eXtreme Scale. Neuere Versionen des Ressourcenadapters können neuere Versionen der Laufzeitumgebung von eXtreme Scale erfordern, sobald diese verfügbar sind

Die Datei `wxsra.rar` erfordert eine der JAR-Dateien für die Clientlaufzeitumgebung von eXtreme Scale, um verwendet werden zu können. Einzelheiten zur geeigneten JAR-Datei für die Clientlaufzeitumgebung finden Sie in den Abschnitten Laufzeitdateien für eine eigenständige Installation von WebSphere eXtreme Scale und Laufzeitdateien für eine integrierte Installation von WebSphere eXtreme Scale in WebSphere Application Server, die Details zu den verfügbaren JAR-Dateien für die Laufzeitumgebungen enthalten.

Informationen zu diesem Vorgang

Sie können den Ressourcenadapter von eXtreme Scale über verschiedene Optionen installieren, die flexible Implementierungsszenarien unterstützen. Der Ressourcenadapter kann mit der Java-EE-Anwendung (Java Platform, Enterprise Edition) eingebettet oder als eigenständige RAR-Datei installiert werden, die von mehreren Anwendungen gemeinsam genutzt wird.

Das Einbetten des Ressourcenadapters mit der Anwendung vereinfacht die Implementierung, weil Verbindungsfactorys nur im Geltungsbereich der Anwendung erstellt werden und nicht von mehreren Anwendungen gemeinsam genutzt werden können. Mit dem in die Anwendung eingebetteten Ressourcenadapter können Sie auch die Cacheobjekte und die ObjectGrid-Client-Plug-in-Klassen in die Anwendung einbetten. Das Einbetten des Ressourcenadapters schützt die Anwendung auch vor einer versehentlichen gemeinsamen Nutzung von Cacheobjekten durch mehrere Anwendungen, was zu Ausnahmen des Typs java.lang.ClassCastException führen kann.

Wenn Sie die Datei `wxsra.rar` als eigenständigen Ressourcenadapter installieren, können Verbindungsfactorys des Ressourcenmanagers auf Knotenebene erstellt werden. Diese Option ist in den folgenden Situationen hilfreich:

- Es ist nicht zweckmäßig, die Datei `wxsra.rar` in die Anwendung einzubetten.
- Die Version von eXtreme Scale ist zur Buildzeit nicht bekannt.
- Sie möchten, dass eine Clientverbindung von eXtreme Scale von mehrere Anwendungen gemeinsam genutzt wird.

Wichtig: In mehreren Versionen von WebSphere Application Server bis hin zu Version 8.0.2 kann der Ressourcenadapter von eXtreme Scale nicht gleichzeitig in einer Anwendungs-EAR-Datei und in einem eigenständigen Server installiert werden. Deshalb tritt bei der Verwendung der EAR-Datei, in der auch die RAR-Datei installiert ist, in der Anwendung eine Ausnahme ein, wie z. B. `ClassCastException: com.ibm.websphere.xs.ra.XSConnectionFactory incompatible with com.ibm.websphere.xs.ra.XSConnectionFactory`. Der folgende Beispielnachrichten- und Beispielaufrufstack von WebSphere Application Server zu diesem Fehler wird angezeigt, wenn diese Ausnahme in einem Servlet auftritt:

```
SRVE0068E: Es wurde eine nicht abgefangene Ausnahme in einer der
Servicemethoden des Servlets [ClientServlet] in der Anwendung [JTASampleClientEAR] erstellt.
Erstellte Ausnahme: [java.lang.ClassCastException:
com.ibm.websphere.xs.ra.XSConnectionFactory incompatible with com.ibm.websphere.xs.ra.XSConnectionFactory
at com.ibm.websphere.xs.sample.jtasample.WXSClientServlet.connectClient(WXSClientServlet.java:484)
at com.ibm.websphere.xs.sample.jtasample.WXSClientServlet.doGet(WXSClientServlet.java:200)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:575)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:668)
at com.ibm.ws.webcontainer.servlet.ServletWrapper.service(ServletWrapper.java:1214)
at com.ibm.ws.webcontainer.servlet.ServletWrapper.handleRequest(ServletWrapper.java:774)
at com.ibm.ws.webcontainer.servlet.ServletWrapper.handleRequest(ServletWrapper.java:456)
```

Vorgehensweise

- **Installieren Sie einen eingebetteten Ressourcenadapter von eXtreme Scale.** Wenn die Datei `wxsra.rar` in die EAR-Datei der Anwendung eingebettet ist, muss der Ressourcenadapter Zugriff auf die Laufzeitbibliotheken von eXtreme Scale haben.

  Für Anwendungen, die in WebSphere Application Server ausgeführt werden, sind die folgenden Optionen und Folgeaktionen verfügbar:

Option	Bezeichnung
Wenn eXtreme Scale mit dem Knoten von WebSphere Application Server integriert ist	Die Laufzeitbibliotheksdateien sind bereits im Systemklassenpfad verfügbar, und es ist keine weitere Aktion erforderlich.
Wenn eXtreme Scale nicht mit dem Knoten von WebSphere Application Server integriert ist	Sie müssen die Datei `wsogclient.jar` in den Klassenpfad von der Datei `wxsra.rar` einfügen.

  Für Anwendungen, die nicht in WebSphere Application Server ausgeführt werden, muss die Laufzeitbibliotheksdatei des Clients, `ogclient.jar`, oder die Laufzeitbibliotheksdatei des Servers, `objectgrid.jar`, im Klassenpfad der RAR-Datei enthalten sein.

- **Installieren Sie einen eigenständigen Ressourcenadapter von eXtreme Scale.** Wenn Sie die Datei `wxsra.rar` als eigenständigen Ressourcenadapter installieren, muss dieser Zugriff auf die Laufzeitbibliotheken von eXtreme Scale haben.

  Für Anwendungen, die in WebSphere Application Server ausgeführt werden, sind die folgenden Optionen und Folgeaktionen verfügbar:

Option	Bezeichnung
Wenn eXtreme Scale mit dem Knoten von WebSphere Application Server integriert ist	Die Laufzeitbibliotheksdateien sind bereits im Systemklassenpfad verfügbar, und es ist keine weitere Aktion erforderlich.
Wenn eXtreme Scale nicht mit dem Knoten von WebSphere Application Server integriert ist	Sie müssen die Datei `wsogclient.jar` in den Klassenpfad von der Datei `wxsra.rar` einfügen.

Für Anwendungen, die nicht in WebSphere Application Server ausgeführt werden, muss die Laufzeitbibliotheksdatei des Clients, `ogclient.jar`, oder die Laufzeitbibliotheksdatei des Servers, `objectgrid.jar`, im Klassenpfad der RAR-Datei enthalten sein.

1. Erteilen Sie dem Ressourcenadapter Zugriff auf alle gemeinsam genutzten Klassen. Alle ObjectGrid-Plug-in-Klassen und -Anwendungen, die diese Klassen verwenden, müssen ein gemeinsames Klassenladeprogramm verwenden. Da der Ressourcenadapter von mehreren Anwendungen gemeinsam genutzt wird, müssen alle Klassen über dasselbe Klassenladeprogramm zugänglich sein. Sie können diesen Zugriff erstellen, indem Sie eine gemeinsam genutzte Bibliothek für alle Anwendungen verwenden, die mit dem Ressourcenadapter interagieren.

Nächste Schritte

Nachdem Sie nun den Ressourcenadapter von eXtreme Scale installiert haben, können Sie Verbindungsfactorys konfigurieren, damit Ihre Java-EE-Anwendungen eine Verbindung zu einem fernen Datengrid von eXtreme Scale herstellen können.

Verbindungsfactorys von eXtreme Scale konfigurieren

Java

Eine eXtreme-Scale-Verbindungsfactory ermöglicht Java-EE-Anwendungen, eine Verbindung zu einem fernen Datengrid von WebSphere eXtreme Scale herzustellen. Verwenden Sie angepasste Eigenschaften, um Ressourcenadapter zu konfigurieren.

Vorbereitende Schritte

Bevor Sie die Verbindungsfactorys erstellen, müssen Sie den Ressourcenadapter installieren.

Informationen zu diesem Vorgang

Nach der Installation des Ressourcenadapters können Sie eine oder mehrere Verbindungsfactorys des Ressourcenadapters erstellen, die Clientverbindungen von eXtreme Scale zu fernen Datengrids darstellen. Führen Sie die folgenden Schritte aus, um eine Verbindungsfactory des Ressourcenadapters zu konfigurieren und in einer Anwendung zu verwenden.

Sie können eine eXtreme-Scale-Verbindungsfactory auf Knotenebene für eigenständige Ressourcenadapter oder in der Anwendung für eingebettete Ressourcenadapter erstellen. Informationen zum Erstellen von Verbindungsfactorys in WebSphere Application Server finden Sie in den zugehörigen Abschnitten.

Vorgehensweise

1. Erstellen Sie in der erstellen von WebSphere Application Server eine Verbindungsfactory von eXtreme Scale, die eine Clientverbindung von eXtreme Scale darstellt. Weitere Informationen finden Sie unter "Java-EE-Connector-Verbindungfactorys in der Administrationskonsole konfigurieren". Nachdem Sie die Eigenschaften für die Verbindungsfactory in der Anzeige "Allgemeine Eigenschaften" angegeben haben, müssen Sie für den Link "Angepasste Eigenschaften" auf den Link **Anwenden** klicken, damit er aktiviert wird.

2. Klicken Sie in der Administrationskonsole auf **Angepasste Eigenschaften**. Setzen Sie die folgenden angepassten Eigenschaften, um die Clientverbindung zum fernen Datengrid zu konfigurieren.

Tabelle 9. Angepasste Eigenschaften für die Konfiguration von Verbindungsfactorys

Eigenschaftsname	Typ	
ConnectionName	String	(Optional) Der Name der Clientverbindung von eXtreme Scale. Mithilfe der Eigenschaft "ConnectionName" kann die Verbindung bei Bereitstellung als Managed Bean identifiziert werden. Diese Eigenschaft ist optional. Wenn Sie die Eigenschaft nicht angeben, wird ConnectionName nicht definiert.
CatalogServiceEndpoints	String	(Optional) Die Endpunkte der Katalogservicedomäne im folgenden Format: `<Host>:<Port>[,<Host><Port>]`. Weitere Informationen finden Sie im Abschnitt Einstellungen der Katalogservicedomäne. Diese Eigenschaft ist erforderlich, wenn die Katalogservicedomäne nicht definiert ist.
CatalogServiceDomain	String	(Optional) Der Katalogservicedomänenname, der in WebSphere Application Server definiert ist. Weitere Informationen hierzu finden Sie im Abschnitt Katalogserver und Katalogservicedomänen konfigurieren. Diese Eigenschaft ist erforderlich, wenn die Eigenschaft "CatalogServiceEndpoints" nicht definiert ist.
ObjectGridName	String	(Optional) Der Name des Datengrids, zu dem diese Verbindungsfactory eine Verbindung herstellt. Wenn Sie diese Eigenschaft nicht angeben, muss die Anwendung den Namen beim Anfordern der Verbindung von der Verbindungsfactory angeben.
ObjectGridURL	String	(Optional) Der URL der XML-Überschreibungsdatei für das Clientdatengrid. Diese Eigenschaft ist nicht gültig, wenn ObjectGridResource ebenfalls angegeben ist. Weitere Informationen finden Sie unter Clients konfigurieren.
ObjectGridResource	String	Der Ressourcenpfad der XML-Überschreibungsdatei für das Clientdatengrid. Diese Eigenschaft ist optional und ungültig, wenn ObjectGridURL ebenfalls angegeben ist. Weitere Informationen finden Sie unter Clients konfigurieren.
ClientPropertiesURL	String	(Optional) Der URL der Clienteigenschaftendatei. Diese Eigenschaft ist nicht gültig, wenn ClientPropertiesResource ebenfalls angegeben ist. Weitere Informationen finden Sie unter Clienteigenschaftendatei.
ClientPropertiesResource	String	(Optional) Der Ressourcenpfad der Clienteigenschaftendatei. Diese Eigenschaft ist nicht gültig, wenn ClientPropertiesURL ebenfalls angegeben ist. Weitere Informationen finden Sie unter Clienteigenschaftendatei.

WebSphere Application Server lässt auch andere Konfigurationsoptionen für die Anpassung von Verbindungspools und die Verwaltung der Sicherheit zu. Unter "Zugehörige Informationen" finden Sie Links zu Abschnitten im Information Center von WebSphere Application Server.

Nächste Schritte

Erstellen Sie eine Referenz auf die eXtreme-Scale-Verbindungsfactory in der Anwendung. Weitere Informationen finden Sie unter „Anwendungen für das Herstellen von Verbindungen zu eXtreme Scale konfigurieren” auf Seite 263.

Eclipse-Umgebungen für die Verwendung von eXtreme-Scale-Verbindungsfactorys konfigurieren

Java

Der Ressourcenadapter von eXtreme Scale enthält angepasste Verbindungsfactorys. Wenn Sie diese Schnittstellen in Ihren Java-EE-Anwendungen (eXtreme Scale Java Platform, Enterprise Edition) verwenden möchten, müssen Sie die Datei `wxsra.rar` in Ihren Arbeitsbereich importieren und mit Ihrem Anwendungsprojekt verknüpfen.

Vorbereitende Schritte

- Sie müssen Rational Application Developer Version 7 oder höher oder Eclipse Java EE IDE for Web Developers Version 1.4 oder höher installieren.
- Es muss eine Serverlaufzeitumgebung konfiguriert werden.

Vorgehensweise

1. Importieren Sie die Datei `wxsra.rar` in Ihr Projekt, indem Sie **Datei** > **Importieren** auswählen. Das Importfenser wird angezeigt.
2. Wählen Sie **Java EE** > **RAR-Datei** aus. Das Fenster "Connector importieren" erscheint.
3. Zum Angeben der Connectordatei klicken Sie auf **Durchsuchen**, um die Datei `wxsra.rar` zu suchen. Die Datei `wxsra.rar` wird installiert, wenn Sie einen Ressourcenadapter installieren. Sie finden die RAR-Datei (Ressourcenadapterarchiv) an der folgenden Position:
   - Für Installationen in WebSphere Application Server: *WXS-Installationsstammverzeichnis*`/optionalLibraries/ObjectGrid`
   - Für eigenständige Installationen: *WXS-Installationsstammverzeichnis*`/ObjectGrid/lib`
4. Erstellen Sie im Feld **Connectorprojekt** einen Namen für das neue Connectorprojekt. Sie können den Standardnamen `wxsra` verwenden.
5. Wählen Sie eine Ziellaufzeitumgebung aus, die auf eine Java-EE-Serverlaufzeitumgebung verweist.
6. Wählen Sie optional **Projekt einer EAR hinzufügen** aus, um die RAR-Datei in ein vorhandenes EAR-Projekt einzubetten.

Ergebnisse

Die RAR-Datei wird jetzt in den Eclipse-Arbeitsbereich importiert.

Nächste Schritte

Sie können das RAR-Projekt über andere Java-EE-Projekte wie folgt referenzieren:

1. Klicken Sie mit der rechten Maustaste auf das Projekt, und klicken Sie auf **Eigenschaften**.
2. Wählen Sie **Java-Erstellungspfad** aus.
3. Wählen Sie das Register "Projekte" aus.
4. Klicken Sie auf **Hinzufügen**.
5. Wählen Sie das Connectorprojekt **wxsra** aus, und klicken Sie auf **OK**.
6. Klicken Sie erneut auf **OK**, um das Fenster "Eigenschaften" zu schließen.

Die Ressourcenadapterklassen von eXtreme Scale sind jetzt im Klassenpfad enthalten. Informationen zum Installieren der Laufzeit-JAR-Dateien des Produkts über die Eclipse-Konsole finden Sie unter Eigenständige Entwicklungsumgebung in Eclipse einrichten.

Anwendungen für das Herstellen von Verbindungen zu eXtreme Scale konfigurieren

Anwendungen verwenden eine Verbindungsfactory von eXtreme Scale, um Verbindungshandles zu einer Clientverbindung von eXtreme Scale zu erstellen. Mit dieser Task können Sie Referenzen auf die Verbindungsfactorys des Ressourcenadapters konfigurieren.

Vorbereitende Schritte

Erstellen Sie eine Java-EE-Anwendungskomponente (Java Platform, Enterprise Edition), z. B. einen EJB-Container (Enterprise JavaBeans) oder ein Servlet.

Vorgehensweise

Erstellen Sie eine Ressourcenreferenz javax.resource.cci.ConnectionFactory in der Anwendungskomponente. Ressourcenreferenzen werden vom Anwendungsprovider im Deploymentdeskriptor deklariert. Die Verbindungsfactory stellt eine Clientverbindung von eXtreme Scale dar, die verwendet werden kann, um mit einem oder mehreren benannten Datengrids zu kommunizieren, die in der Katalogservicedomäne verfügbar sind.

J2C-Clientverbindungen sichern

Verwenden Sie die J2C-Architektur (Java 2 Connector), um Verbindungen zwischen Clients von WebSphere eXtreme Scale und Ihren Anwendungen zu sichern.

Informationen zu diesem Vorgang

Anwendungen referenzieren die Verbindungsfactory, woraufhin die Verbindung zum fernen Datengrid aufgebaut wird. Jede Verbindungsfactory hostet eine einzelne Clientverbindung von eXtreme Scale, die für alle Anwendungskomponenten wiederverwendet wird.

Wichtig: Da die Clientverbindung von eXtreme Scale einen nahen Cache enthalten kann, ist es wichtig, dass Anwendungen eine Verbindung nicht gemeinsam nutzen. Es muss eine Verbindungsfactory für jede Anwendungsinstanz vorhanden sein, um Probleme mit der gemeinsamen Nutzung von Objekten durch mehrere Anwendungen zu vermeiden.

Sie können den Berechtigungsnachweisgenerator mit der API oder in der Clienteigenschaftendatei festlegen. In der Clienteigenschaftendatei werden die Eigenschaften "securityEnabled" und "credentialGenerator" verwendet. Das folgende Codebeispiel ist aus Gründen der besseren Übersichtlichkeit auf mehrere Zeilen verteilt:

```
securityEnabled=true
credentialGeneratorClass=com.ibm.websphere.objectgrid.security.plugins.builtins.
     UserPasswordCredentialGenerator
credentialGeneratorProps=operator XXXXXX
```

Der Berechtigungsnachweisgenerator und die Berechtigungsnachweise in der Clienteigenschaftendatei werden für die Verbindungsoperation von eXtreme Scale und die J2C-Standardberechtigungsnachweise verwendet. Deshalb werden die Berechtigungsnachweise, die mit der API angegeben werden, beim Herstellen der J2C-Verbindung für die J2C-Verbindung verwendet. Wenn beim Herstellen der J2C-Verbindung jedoch keine Berechtigungsnachweise angegeben werden, wird der Berechtigungsnachweisgenerator in der Clienteigenschaftendatei verwendet.

Vorgehensweise

1. Konfigurieren Sie den sicheren Zugriff, wobei die J2C-Verbindung den Client von eXtreme Scale darstellt. Verwenden Sie die Verbindungsfactoryeigenschaft "ClientPropertiesResource" oder "ClientPropertiesURL", um die Clientauthentifizierung zu konfigurieren.

   Wenn Sie WebSphere eXtreme Scale mit WebSphere Application Server verwenden, geben Sie die Clienteigenschaften in der Konfiguration der Katalogser-

vicedomäne an. Wenn die Verbindungsfactory auf die Domäne verweist, wird diese Konfiguration automatisch verwendet.

2. Konfigurieren Sie die Clientsicherheitseigenschaften so, dass die Verbindungsfactory verwendet wird, die auf das entsprechenden Berechtigungsnachweisgeneratorobjekt für eXtreme Scale verweist. Diese Eigenschaften sind auch mit der Serversicherheit von eXtreme Scale kompatibel. Verwenden Sie beispielsweise den Berechtigungsnachweisgenerator "WSTokenCredentialGenerator" für WebSphere-Berechtigungsnachweise, wenn eXtreme Scale mit WebSphere Application Server installiert wird. Verwenden Sie alternativ den Berechtigungsnachweisgenerator "UserPasswordCredentialGenerator", wenn Sie eXtreme Scale in einer eigenständigen Umgebung ausführen. Im folgenden Beispiel werden die Berechtigungsnachweise programmgesteuert mit dem API-Aufruf und nicht über die Konfiguration in den Clienteigenschaften übergeben:

   ```
   XSConnectionSpec spec = new XSConnectionSpec();
   spec.setCredentialGenerator(new UserPasswordCredentialGenerator("operator", "xxxxxx"));
   Connection conn = connectionFactory.getConnection(spec);
   ```

3. (Optional) Inaktivieren Sie ggf. den nahen Cache.

   Alle J2C-Verbindungen aus einer Verbindungsfactory verwenden denselben nahen Cache. Grideintragsberechtigungen und Mapberechtigungen werden im Server, aber nicht im nahen Cache validiert. Wenn eine Anwendung mehrere Berechtigungsnachweise zum Erstellen von J2C-Verbindungen verwendet und die Konfiguration bestimmte Berechtigungen für Grideinträge und Maps für diese Berechtigungsnachweise verwendet, inaktivieren Sie den nahen Cache. Inaktivieren Sie den nahen Cache mit der Verbindungsfactoryeigenschaft "ObjectGridResource" oder "ObjectGridURL". Weitere Informationen zum Inaktivieren des nahen Caches finden Sie unter Nahen Cache konfigurieren.

4. (Optional) Legen Sie die ggf. die Sicherheitsrichtlinieneinstellungen fest.

   Wenn die J2EE-Anwendung die Konfiguration der eingebetteten RAR-Datei von eXtreme Scale enthält, müssen Sie möglicherweise weitere Sicherheitsrichtlinieneinstellungen in der Sicherheitsrichtliniendatei für die Anwendung festlegen. Die folgenden Richtlinien sind beispielsweise erforderlich:

   ```
   permission com.ibm.websphere.security.WebSphereRuntimePermission "accessRuntimeClasses";
   permission java.lang.RuntimePermission "accessDeclaredMembers";
   permission javax.management.MBeanTrustPermission "register";
   permission java.lang.RuntimePermission "getClassLoader";
   ```

   Außerdem erfordern alle Eigenschaften- und Ressourcendateien, die von Verbindungsfactorys verwendet werden, Datei- oder andere Berechtigungen, wie z. B. `permission java.io.FilePermission "filePath";`. Die Richtliniendatei für WebSphere Application Server ist `META-INF/was.policy` und befindet sich in der J2EE-EAR-Datei.

Ergebnisse

Die Clientsicherheitseigenschaften, die Sie in der Katalogservicedomäne konfiguriert haben, werden als Standardwerte verwendet. Die Werte, die Sie angeben, überschreiben alle Eigenschaften, die in den Dateien `client.properties` definiert sind.

Nächste Schritte

Verwenden Sie die Datenzugriffs-APIs von eXtreme Scale, um Clientkomponenten zu entwickeln, die Transaktionen verwenden sollen.

eXtreme-Scale-Clientkomponenten für die Verwendung von Transaktionen entwickeln

Java

Der Ressourcenadapter von WebSphere eXtreme Scale stellt Unterstützung für das Management von Clientverbindungen und für lokale Transaktionen bereit. Mit dieser Unterstützung können Java-EE-Anwendungen (Java Platform, Enterprise Edition) Clientanwendungen von eXtreme Scale suchen und lokale Transaktionen mit lokalen Java-EE-Transaktionen oder mit den APIs von eXtreme Scale abgrenzen.

Vorbereitende Schritte

Erstellen Sie eine Ressourcenreferenz auf die Verbindungsfactory von eXtreme Scale.

Informationen zu diesem Vorgang

Es gibt mehrere Optionen für die Arbeit mit den Datenzugriffs-APIs von eXtreme Scale. In allen Fällen muss die Verbindungsfactory von eXtreme Scale in die Anwendungskomponente injiziert oder mit JNDI (Java Naming Directory Interface) gesucht werden. Nachdem die Verbindungsfactory gefunden wurde, können Sie Transaktionen abgrenzen und Verbindungen für den Zugriff auf die APIs von eXtreme Scale erstellen.

Die javax.resource.cci.ConnectionFactory-Instanz kann optional in eine com.ibm-.websphere.xs.ra.XSConnectionFactory umgesetzt werden, die weitere Optionen für den Abruf von Verbindungshandles bereitstellt. Die abgerufenen Verbindungshandles muss in die Schnittstelle "com.ibm.websphere.xs.ra.XSConnection" umgesetzt werden, die die Methode "getSession" bereitstellt. Die Methode "getSession" gibt ein com.ibm.websphere.objectgrid.Session-Objekthandle zurück, das Anwendungen die Verwendung alle Datenzugriffs-APIs von eXtreme Scale wie ObjectMap und EntityManager ermöglicht.

Das Sitzungshandle und alle abgeleiteten Objekte sind für die Lebensdauer des XSConnection-Handles gültig.

Die folgenden Prozeduren können verwendet werden, um Transaktionen von eXtreme Scale abzugrenzen. Die Prozeduren können nicht gemischt verwendet werden. Es ist beispielsweise nicht möglich, globale Transaktionsdemarkation und lokale Transaktionsdemarkation im Kontext derselben Anwendungskomponente gemischt zu verwenden.

Vorgehensweise

- Lokale Transaktionen mit automatischer Festschreibung verwenden. Verwenden Sie die folgenden Schritte, um Datenzugriffsoperationen mit automatischer Festschreibung oder Operationen zu verwenden, die aktive Transaktionen nicht unterstützen:
  1. Rufen Sie eine com.ibm.websphere.xs.ra.XSConnection-Verbindung außerhalb des Kontextes einer globalen Transaktion ab.
  2. Rufen Sie die com.ibm.websphere.objectgrid.Session-Sitzung für die Interaktion mit dem Datengrid ab, und verwenden Sie sie.
  3. Rufen Sie eine Datenzugriffsoperation auf, die Transaktionen mit automatischer Festschreibung unterstützt.
  4. Schließen Sie die Verbindung.

- Verwenden Sie eine ObjectGrid-Sitzung, um eine lokale Transaktion abzugrenzen. Verwenden Sie die folgenden Schritte, um eine ObjectGrid-Transaktion mit dem Session-Objekt abzugrenzen:
  1. Rufen Sie eine com.ibm.websphere.xs.ra.XSConnection-Verbindung ab.
  2. Rufen Sie die com.ibm.websphere.objectgrid.Session-Sitzung ab.
  3. Verwenden Sie die Methode "Session.begin()", um die Transaktion zu starten.
  4. Verwenden Sie die Sitzung, um mit dem Datengrid zu interagieren.
  5. Verwenden Sie die Methode "Session.commit()" oder "rollback()", um die Transaktion zu beenden.
  6. Schließen Sie die Verbindung.
- Verwenden Sie eine javax.resource.cci.LocalTransction-Transaktion, um eine lokale Transaktion abzugrenzen. Verwenden Sie die folgenden Schritte, um eine ObjectGrid-Transaktion mit der Schnittstelle javax.resource.cci.LocalTransaction abzugrenzen:
  1. Rufen Sie eine com.ibm.websphere.xs.ra.XSConnection-Verbindung ab.
  2. Rufen Sie die javax.resource.cci.LocalTransaction-Transaktion mit der Methode XSConnection.getLocalTransaction() ab.
  3. Verwenden Sie die Methode "LocalTransaction.begin()", um die Transaktion zu starten.
  4. Rufen Sie die com.ibm.websphere.objectgrid.Session-Sitzung für die Interaktion mit dem Datengrid ab, und verwenden Sie sie.
  5. Verwenden Sie die Methode "LocalTransaction.commit()" oder "rollback()", um die Transaktion zu beenden.
  6. Schließen Sie die Verbindung.
- Registrieren Sie die Verbindung in einer globalen Transaktion. Diese Prozedur gilt auch für containerverwaltete Transaktionen:
  1. Starten Sie die globale Transaktion über die Schnittstelle "javax.transaction.UserTransaction" oder mit einer containerverwalteten Transaktion.
  2. Rufen Sie eine com.ibm.websphere.xs.ra.XSConnection-Verbindung ab.
  3. Rufen Sie eine com.ibm.websphere.objectgrid.Session-Sitzung ab, und verwenden Sie sie.
  4. Schließen Sie die Verbindung.
  5. Schreiben Sie die globale Transaktion fest, oder machen Sie sie rückgängig.
- **8.6+** Konfigurieren Sie eine Verbindung, um in einer einzigen Transaktion auf mehrere Partitionen zu schreiben. Verwenden Sie die folgenden Schritte, um eine ObjectGrid-Transaktion mit dem Session-Objekt abzugrenzen:
  1. Erstellen Sie ein neues com.ibm.websphere.xs.ra.XSConnectionSpec-Objekt.
  2. Rufen Sie die Methode "XSConnectionSpec" und die Methode "setMultiPartitionSupportEnabled" mit dem Argument `true` auf.
  3. Rufen Sie die com.ibm.websphere.xs.ra.XSConnection-Verbindung ab, um XSConnectionSpec an die Methode "ConnectionFactory.getConnection" zu übergeben.
  4. Rufen Sie eine com.ibm.websphere.objectgrid.Session-Sitzung ab, und verwenden Sie sie.

Beispiel

Sehen Sie sich das folgende Codebeispiel an, das die vorherigen Schritte für die Demarkation der Transaktionen von eXtreme Scale veranschaulicht.

```
// (C) Copyright IBM Corp. 2001, 2012.
// All Rights Reserved. Licensed Materials - Property of IBM.
package com.ibm.ws.xs.ra.test.ee;

import javax.naming.InitialContext;
import javax.resource.cci.Connection;
import javax.resource.cci.ConnectionFactory;
import javax.resource.cci.LocalTransaction;
import javax.transaction.Status;
import javax.transaction.UserTransaction;

import junit.framework.TestCase;

import com.ibm.websphere.objectgrid.ObjectMap;
import com.ibm.websphere.objectgrid.Session;
import com.ibm.websphere.xs.ra.XSConnection;

/**
* Dieses Beispiel muss in einem J2EE-Kontext in Ihrem Anwendungsserver
* ausgeführt werden, z. B. über das JUnitEE-Framework-Servlet.
*
* Der Code in diesen Testmethoden befindet sich gewöhnlich in Ihrem
* eigenen Servlet, Ihrer eigenen EJB oder einer anderen Webkomponente.
*
* Das Beispiel ist von einer konfigurierten Verbindungsfactory von WebSphere eXtreme Scale
* abhängig, die unter dem JNDI-Namen "eis/embedded/wxscf" registriert wurde und
* eine Verbindung zu einem Grid definiert, das eine Map mit dem Namen "Map1" enthält.
*
* Das Beispiel führt ein Direkt-Lookup des JNDI-Namens durch und erfordert
* Keine Ressourceninjektion.
*/
public class DocSampleTests extends TestCase {
    public final static String CF_JNDI_NAME = "eis/embedded/wxscf";
    public final static String MAP_NAME = "Map1";

    Long                key = null;
    Long                value = null;
    InitialContext      ctx = null;
    ConnectionFactory   cf = null;

    public DocSampleTests() {
    }
    public DocSampleTests(String name) {
        super(name);
    }
    protected void setUp() throws Exception {
        ctx = new InitialContext();
        cf = (ConnectionFactory)ctx.lookup(CF_JNDI_NAME);
        key = System.nanoTime();
        value = System.nanoTime();
    }
    /**
     * Dieses Beispiel wird ausgeführt, wenn kein globaler Transaktionskontext
     * vorhanden ist, und verwendet automatische Festschreibung.
     */
    public void testLocalAutocommit() throws Exception {
        Connection conn = cf.getConnection();
        try  {
            Session    session = ((XSConnection)conn).getSession();
            ObjectMap  map = session.getMap(MAP_NAME);
            map.insert(key, value); // Or various data access operations
        }
        finally {
            conn.close();
        }
    }

    /**
     * Dieses Beispiel wird ausgeführt, wenn kein globaler Transaktionskontext
     * vorhanden ist, und grenzt die Transaktion mit session.begin()/session.commit() ab.
     */
    public void testLocalSessionTransaction() throws Exception {
        Session    session = null;
        Connection conn = cf.getConnection();
        try  {
            session = ((XSConnection)conn).getSession();
            session.begin();
            ObjectMap  map = session.getMap(MAP_NAME);
            map.insert(key, value); // Or various data access operations
```

```
        session.commit();
    }
    finally {
        if (session != null && session.isTransactionActive()) {
            try { session.rollback(); }
            catch (Exception e) { e.printStackTrace(); }
        }
        conn.close();
    }
}

/**
 * Dieses Beispiel verwendet die Schnittstelle LocalTransaction für die
 * Demarkation von Transaktionen.
 */
public void testLocalTranTransaction() throws Exception {
    LocalTransaction tx = null;
    Connection conn = cf.getConnection();
    try  {
        tx = conn.getLocalTransaction();
        tx.begin();
        Session    session = ((XSConnection)conn).getSession();
        ObjectMap  map = session.getMap(MAP_NAME);
        map.insert(key, value); // Or various data access operations
        tx.commit(); tx = null;
    }
    finally {
        if (tx != null) {
            try { tx.rollback(); }
            catch (Exception e) { e.printStackTrace(); }
        }
        conn.close();
    }
}

/**
 * Dieses Beispiel ist von einer extern verwalteten Transaktion abhängig,
 * die gewöhnlich in einer EJB enthalten ist, deren Transaktionsattribute
 * auf REQUIRED oder REQUIRES_NEW gesetzt sind.
 * ANMERKUNG: Wenn keine globale Transaktion aktiv ist, wird dieses Beispiel
 *            im Modus für automatische Festschreibung ausgeführt, weil nicht
 *            geprüft wird, ob eine Transaktion vorhanden ist.
 */
public void testGlobalTransactionContainerManaged() throws Exception {
    Connection      conn = cf.getConnection();
    try  {
        Session    session = ((XSConnection)conn).getSession();
        ObjectMap  map = session.getMap(MAP_NAME);
        map.insert(key, value); // Or various data access operations
    }
    catch (Throwable t) {
        t.printStackTrace();
        UserTransaction tx = (UserTransaction)ctx.lookup("java:comp/UserTransaction");
        if (tx.getStatus() != Status.STATUS_NO_TRANSACTION) {
            tx.setRollbackOnly();
        }
    }
    finally {
        conn.close();
    }
}

/**
 * Dieses Beispiel veranschaulicht das Starten einer neuen globalen Transaktion
 * mit der Schnittstelle UserTransaction. Gewöhnlich startet der Container die
 * globale Transaktion (z. B. in einer EJB, deren Transaktionsattribut auf
 * REQUIRES_NEW gesetzt ist), aber auch dieses Beispiel startet die globale Transaktion
 * mit der API UserTransaction, falls sie momentan nicht aktiv ist.
 */
public void testGlobalTransactionTestManaged() throws Exception {
    boolean         started = false;
    UserTransaction tx = (UserTransaction)ctx.lookup("java:comp/UserTransaction");
    if (tx.getStatus() == Status.STATUS_NO_TRANSACTION) {
        tx.begin();
        started = true;
    }
    // else { called with an externally/container managed transaction }
    Connection conn = null;
    try  {
```

```
            conn = cf.getConnection(); // Verbindung nach Start der globalen Transaktion abrufen
            Session    session = ((XSConnection)conn).getSession();
            ObjectMap  map = session.getMap(MAP_NAME);
            map.insert(key, value); // Or various data access operations
            if (started) {
                tx.commit(); started = false; tx = null;
            }
        }
        finally {
            if (started) {
                try { tx.rollback(); }
                catch (Exception e) { e.printStackTrace(); }
            }
            if (conn != null) { conn.close(); }
        }
    }
/**
     /**
     * Dieses Beispiel veranschaulicht eine Transaktion mit mehreren Partitionen.
     */

  public void testGlobalTransactionTestManagedMultiPartition() throws Exception {
    boolean started = false;
    XSConnectionSpec connSpec = new XSConnectionSpec();
    connSpec.setWriteToMultiplePartitions(true);
    UserTransaction tx = (UserTransaction)ctx.lookup("java:comp/UserTransaction");
    if (tx.getStatus() == Status.STATUS_NO_TRANSACTION) {
      tx.begin();
    started = true;
   }
  // else { wird mit einer externen/containergesteuerten Transaktion aufgerufen }
    Connection conn = null;
       try {
    conn = cf.getConnection(connSpec); // Get connection after the global tran starts
    Session session = ((XSConnection)conn).getSession();
    ObjectMap map = session.getMap(MAP_NAME);
    map.insert(key, value); // Or various data access operations
    if (started) {
      tx.commit(); started = false; tx = null;
      }
    }
    finally {
     if (started) {
       try { tx.rollback(); }
       catch (Exception e) { e.printStackTrace(); }
     }
     if (conn != null) { conn.close(); }
    }
   }
```

J2C-Clientverbindungen verwalten

Java

Die Verbindungsfactory von WebSphere eXtreme Scale enthält eine Clientverbindung von eXtreme Scale, die von Anwendungen gemeinsam genutzt und auch nach Neustarts der Anwendung bestehen bleibt.

Informationen zu diesem Vorgang

Die Clientverbindung enthält eine Management-Bean, die Informationen zum Verbindungsstatus sowie Operationen für das Life-Cycle-Management bereitstellt.

Vorgehensweise

Clientverbindungen verwalten. Beim Abruf der ersten Verbindung vom Verbindungsfactoryobjekt XSConnectionFactory wird eine Clientverbindung von eXtreme Scale zum fernen Datengrid aufgebaut und die ObjectGridJ2CConnection-MBean erstellt. Die Clientverbindung wird für die Dauer des Prozesses verwaltet. Zum Beenden einer Clientverbindung rufen Sie eines der folgenden Ereignisse auf:

- Stoppen Sie den Ressourcenadapter. Ein Ressourcenadapter kann beispielsweise gestoppt werden, wenn er in eine Anwendung eingebettet ist und die Anwendung gestoppt wird.
- Rufen Sie die MBean-Operation resetConnection in der MBean "ObjectGridJ2CConnection" auf. Wenn die Verbindung zurückgesetzt wird, werden alle Verbindungen inaktiviert, Transaktionen beendet und die ObjectGrid-Clientverbindung gelöscht. Nachfolgende Aufrufe der Methoden getConnection in der Verbindungsfactory führen zu einer neuen Clientverbindung.

WebSphere Application Server stellt auch noch weitere Management-Beans für die Verwaltung von J2C-Verbindungen, die Überwachung von Verbindungspools und die Leistung bereit.

Szenario: HTTP-Sitzungsübernahme im Liberty-Profil konfigurieren

Sie können einen Webanwendungsserver so konfigurieren, dass in dem Fall, dass der Web-Server eine HTTP-Anforderung für die Sitzungsreplikation empfängt, die Anforderung an einen oder mehrere Server weiterleitet wird, die im Liberty-Profil ausgeführt werden.

Vorbereitende Schritte

Zum Ausführen dieser Aufgabe müssen Sie das Liberty-Profil installieren. Weitere Informationen finden Sie unter Liberty-Profil installieren.

Informationen zu diesem Vorgang

Das Liberty-Profil enthält keine Sitzungsreplikation. Wenn Sie jedoch WebSphere eXtreme Scale mit dem Liberty-Profil verwenden, können Sie Sitzungen replizieren. Beim Ausfall eines Servers verlieren die Anwendungsbenutzer somit keine Sitzungsdaten.

Wenn Sie das Feature 'webApp' der Serverdefinition hinzufügen und den Sitzungsmanager konfigurieren, können Sie die Sitzungsreplikation in Ihren eXtreme-Scale-Anwendungen verwenden, die im Liberty-Profil ausgeführt werden.

Web-Feature von eXtreme Scale im Liberty-Profil aktivieren

Java

Sie können das Web-Feature aktivieren, um die HTTP-Sitzungsübernahme im Liberty-Profil zu verwenden.

Informationen zu diesem Vorgang


Das Web-Feature ist veraltet. Verwenden Sie stattdessen das Feature "webApp". Wenn Sie das Feature "webApp" der Serverdefinition hinzufügen und den Sitzungsmanager konfigurieren, können Sie die Sitzungsreplikation in Ihren Anwendungen von WebSphere eXtreme Scale verwenden, die im Liberty-Profil ausgeführt werden.

Wenn Sie das Liberty-Profil von WebSphere Application Server installieren, enthält es keine Sitzungsreplikation. Wenn Sie jedoch WebSphere eXtreme Scale mit dem Liberty-Profil verwenden, können Sie Sitzungen replizieren, sodass die Anwendungsbenutzer keine Sitzungsdaten verlieren, falls ein Server ausfällt.

Wenn Sie das Web-Feature der Serverdefinition hinzufügen und den Sitzungsmanager konfigurieren, können Sie die Sitzungsreplikation in Ihren eXtreme-Scale-Anwendungen verwenden, die im Liberty-Profil ausgeführt werden.

Vorgehensweise

Definieren Sie eine Webanwendung für die Ausführung im Liberty-Profil.

Nächste Schritte

Konfigurieren Sie als Nächstes ein Web-Server-Plug-in, das HTTP-Anforderungen an mehrere Server im Liberty-Profil weiterleitet.

Feature "webGrid" von eXtreme Scale im Liberty-Profil aktivieren

Verwenden Sie das Feature "webGrid", wenn automatisch ein Container zum Hosten der Clients für die HTTP-Sitzungsreplikation im Liberty-Profil gestartet werden soll.

Informationen zu diesem Vorgang

Wenn Sie das Liberty-Profil von WebSphere Application Server installieren, enthält es keine Sitzungsreplikation. Wenn Sie jedoch WebSphere eXtreme Scale mit dem Liberty-Profil verwenden, können Sie Sitzungen replizieren, sodass die Anwendungsbenutzer keine Sitzungsdaten verlieren, falls ein Server ausfällt.

Wenn Sie das Feature "webGrid" der Serverdefinition hinzufügen und den Sitzungsmanager konfigurieren, können Sie die Sitzungsreplikation in Ihren eXtreme-Scale-Anwendungen verwenden, die im Liberty-Profil ausgeführt werden.

Vorgehensweise

Fügen Sie das Feature "webGrid" der Datei `server.xml` des Liberty-Profils hinzu. Das Feature ""webGrid" umfasst das Client-Feature und das Server-Feature. Wahrscheinlich möchten Sie Ihre Webanwendungen von den Datengrids trennen. Angenommen, Sie haben einen Liberty-Profilserver für Ihre Webanwendungen und einen anderen Liberty-Profilserver für das Hosten des Datengrids.

```
<featureManager>
<feature>eXtremeScale_webGrid-1.1</feature>
</featureManager>
```

Ergebnisse

Ihre Webanwendungen können ihre Sitzungsdaten jetzt in einem Grid von WebSphere eXtreme Scale persistent speichern.

Beispiel

Das Feature "webGrid" hat Metaeigenschaften, die Sie im Element "xsWebGrid" der Datei `server.xml` festlegen können. Sehen Sie sich die folgende Beispieldatei `server.xml` an, die das Feature "webGrid" enthält, das Sie beim Herstellen der Verbindung zum Datengrid über Fernzugriff verwenden.

```
<server description="Airport Entry eXtremeScale Getting Started Client Web Server">
<!--
This sample program is provided AS IS and may be used, executed, copied and modified
without royalty payment by customer
```

```
(a) for its own instruction and study,
(b) in order to develop applications designed to run with an IBM WebSphere product,
either for customer's own internal use or for redistribution by customer, as part of such an
application, in customer's own products.
Lizenziertes Material - Eigentum der IBM
5724-X67, 5655-V66 (C) COPYRIGHT International Business Machines Corp. 2012
-->
<!-- Features aktivieren -->
<featureManager>
<feature>eXtremeScale.webGrid-1.1</feature>
</featureManager>

<xsServer catalogServer="true"/>

<xsWebGrid objectGridName="session" catalogHostPort="remoteHost:2809" securityEnabled="false" />

</server>
```

eXtreme-Scale-Feature "webApp" im Liberty-Profil aktivieren

Ein Liberty-Profilserver kann ein Datengrid hosten, in dem Daten für Anwendungen zwischengespeichert werden, um HTTP-Sitzungsdaten für die Fehlertoleranz zu replizieren.

Informationen zu diesem Vorgang

Wenn Sie das Liberty-Profil von WebSphere Application Server installieren, enthält es keine Sitzungsreplikation. Wenn Sie jedoch WebSphere eXtreme Scale mit dem Liberty-Profil verwenden, können Sie Sitzungen replizieren, sodass die Anwendungsbenutzer keine Sitzungsdaten verlieren, falls ein Server ausfällt.

Wenn Sie das Feature "webApp" der Serverdefinition hinzufügen und den Sitzungsmanager konfigurieren, können Sie die Sitzungsreplikation in Ihren eXtreme-Scale-Anwendungen verwenden, die im Liberty-Profil ausgeführt werden.

Vorgehensweise

Fügen Sie das Feature "webApp" der Datei `server.xml` des Liberty-Profils hinzu. Das Feature "webApp" enthält das Client-Feature, aber nicht das Server-Feature. Wahrscheinlich möchten Sie Ihre Webanwendungen von den Datengrids trennen. Angenommen, Sie haben einen Liberty-Profilserver für Ihre Webanwendungen und einen anderen Liberty-Profilserver für das Hosten des Datengrids.

```
<featureManager>
<feature>eXtremeScale_webapp-1.1</feature>
</featureManager>
```

Ergebnisse

Ihre Webanwendungen können ihre Sitzungsdaten jetzt in einem Grid von WebSphere eXtreme Scale persistent speichern.

Beispiel

Sehen Sie sich die folgende Beispieldatei `server.xml` an, die das Feature "webApp" enthält, das Sie beim Herstellen der Verbindung zum Datengrid über Fernzugriff verwenden.

```
<server description="Airport Entry eXtremeScale Getting Started Client Web Server">
<!--
This sample program is provided AS IS and may be used, executed, copied and modified
without royalty payment by customer
(a) for its own instruction and study,
(b) in order to develop applications designed to run with an IBM WebSphere product,
either for customer's own internal use or for redistribution by customer, as part of such an
```

```
application, in customer's own products.
Lizenziertes Material - Eigentum der IBM
5724-X67, 5655-V66 (C) COPYRIGHT International Business Machines Corp. 2012
-->
<!-- Enable features -->
<featureManager>
<feature>eXtremeScale.webapp-1.1</feature>
</featureManager>

<httpEndpoint id="defaultHttpEndpoint"
host="*"
httpPort="${default.http.port}"
httpsPort="${default.https.port}" />

<xsWebApp objectGridName="session" catalogHostPort="remoteHost:2809" securityEnabled="false" />

</server>
```

Nächste Schritte

Das Feature "webApp" hat Metaeigenschaften, die Sie im Element "xsWebApp" der Datei `server.xml` festlegen können. Weitere Informationen finden Sie unter Eigenschaften des Features "webApp" des Liberty-Profils.

Web-Server-Plug-in für die Weiterleitung von Anforderungen an mehrere Server im Liberty-Profil konfigurieren

Java

Verwenden Sie diese Aufgabe, um das Web-Server-Plug-in für die Verteilung von HTTP-Server-Anforderungen zwischen mehreren Servern im Liberty-Profil zu verteilen.

Vorbereitende Schritte

Führen Sie die folgende Aufgabe aus, bevor Sie das Web-Server-Plug-in für die Weiterleitung von HTTP-Anforderungen an mehrere Server konfigurieren:

- „eXtreme-Scale-Feature "webApp" im Liberty-Profil aktivieren” auf Seite 273

Informationen zu diesem Vorgang

Sie können das Web-Server-Plug-in so konfigurieren, dass in dem Fall, dass der Web-Server eine HTTP-Anforderung für dynamische Ressourcen empfängt, die Anforderung an einen oder mehrere Server weiterleitet wird, die im Liberty-Profil ausgeführt werden.

Vorgehensweise

Informationen zum Ausführen dieser Aufgabe finden Sie unter Liberty-Profil mit einem Web-Server-Plug-in konfigurieren im Information Center von WebSphere Application Server.

Nächste Schritte

Führen Sie als Nächstes die Dateien vom Typ `plugin-cfg.xml` von mehreren Anwendungsserverzellen zusammen. Außerdem müssen Sie sicherstellen, dass eindeutige Klon-IDs für jeden Anwendungsserver vorhanden sind, der im Liberty-Profil ausgeführt wird.

Plug-in-Konfigurationsdateien für die Implementierung im Anwendungsserver-Plug-in zusammenführen

Java

Generieren Sie nach der Konfiguration einer eindeutigen Klon-ID in der Liberty-Konfigurationsdatei "server.xml" Plug-in-Konfigurationsdateien.

Vorbereitende Schritte

Wenn Sie Plug-in-Konfigurationsdateien generieren und zusammenführen, um die HTTP-Sitzungsübernahme in einem Liberty-Profil zu konfigurieren, müssen Sie die folgenden Aufgaben ausführen:

- „Web-Feature von eXtreme Scale im Liberty-Profil aktivieren" auf Seite 271
- „Web-Server-Plug-in für die Weiterleitung von Anforderungen an mehrere Server im Liberty-Profil konfigurieren" auf Seite 274

Informationen zu diesem Vorgang

Verwenden Sie für das Ausführen dieser Aufgabe die Administrationskonsole von WebSphere Application Server.

Vorgehensweise

1. Führen Sie die Dateien mit dem Namen `plugin-cfg.xml` aus mehreren Anwendungsserverzellen zusammen. Sie können die Dateien mit dem Namen `plugin-cfg.xml` entweder manuell zusammenführen oder diese Dateien mit dem Tool "pluginCfgMerge" aus mehreren Anwendungsserverprofilen automatisch in einer einzigen Ausgabedatei zusammenführen lassen. Die Dateien "pluginCfgMerge.bat" und "pluginCfgMerge.sh" befinden sich im Verzeichnis *Installationsstammverzeichnis*`/bin`.

   Weitere Informationen zum manuellen Zusammenführen der Dateien mit dem Namen `plugin-cfg.xml` finden Sie im technischen Hinweis zum Zusammenführen der Dateien `plugin-cfg.xml` aus mehreren Anwendungsserverprofilen.
2. Stellen Sie sicher, dass der cloneID-Wert für jeden Anwendungsserver eindeutig ist. Überprüfen Sie den cloneID-Wert für jeden Anwendungsserver in der zusammengeführten Datei, um sicherzustellen, dass dieser Wert für jeden Anwendungsserver eindeutig ist. Wenn die cloneID-Werte in der zusammengeführten Datei nicht eindeutig sind oder wenn Sie mit Speicher-zu-Speicher-Sitzungsreplikation im Peer-to-Peer-Modus arbeiten, verwenden Sie die Administrationskonsole, um eindeutige cloneID-Werte für die HTTP-Sitzungen zu konfigurieren.

   Führen Sie die folgenden Schritte aus, um eine eindeutige Klon-ID für die HTTP-Sitzung über die Administrationskonsole von WebSphere Application Server zu konfigurieren:

   a. Klicken Sie auf **Server** > **Servertypen** > **WebSphere-Anwendungsserver** > *Servername*.
   b. Klicken Sie unter "Containereinstellungen" auf **Einstellungen des Webcontainers** > **Webcontainer**.
   c. Klicken Sie unter "Weitere Eigenschaften" auf **Angepasste Eigenschaften** > **Neu**.

d. Geben Sie `HttpSessionCloneId` im Feld **Name** und einen eindeutigen Wert für den Server im Feld **Wert** ein. Der eindeutige Wert muss aus acht bis neun alphanumerischen Zeichen bestehen. Ein gültiger cloneID-Wert ist beispielsweise `test1234`.

e. Klicken Sie auf **Anwenden** oder **OK**.

f. Klicken Sie auf **Speichern**, um die Konfigurationsänderungen in der Masterkonfiguration zu speichern.

3. Kopieren Sie die zusammengeführte Datei `plugin-cfg.xml` in das Verzeichnis *Installationsstammverzeichnis_für_Plug-ins*`/config/Name_des_Web-Servers` auf dem Web-Server-Host.
4. Stellen Sie sicher, dass Sie das richtige Betriebssystem und die richtigen Dateizugriffsberechtigungen für die zusammengeführte Datei `plugin-cfg.xml` definiert haben. Diese Dateizugriffsberechtigungen benötigt der HTTP-Server-Plug-in-Prozess zum Lesen der Datei.

Ergebnisse

Nach Abschluss dieser Aufgabe haben Sie eine einzige Plug-in-Konfigurationsdatei für mehrere Anwendungsserverzellen, und Ihre eXtreme-Scale-Anwendungen, die im Liberty-Profil ausgeführt werden, sind für die Sitzungsreplikation aktiviert.

Szenario: Grid-Server mit Eclipse-Tools im Liberty-Profil ausführen

Sie können Eclipse-Tools verwenden, um Server von WebSphere eXtreme Scale im Liberty-Profil von WebSphere Application Server auszuführen. Die Eclipse-Tools bieten eine komfortable Methode für die Ausführung der Server in derselben Eclipse-Umgebung, in der Sie auch Ihre eXtreme-Scale-Anwendungen entwickeln, konfigurieren und implementieren.

Informationen zu diesem Vorgang

Mit den Eclipse-Tools können Sie eXtreme-Scale-Server für die Ausführung im Liberty-Profil konfigurieren. Wenn Sie diese Aufgabe manuell ausführen, fügen Sie der Datei `server.xml` die unterstützten Liberty-Features hinzu. Wenn Sie die Eclipse-Tools verwenden, können Sie diese Aufgabe und weitere Entwicklungsaufgaben jedoch mit Eclipse Java EE IDE for Web Developers Version Indigo Service, Release 1 ausführen.

Entwicklertools des Liberty-Profils für WebSphere eXtreme Scale installieren

Eclipse stellt eine grafische Benutzerschnittstelle (GUI) bereit, die Sie für die Ausführung der Server von WebSphere eXtreme Scale im Liberty-Profil verwenden können. Zur Verwendung dieser GUI müssen Sie die Tools des Liberty-Profils von WebSphere eXtreme Scale Version 8.5 installieren.

Informationen zu diesem Vorgang

Die Tools können mit einer der folgenden Methoden installiert werden:

- Installation über Eclipse Marketplace. Klicken Sie auf **Hilfe** > **Eclipse Marketplace**.
- Installation durch Ziehen eines Installationssymbols in eine aktive Workbench. Diese Option ist nur für die Installation der Entwicklertools in Eclipse IDE for Java EE Developers 3.7 oder höher verfügbar.

Sie müssen IBM WebSphere Application Server V8.5 Liberty Profile Developer Tools installieren, um IBM WebSphere eXtreme Scale V8.5 Liberty Profile Developer Tools verwenden zu können. Deshalb wird in den Schritten dieser Aufgabe die Installation beider Entwicklertools beschrieben.

Vorgehensweise

- Installation über Eclipse Marketplace.
  1. Starten Sie Ihre Eclipse-Workbench.
  2. Klicken Sie auf "Hilfe > Eclipse Marketplace".
  3. Geben Sie im Feld "Suchen" WebSphere ein.
  4. Suchen Sie in der Ergebnisliste **IBM WebSphere Application Server V8.5 Liberty Profile Developer Tools**, und klicken Sie dann auf **Installieren**.
  5. Die Seite "Ausgewählte Features bestätigen" erscheint. FAhren Sie mit der Installation im Schritt "Installation durchführen" fort.
  6. Führen Sie jeden der vorherigen Schritte aus, um **IBM WebSphere eXtreme Scale V8.5 Liberty Profile Developer Tools** zu installieren.
- Installation durchführen.
  1. Blenden Sie den Knoten für die Tools ein, die Sie installiert haben.
  2. Wählen Sie **IBM WebSphere Application Server V8.5 Liberty Profile Developer Tools** oder **IBM WebSphere eXtreme Scale V8.5 Liberty Profile Developer Tools** aus.
  3. Wählen Sie die optionalen Features aus, die Sie installieren möchten. Klicken Sie abschließend auf **Weiter**.

     Hinweis: Wenn Sie weitere optionale Installationsfeatures, wie z. B. Tool-Features von WebSphere Application Server für Version 8.5, 8.0 oder 7.0, ist ein separater Satz von Installationsanweisungen im Abschnitt Übersicht über IBM WebSphere Application Server Developer Tools for Eclipse Version 8.5 im Information Center von WebSphere Application Server verfügbar.
  4. Lesen Sie auf der Seite "Lizenzen überprüfen" die Lizenzinformation.
  5. Wenn Sie den Bedingungen zustimmen, klicken Sie auf **Ich akzeptiere die Bedingungen der Lizenzvereinbarung**, und klicken Sie dann auf **Fertig stellen**. Der Installationsprozess wird gestartet.
  6. Starten Sie die Workbench nach Abschluss der Installation erneut.

Entwicklungsumgebung in Eclipse einrichten

Nach der Installation der Eclipse-Tools des Liberty-Profils für WebSphere eXtreme Scale müssen Sie Ihre eXtreme-Scale-Server im Liberty-Profil konfigurieren und ein Eclipse-Projekt generieren, in dem Sie mit den Entwicklungsaufgaben beginnen können.

eXtreme Scale im Liberty-Profil mit Eclipse-Tools konfigurieren

Sie müssen Ihre Server von WebSphere eXtreme Scale für die Ausführung im Liberty-Profil von WebSphere Application Server aktivieren. Führen Sie diese Aufgabe aus, um eXtreme-Scale-Server mit Eclipse-Tools zu konfigurieren.

Vorbereitende Schritte

Sie müssen einen Liberty-Profilserver in Eclipse definieren. Informationen zum Ausführen dieser Aufgabe finden Sie unter Liberty-Profilserver mit Entwicklertools erstellen.

Informationen zu diesem Vorgang

Wenn Sie den eXtreme-Scale-Server konfigurieren, müssen Sie die Servereigenschaften angeben und diese Eigenschaften in der Datei `server.xml` des Liberty-Profils im Verzeichnis *`WLP-Ausgangsverzeichnis`*`/usr/servers/`*`Name_Ihres_Servers`* angeben. Diese Serverdefinition muss eXtreme Scale im Liberty-Profil ausführen.

Diese Prozedur beinhaltet auch das Hinzufügen der Konfiguration aus der eXtreme-Scale-Servereigenschaftendatei `xsServerConfig.xml` zur Datei `server.xml`.

Vorgehensweise

1. Generieren Sie die eXtreme-Scale-Servereigenschaftendatei.
   a. Klicken Sie auf **Datei** > **Neu** > **Sonstige**.
   b. Erweitern Sie den Eintrag **WebSphere eXtreme Scale**, und wählen Sie **Konfigurationsdatei des Container-Servers** aus. Klicken Sie auf **Weiter**. Das Fenster "Konfigurationsdatei des eXtreme-Scale-Servers" wird angezeigt.
   c. Klicken Sie auf **Durchsuchen**, um anzugeben, wo das Liberty-Profil installiert ist. Wählen Sie anschließend die Liberty-Profilserverdefinition aus, für die Sie Ihre eXtreme-Scale-Server konfigurieren möchten. Klicken Sie auf **Weiter**. Das Fenster "Allgemeine Serverkonfiguration" erscheint.
   d. Führen Sie die Serverkonfiguration durch. Klicken Sie auf **Weiter**. Das Fenster "Konfiguration des Container-Servers" erscheint.
   e. Führen Sie die Konfiguration des Container-Servers durch. Klicken Sie auf **Weiter**.
   f. Wenn Sie die Katalogserverkonfiguration eingeschlossen haben, erscheint ein weiteres Fenster, in dem Sie die Einstellungen des Katalogservers angeben. Klicken Sie auf **Weiter**. Das Fenster "Konfiguration der Serverprotokollierung" erscheint.
   g. Füllen Sie die Seiten mit den Protokolldaten aus, und klicken Sie dann auf **Weiter**, bis das Fenster "Sicherheitskonfiguration" erscheint.
   h. Optional: Geben Sie die Position der Datei `objectGridSecurity.xml` an, in der die Sicherheitseigenschaften beschrieben sind, die für alle Server gelten, einschließlich Katalogservern und Container-Servern. Ein Beispiel für die definierten Sicherheitseigenschaften ist die Authentifikatorkonfiguration, die die Benutzerregistry und den Authentifizierungsmechanismus darstellt. Der für diese Eigenschaft angegebene Dateiname muss das URL-Format haben, z. B. `file:///tmp/og/objectGridSecurity.xml`.
   i. Klicken Sie auf **Fertig stellen**.

   Es wird eine Konfigurationsdatei im Liberty-Profil generiert.
2. Schließen Sie die Konfiguration aus der Servereigenschaftendatei eXtreme Scale in die Datei `server.xml` ein.
   a. Öffnen Sie die Ansicht "Server" in Eclipse.
   b. Erweitern Sie den Eintrag "Liberty-Server", um Ihre XML-Serverkonfigurationsdatei zu suchen.
   c. Klicken Sie doppelt auf den Eintrag für Ihre Serverkonfiguration, um die Datei zu öffnen.
   d. Klicken Sie auf **Hinzufügen**, und wählen Sie **Include** aus, um der Datei `server.xml` eine Include-Anweisung hinzuzufügen. Klicken Sie auf **OK**.
   e. Klicken Sie unter "Include-Details" auf **Durchsuchen**. Das Fenster "Zur Include-Datei navigieren" erscheint.

f. Wählen Sie die Datei **xsServerConfig.xml** aus, um die Serverkonfigurationseinstellungen, die Sie in Schritt 1 erstellt haben, einzuschließen. Klicken Sie auf **OK**.

Nächste Schritte

Die eXtreme-Scale-Serverkonfigurationsdatei `xsServerConfig.xml` ist jetzt in der Datei `server.xml` des Liberty-Profils enthalten. Jetzt können Sie den Liberty-Profilserver, starten, in dem Ihre eXtreme-Scale-Server ausgeführt werden.

OSGi-Bundleprojekt für eXtreme-Scale-Grid-Entwicklung

Wenn Sie Eclipse als Entwicklungsumgebung für Ihre Server von WebSphere eXtreme Scale im Liberty-Profil verwenden möchten, müssen Sie im unterstützten OSGi-Framework (Open Services Gateway initiative) ein Eclipse-Projekt erstellen.

Vorgehensweise

1. Erstellen Sie das OSGi-Bundleprojekt in Eclipse.
   a. Klicken Sie auf **Datei** > **Neu** > **Projekt**. Das Fenster zur Assistentenauswahl erscheint.
   b. Blenden Sie den Ordner von WebSphere eXtreme Scale ein, und wählen Sie das Projekt **Object-Grid** aus. Das Fenster "Object-Grid-Projekt" erscheint.
   c. Klicken Sie auf **Hinzufügen**, und geben Sie den Namen einer BackingMap ein, um die Object-Grid-Map hinzuzufügen, für die Sie die Entwicklungsaktivitäten ausführen möchten. Sie können auf dieser Seite mehrere Maps eingeben. Klicken Sie auf **Weiter**.
   d. Geben Sie für jede eingegebene Map Object-Grid-Parameter an. Klicken Sie auf **Weiter**.
   e. Geben Sie die Implementierungsparameter an, und klicken Sie dann auf **Fertig stellen**.

   Das OSGi-Bundleprojekt wird erstellt, und Sie können auf die APIs von eXtreme Scale zugreifen, um Entwicklungsaktivitäten im Liberty-Profil auszuführen. Das Bundle enthält die Datei `gridBlueprint.xml`. Diese Datei enthält die Position der eXtreme-Scale-Konfigurationsdateien, `objectGrid.xml` und `gridDeployment.xml`. Diese Konfigurationsdateien enthalten die Maps, die Sie in Schritt c erstellt haben.
2. Exportieren Sie das Bundleprojekt, und speichern Sie das Bundle im Ordner "grids". Sie müssen das Projekt exportieren, um eXtreme-Scale-Anwendungen im Liberty-Profil zu implementieren. Wenn Sie das Projekt exportieren, wird es als Bundle-JAR-Datei (Java-Archiv) in den Ordner *Serverdefinition_des_Liberty-Profils*`/grids` exportiert.
   a. Klicken Sie mit der rechten Maustaste auf das soeben erstellte Projekt, und wählen Sie dann **Exportieren** > **OSGi-Bundle oder -Fragment** aus. Das Fenster für den OSGi-Anwendungsexport erscheint.
   b. Geben Sie an, an welche Position Sie die Bundle-JAR-Datei exportieren möchten. Klicken Sie auf **Fertig stellen**.

Speicher-zu-Speicher-Replikations- oder Datenbanksitzung von WebSphere Application Server auf die Sitzungsverwaltung von WebSphere eXtreme Scale migrieren

Java

Sie können alle zuvor definierten Speicher-zu-Speicher-Replikations- oder Datenbanksitzungen auf die Sitzungsverwaltung von WebSphere eXtreme Scale migrieren.

Vorbereitende Schritte

- Für die Sitzungsunterstützung für Clientanwendungen, die in WebSphere Application Server im Cluster ausgeführt werden, muss WebSphere eXtreme Scale in den Knotenimplementierungen von WebSphere Application Server installiert werden. Dies gilt auch für den Deployment-Manager-Knoten. Weitere Informationen finden Sie unter WebSphere eXtreme Scale oder WebSphere eXtreme Scale Client mit WebSphere Application Server installieren.
- Eine Gridumgebung von WebSphere eXtreme Scale, die sich aus einem oder mehreren Katalog- und Container-Servern zusammensetzt, muss gestartet sein. Weitere Informationen finden Sie unter Eigenständige Server starten und stoppen.

  Anmerkung: Wenn WebSphere eXtreme Scale nicht angezeigt wird, wurde Ihr Profil von WebSphere Application Server nicht für WebSphere eXtreme Scale erweitert. Weitere Informationen finden Sie unter Profile für WebSphere eXtreme Scale erstellen und erweitern.
- Wenn in den Katalogservern in Ihrer Katalogservicedomäne Secure Sockets Layer (SSL) aktiviert ist oder Sie SSL für eine Katalogservicedomäne mit SSL-Unterstützung verwenden möchten, muss die globale Sicherheit in der Administrationskonsole von WebSphere Application Server aktiviert werden. Sie benötigen SSL für einen Katalogserver, indem Sie in Servereigenschaftendatei für das Attribut "transportType" den Wert "SSL-Required" festlegen. Weitere Informationen finden Sie unter Globale Sicherheitseinstellungen.

Informationen zu diesem Vorgang

Die Schritte in diesem Szenario beziehen sich auf Version 8.5 der Administrationskonsole von WebSphere Application Server. Diese Informationen können je nach Version von WebSphere Application Server, die Sie verwenden, geringfügig variieren.

Anmerkung: WebSphere eXtreme Scale Version 8.6 wird in den Versionen von WebSphere Application Server vor Version 7.0 nicht unterstützt.

Vorherige Konfigurationseinstellungen in der Administrationskonsole von WebSphere Application Server notieren

Java

Im Rahmen der Migration auf eine Sitzung von WebSphere eXtreme Scale sollten Sie Ihre vorherigen Konfigurationseinstellungen in der Administrationskonsole von WebSphere Application Server notieren. Bei der Migration auf eine Sitzung von WebSphere eXtreme Scale müssen die Konfigurationseinstellungen das widerspiegeln, was Sie bereits für Ihre Datenbank- bzw. Speicher-zu-Speicher-Sitzung konfiguriert haben.

Informationen zu diesem Vorgang

Es gibt bestimmte Einstellungen in der Administrationskonsole von WebSphere Application Server, die Sie notieren sollten. Sie benötigen diese Werte, wenn Sie die Datei `splicer.properties` aktualisieren. Die Schritte in dieser Prozedur beziehen

sich auf Version 8.5 der Administrationskonsole von WebSphere Application Server. Diese Informationen können je nach Version von WebSphere Application Server, die Sie verwenden, geringfügig variieren.

Anmerkung: WebSphere eXtreme Scale Version 8.6 wird in den Versionen von WebSphere Application Server vor Version 7.0 nicht unterstützt.

Vorgehensweise

1. Starten Sie die Administrationskonsole von WebSphere Application Server.
   - Wenn Sie bereits konfigurierte Einstellungen auf Serverebene haben, verwenden Sie den folgenden Navigationspfad:
     a. **Server > Servertypen > WebSphere-Anwendungsserver**
     b. Wählen Sie im Bereich **Anwendungsserver** den Eintrag für **Ihren Servernamen** aus.
     c. Klicken Sie im Bereich **Containereinstellungen** auf **Sitzungsverwaltung**.
   - Wenn Sie bereits konfigurierte Einstellungen auf Anwendungsebene haben, verwenden Sie den folgenden Navigationspfad:
     a. **Anwendungen > Alle Anwendungen**
     b. Wählen Sie im Bereich **Anwendungsserver** den Eintrag für **Ihren Anwendungsnamen** aus.
     c. Klicken Sie im Bereich **Eigenschaften des Webmoduls** auf **Sitzungsverwaltung**.
2. Wählen Sie unter **Allgemeine Eigenschaften** das Kontrollkästchen **Überlauf zulassen** aus.
3. Notieren Sie die Einstellungen von WebSphere Application Server im Abschnitt **Allgemeine Eigenschaften**. Sie benötigen diese Werte später, um die Eigenschaften in der Datei `splicer.properties` zu aktualisieren.

Tabelle 10. Konfigurationseinstellungen zum Aktualisieren der Datei "splicer.properties"

Einstellungen in der Administrationskonsole von WebSphere Application Server	**Zu aktualisierende Eigenschaften in der Datei "splicer.properties"**
Cookies aktivieren	useCookies
Umschreiben von URLs aktivieren	useURLEncoding
Maximale Anzahl der Sitzungen im Speicher	sessionTableSize

4. Wenn im Bereich **Allgemeine Eigenschaften** das Kontrollkästchen **Cookies aktivieren** ausgewählt ist, klicken Sie auf das Kontrollkästchen, und notieren Sie die Einstellungen von WebSphere Application Server. Sie benötigen diese Werte später, um die Eigenschaften in der Datei `splicer.properties` zu aktualisieren.

Tabelle 11. Konfigurationseinstellungen für die Eigenschaften in der Datei "splicer.properties"

Einstellungen in der Administrationskonsole von WebSphere Application Server	**Zu aktualisierende Eigenschaften in der Datei "splicer.properties"**
Cookiedomäne	cookieDomain
Cookiepfad	cookiePath

5. Klicken Sie auf **Sitzungsverwaltung** und anschließend im Bereich **Weitere Eigenschaften** auf **Einstellungen für eine verteilte Umgebung**.

6. Ändern Sie im Bereich **Verteilte Sitzungen** die vorherige Datenbankkonfiguration bzw. Konfiguration für die Speicher-zu-Speicher-Replikation in **Ohne**.
7. Klicken Sie auf **Angepasste Optimierungsparameter**, und notieren Sie die Einstellungen von WebSphere Application Server. Sie benötigen diese Werte später, um die Eigenschaften in der Datei `splicer.properties` zu aktualisieren.

Tabelle 12. Konfigurationseinstellungen für die Eigenschaften in der Datei "splicer.properties"

Einstellungen in der Administrationskonsole von WebSphere Application Server	**Zu aktualisierende Eigenschaften in der Datei "splicer.properties"**
Schreibfrequenz	replicationInterval
Zu schreibender Inhalt	fragmentedSession

Nächste Schritte

Erstellen Sie als Nächstes die Katalogservicedomäne für eine Sitzung von WebSphere eXtreme Scale.

Katalogservicedomäne für die Sitzungsverwaltung von WebSphere eXtreme Scale erstellen

Java

Im Rahmen der Migration auf eine Sitzung von WebSphere eXtreme Scale müssen Sie eine Katalogservicedomäne in der Administrationskonsole von WebSphere Application Server erstellen.

Informationen zu diesem Vorgang

Die Schritte in dieser Prozedur beziehen sich auf Version 8.5 der Administrationskonsole von WebSphere Application Server. Diese Informationen können je nach Version von WebSphere Application Server, die Sie verwenden, geringfügig variieren.

Anmerkung: WebSphere eXtreme Scale Version 8.6 wird in den Versionen von WebSphere Application Server vor Version 7.0 nicht unterstützt.
Erstellen Sie eine Katalogservicedomäne für WebSphere eXtreme Scale in der Administrationskonsole von WebSphere Application Server administrative. Weitere Informationen finden Sie unter Katalogservicedomänen in WebSphere Application Server erstellen.

Vorgehensweise

1. Starten Sie die Administrationskonsole von WebSphere Application Server.
2. Klicken Sie im Ausgangsmenü auf **Systemverwaltung > WebSphere eXtreme Scale > Katalogservicedomänen**.

   Anmerkung: Wenn WebSphere eXtreme Scale nicht angezeigt wird, wurde Ihr Profil von WebSphere Application Server nicht für WebSphere eXtreme Scale erweitert. Weitere Informationen finden Sie unter Profile für WebSphere eXtreme Scale erstellen und erweitern.
3. Klicken Sie auf **Neu**.
4. Geben Sie im Feld **Name** einen Namen für den Katalogservice an.

5. Wählen Sie im Bereich **Katalogserver** den Eintrag **Ferner Server** aus, und geben Sie die Position oder den Namen des fernen Servers im Feld an.
6. Geben Sie im Feld **Listener-Port** eine Portnummer an.
7. Klicken Sie auf **Anwenden** oder **OK**, und speichern Sie die Konfiguration.

Nächste Schritte

Verwenden Sie als Nächstes die vorherigen Konfigurationseinstellungen, die Sie in der Administrationskonsole von WebSphere Application Server notiert haben, um der Sitzungsverwaltung von WebSphere eXtreme Scale eine Anwendung oder einen Anwendungsserver zuzuordnen.

WebSphere eXtreme Scale für die Verwendung der vorherigen Konfigurationseinstellungen konfigurieren

Java

Wenn Sie Ihre vorherigen Konfigurationseinstellungen verwenden, die Sie in der Administrationskonsole von WebSphere Application Server notiert haben, müssen Sie diese Einstellungen verwenden, um der Sitzungsverwaltung von WebSphere eXtreme Scale eine Anwendung oder einen Anwendungsserver zuzuordnen.

Informationen zu diesem Vorgang

Die Schritte in dieser Prozedur beziehen sich auf Version 8.5 der Administrationskonsole von WebSphere Application Server. Diese Informationen können je nach Version von WebSphere Application Server, die Sie verwenden, geringfügig variieren.

Anmerkung: WebSphere eXtreme Scale Version 8.6 wird in den Versionen von WebSphere Application Server vor Version 7.0 nicht unterstützt.

Vorgehensweise

- Wenn Sie eine Anwendung so konfigurieren möchten, dass sie der Sitzungsverwaltung von WebSphere eXtreme Scale zugeordnet wird, führen Sie die folgenden Schritte aus:
  1. Starten Sie die Administrationskonsole von WebSphere Application Server.
  2. Klicken Sie im Ausgangsmenü auf **Anwendungen > Alle Anwendungen**.
  3. Wählen Sie im Bereich **Anwendungsserver** den **Anwendungsnamen** aus.
  4. Klicken Sie im Bereich **Eigenschaften des Webmoduls** auf **Sitzungsverwaltung**.
  5. Klicken Sie auf **eXtreme Scale - Einstellungen für die Sitzungsverwaltung**.
  6. Wenn WebSphere eXtreme Scale nicht angezeigt wird, wurde Ihr Profil von WebSphere Application Server nicht für WebSphere eXtreme Scale erweitert. Weitere Informationen finden Sie unter Profile für WebSphere eXtreme Scale erstellen und erweitern.
  7. Führen Sie die folgenden Schritte aus, um eine Anwendung für WebSphere eXtreme Scale in einer eigenständigen Umgebung zu konfigurieren:
     a. Wählen Sie in der Liste **Sitzungspersistenz verwalten mit** den Eintrag **Fernes eXtreme-Scale-Datengrid** aus.
     b. Wählen Sie in der Liste die Katalogservicedomäne aus, die Sie erstellt haben.
     c. Klicken Sie auf **Durchsuchen**, um das Grid auszuwählen.

8. Klicken Sie auf **Anwenden** oder **OK**, und speichern Sie die Konfiguration.
9. Es wird eine neue Datei "splicer.properties" für diese Anwendung erstellt. Die Position der Datei `splicer.properties` ist der Wert der neuen Eigenschaft `{Anwendungsname},com.ibm.websphere.xs.sessionFilterProps`. Zum Anzeigen dieser angepassten Eigenschaften navigieren Sie zu **Systemverwaltung > Zelle**, und klicken Sie dann auf **Angepasste Eigenschaften**.
10. Aktualisieren Sie die Datei `splicer.properties` mit den Werten, die Sie in „Vorherige Konfigurationseinstellungen in der Administrationskonsole von WebSphere Application Server notieren" auf Seite 280 abgerufen haben.
11. Starten Sie die Anwendungsserverprozesse erneut.

Anmerkung: Ändern Sie die Datei `splicer.properties` auf Deployment-Manager-Ebene so, dass die Eigenschaften mit dem Node Agent synchronisiert sind. Wenn Sie die Datei `splicer.properties` auf Knotenebene aktualisieren, überschreibt der Deployment Manager die Datei `splicer.properties` bei der nächsten Synchronisation.

Anmerkung: Wenn Sie zur Verwaltung der Datenbanksitzungen und dann zur Sitzungsverwaltung von WebSphere eXtreme Scale zurückkehren, wird die Datei `splicer.properties` erneut erstellt, sodass alle vorgenommenen Änderungen überschrieben werden. Eine Beschreibung des Synchronisationsprozesses (vom Deployment Manager über die Anmerkungen bis hin zu den Änderungen) finden Sie unter System Management File Synchronization.

- Wenn Sie einen Anwendungsserver so konfigurieren möchten, dass er der Sitzungsverwaltung von WebSphere eXtreme Scale zugeordnet wird, führen Sie die folgenden Schritte aus:
  1. Starten Sie die Administrationskonsole von WebSphere Application Server.
  2. Klicken Sie im Ausgangsmenü auf **Server > Servertypen > WebSphere-Anwendungsserver**.
  3. Wählen Sie im Bereich **Anwendungsserver** den Eintrag für **Ihren Servernamen** aus.
  4. Klicken Sie im Bereich **Containereinstellungen** auf **Sitzungsverwaltung**.
  5. Klicken Sie auf **eXtreme Scale - Einstellungen für die Sitzungsverwaltung**.

     Anmerkung: Wenn WebSphere eXtreme Scale nicht angezeigt wird, wurde Ihr Profil von WebSphere Application Server nicht für WebSphere eXtreme Scale erweitert. Weitere Informationen finden Sie unter Profile für WebSphere eXtreme Scale erstellen und erweitern.
  6. Führen Sie die folgenden Schritte aus, um einen Anwendungsserver für WebSphere eXtreme Scale in einer eigenständigen Umgebung zu konfigurieren:
     a. Wählen Sie in der Liste **Sitzungspersistenz verwalten mit** den Eintrag **Fernes eXtreme-Scale-Datengrid** aus.
     b. Wählen Sie in der Liste die Katalogservicedomäne aus, die Sie erstellt haben.
     c. Klicken Sie auf **Durchsuchen**, um das Grid auszuwählen.
  7. Klicken Sie auf **Anwenden** oder **OK**, und speichern Sie die Konfiguration.
  8. Es wird eine neue Datei "splicer.properties" für diese Anwendung erstellt. Die Position der Datei "splicer.properties" ist der Wert der neuen Eigenschaft `com.ibm.websphere.xs.sessionFilterProps`. Zum Anzeigen dieser angepassten Eigenschaft navigieren Sie zu **Server > Servertypen > WebSphere-Anwendungsserver**.

9. Wählen Sie im Bereich **Anwendungsserver** den Eintrag für **Ihren Servernamen** aus.
10. Wählen Sie im Bereich **Serverinfrastruktur** den Eintrag **Angepasste Eigenschaften** aus.
11. Aktualisieren Sie die Datei `splicer.properties` mit den Werten, die Sie in „Vorherige Konfigurationseinstellungen in der Administrationskonsole von WebSphere Application Server notieren" auf Seite 280 abgerufen haben.
12. Starten Sie die Anwendungsserverprozesse erneut.

Anmerkung: Ändern Sie die Datei `splicer.properties` auf Deployment-Manager-Ebene so, dass die Eigenschaften mit dem Node Agent synchronisiert sind. Wenn Sie die Datei `splicer.properties` auf Knotenebene aktualisieren, überschreibt der Deployment Manager die Datei `splicer.properties` bei der nächsten Synchronisation.

Anmerkung: Wenn Sie zur Verwaltung der Datenbanksitzungen und dann zur Sitzungsverwaltung von WebSphere eXtreme Scale zurückkehren, wird die Datei `splicer.properties` erneut erstellt, sodass alle vorgenommenen Änderungen überschrieben werden. Eine Beschreibung des Synchronisationsprozesses (vom Deployment Manager über die Anmerkungen bis hin zu den Änderungen) finden Sie unter System Management File Synchronization.

Ergebnisse

Sie haben Ihre vorherigen Konfigurationseinstellungen für die Speicher-zu-Speicher- oder Datenbanksitzungsverwaltung mit der Sitzungsverwaltung von WebSphere eXtreme Scale geändert.

Szenario: WebSphere eXtreme Scale als dynamischen Cache-Provider verwenden

WebSphere Application Server stellt einen dynamischen Cache-Service für implementierte Java-EE-Anwendungen bereit. Dieser Service wird verwendet, um Geschäftsdaten, generierte HTML, Befehlsausgaben usw. zwischenzuspeichern. Ursprünglich war der einzige Provider für den dynamischen Cache-Service der dynamische Standard-Cache-Provider, der in WebSphere Application Server integriert ist. Jetzt können Kunden auch WebSphere eXtreme Scale als Cache-Provider für eine Cacheinstanz angeben. Auf diese Weise können Anwendungen, die den dynamischen Cache-Service verwenden, die Features und die Durchsatzkapazität von WebSphere eXtreme Scale nutzen.

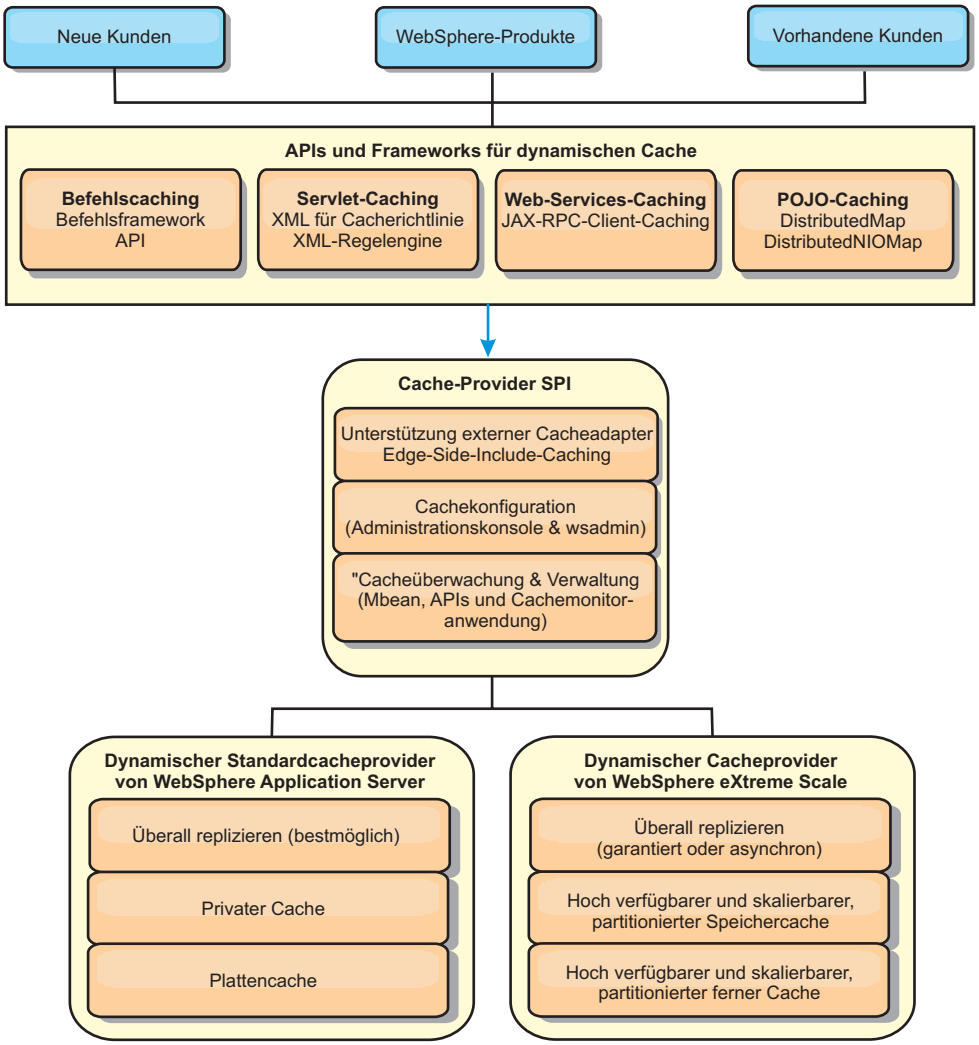
Informationen zu diesem Vorgang

Übersicht über den dynamischen Cache-Provider

WebSphere Application Server stellt einen dynamischen Cache-Service bereit, der implementierten Java EE zur Verfügung steht. Dieser Service wird verwendet, um Daten wie die Ausgabe von Servlets, JSPs oder Befehlen sowie Objektdaten, die über das Programm in einer Unternehmensanwendung angegeben werden, mit DistributedMap-APIs zwischenzuspeichern.

Anfänglich war der einzige Serviceprovider für den dynamischen Cache-Service die in WebSphere Application Server integrierte Standardengine für dynamischen Cache. Jetzt können Kunden auch WebSphere eXtreme Scale als Cache-Provider für eine Cacheinstanz angeben. Wenn Sie diese Funktion konfigurieren, können Sie

Anwendungen, die den dynamischen Cache-Service verwenden, ermöglichen, die Features und Leistungsfunktionen von WebSphere eXtreme Scale zu verwenden.



Sie können den dynamischen Cache-Provider, wie im Abschnitt Standardinstanz des dynamischen Caches (baseCache) konfigurieren beschrieben, installieren und konfigurieren.

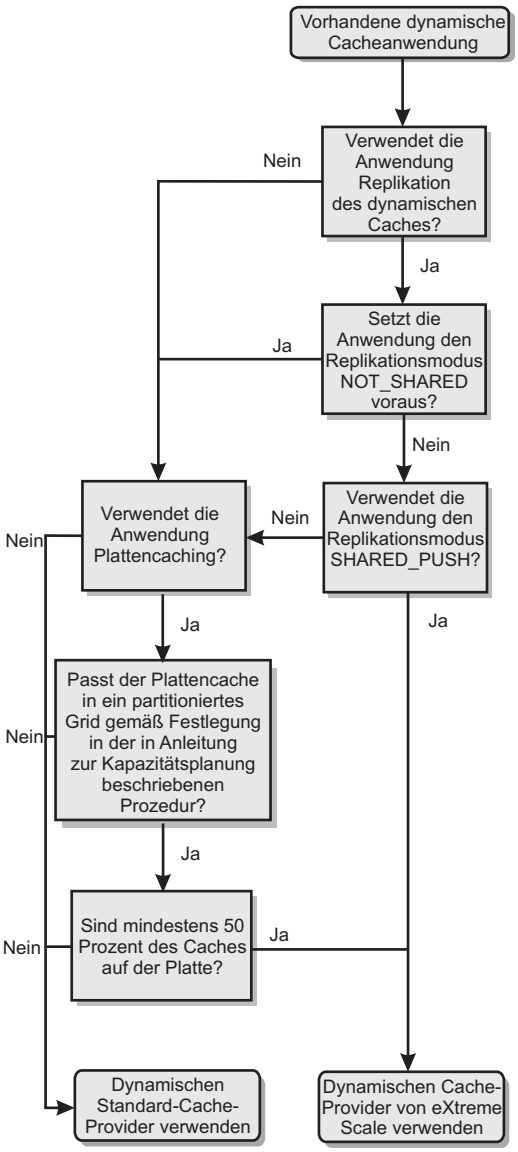
Einsatz von WebSphere eXtreme Scale festlegen

Die verfügbaren Features in WebSphere eXtreme Scale erweitern die verteilten Funktionen des dynamischen Cache-Service weit über das hinaus, was der Standardprovider für dynamischen Cache und der Datenreplikationsservice bieten. Mit eXtreme Scale können Sie Caches erstellen, die wirklich auf mehrere Server verteilt, anstatt nur repliziert und unter den Servern synchronisiert werden. Außerdem sind die Caches von eXtreme Scale transaktionsorientiert und hoch verfügbar, wodurch sichergestellt wird, dass jeder Server denselben Inhalt für den dynamischen Cacheservice sieht. WebSphere eXtreme Scale bietet eine höhere Servicequalität für die Cachereplikation über den Datenreplikationsservice.

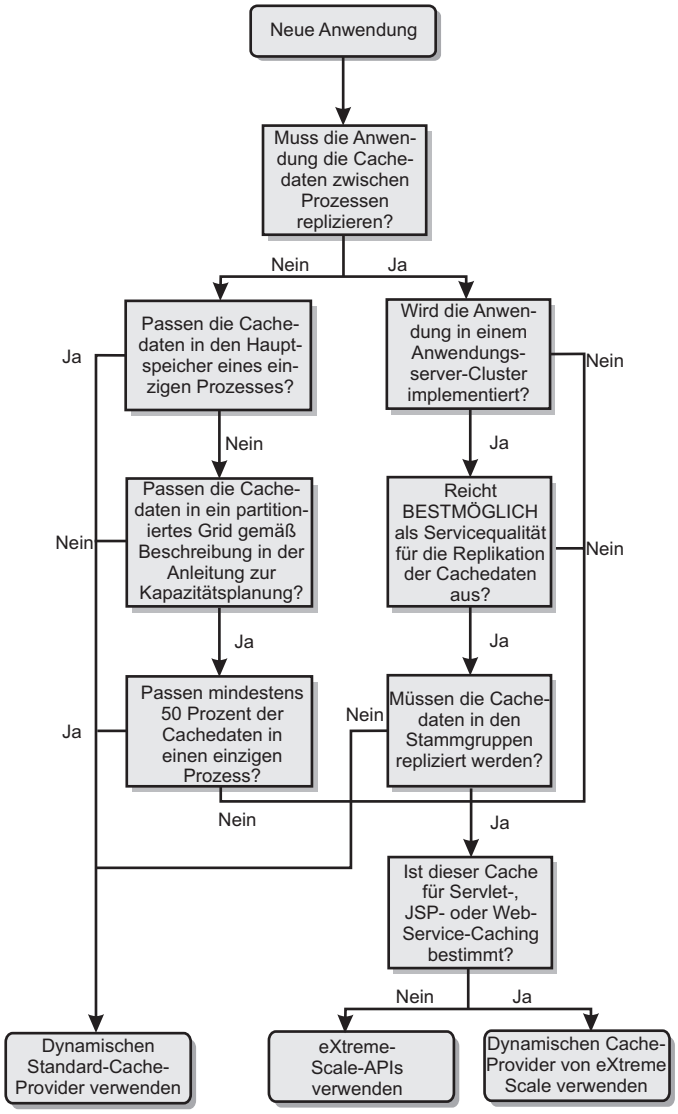
Diese Vorteile bedeuten jedoch nicht, dass der dynamische Cache-Provider von eXtreme Scale die richtige Wahl für jede Anwendung ist. Verwenden Sie die folgen-

den Entscheidungsbäume und Featurevergleichsmatrizes, um festzustellen, welche Technologie sich am besten für Ihre Anwendung eignet.

Entscheidungsbaum für die Migration vorhandener Anwendungen mit dynamischem Cache



Entscheidungsbaum für die Auswahl eines Cache-Providers für neue Anwendungen



Featurevergleich

Tabelle 13. Featurevergleich

Cache-Features	Standardprovider	eXtreme-Scale-Provider	eXtreme-Scale-API
Lokales Speicher-Caching	Ja	Über die Funktion für nahen Cache	Über die Funktion für nahen Cache
Verteiltes Caching	Über den Datenreplikationsservice	Ja	Ja
Linear skalierbar	Nein	Ja	Ja
Zuverlässige Replikation (synchron)	Nein	Ja	Ja
Plattenüberlauf	Ja	Nicht zutreffend	Nicht zutreffend

Tabelle 13. Featurevergleich (Forts.)

Cache-Features	Standardprovider	eXtreme-Scale-Provider	eXtreme-Scale-API
Bereinigung	LRU-/TTL-/Heapbasiert	LRU/TTL (pro Partition)	LRU/TTL (pro Partition)
Invalidierung	Ja	Ja	Ja
Beziehungen	Abhängigkeit/Vorlagen-ID-Beziehungen	Ja	Nein (keine anderen Beziehungen möglich)
Suchoperationen ohne Schlüssel	Nein	Nein	Über Abfrage und Index
Back-End-Integration	Nein	Nein	Über Ladeprogramme
Transaktionsorientiert	Nein	Ja	Ja
Schlüsselbasierter Speicher	Ja	Ja	Ja
Ereignisse und Listener	Ja	Nein	Ja
Integration von WebSphere Application Server	Nur einzelne Zelle	Mehrere Zellen	Zellenunabhängig
Unterstützung von Java Standard Edition	Nein	Ja	Ja
Überwachung und Statistiken	Ja	Ja	Ja
Sicherheit	Ja	Ja	Ja

Eine ausführlichere Beschreibung zur Funktionsweise der verteilten Caches in eXtreme Scale finden Sie in den Informationen zu Implementierungskonfigurationen in der Veröffentlichung *Verwaltung*.

Anmerkung: In einem verteilten eXtreme-Scale-Cache können nur Einträge gespeichert werden, bei denen sowohl der Schlüssel als auch der Wert die Schnittstelle "java.io.Serializable" implementieren.

Topologietypen

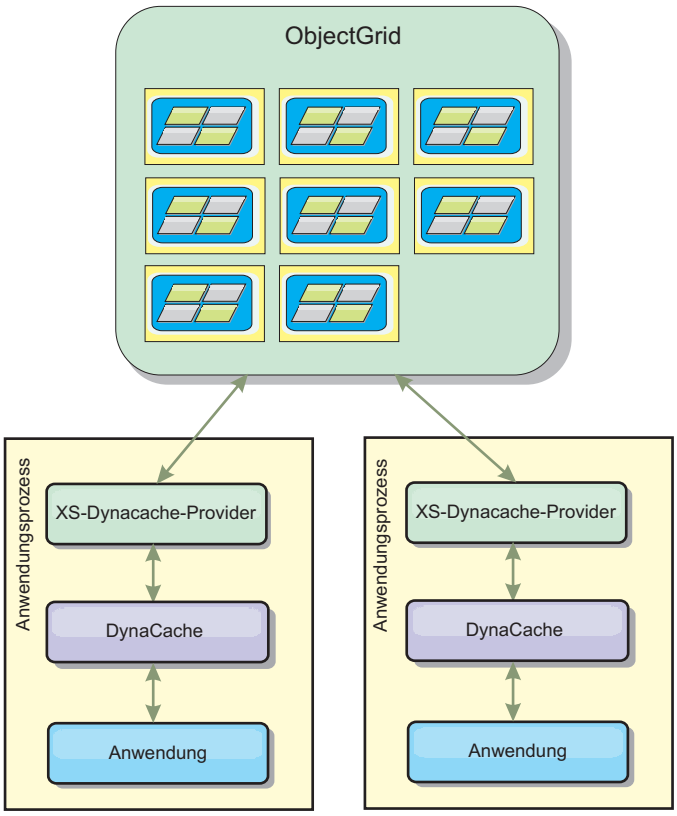
Veraltet: 8.6+ Die Topologietypen "local", "embedded" und "embedded-partitioned" sind veraltet.

Ein dynamischer Cache-Service, der mit eXtreme Scale als Provider erstellt wurde, kann in einer fernen Topologie implementiert werden.

Ferne Topologie

Bei der fernen Topologie ist kein Plattencache erforderlich. Alle Cachedaten werden außerhalb der Prozesse von WebSphere Application Server gespeichert. WebSphere eXtreme Scale unterstützt eigenständige Containerprozesse für Cachedaten. Diese Containerprozesse haben geringere Kosten als ein Prozess von WebSphere Application Server und sind auch nicht auf die Verwendung einer bestimmten Java Virtual Machine (JVM) beschränkt. Die Daten für einen dynamischen Cacheservice, auf

den mit einem 32-Bit-Prozess von WebSphere Application Server zugegriffen wird, könnte sich beispielsweise in einem eXtreme-Scale-Containerprozess befinden, der in einer 64-Bit-JVM ausgeführt wird. Dies ermöglicht Benutzern, die höhere Speicherkapazität von 64-Bit-Prozessen für das Caching zu nutzen, ohne das zusätzliche Kosten für den 64-Bit-Betrieb bei den Anwendungsserverprozessen anfallen. Die ferne Topologie wird in der folgenden Abbildung veranschaulicht:



Funktionale Unterschiede zwischen der dynamischen Cache-Engine und eXtreme Scale

Dem Benutzer sollten eigentlich keine funktionalen Unterschiede zwischen den beiden Caches auffallen, abgesehen davon, dass von WebSphere eXtreme Scale unterstützte Caches keine Auslagerung auf die Platte oder Statistiken und Operationen unterstützen, die sich auf die Größe des Caches im Speicher beziehen.

Die von den meisten Aufrufen der API "Dynamic Cache" zurückgegebenen Ergebnisse unterscheiden sich kaum, unabhängig davon, ob der Kunde den dynamischen Standard-Cache-Provider oder den Cache-Provider von eXtreme Scale verwendet. Für einige Operationen kann das Verhalten der dynamischen Cache-Engine nicht mit eXtreme Scale emuliert werden.

Statistiken zum dynamischen Cache

Statistische Daten für einen dynamischen Cache von WebSphere eXtreme Scale können mit den Überwachungstools von eXtreme Scale abgerufen werden. Weitere Informationen finden Sie im Abschnitt Überwachung.

MBean-Aufrufe

Der dynamische Cache-Provider von WebSphere eXtreme Scale unterstützt kein Platten-Caching. Alle MBean-Aufrufe, die sich auf Platten-Caching beziehen, funktionieren nicht.

Zuordnung einer Replikationsrichtlinie für den dynamischen Cache

Die ferne Topologie des dynamischen Cache-Providers von eXtreme Scale unterstützt eine Replikationsrichtlinie, die den Richtlinien SHARED_PULL und SHARED_PUSH_PULL am ehesten entspricht (und die Terminologie verwendet, die vom dynamischen Cache-Provider von WebSphere Application Server verwendet wird). In einem dynamischen Cache von eXtreme Scale ist der verteilte Status des Caches auf allen Servern vollkommen konsistent.

8.6+ Invalidierung mithilfe eines globalen Index

Sie können einen globalen Index verwenden, um die Invalidierungseffizienz in großen partitionierten Umgebungen zu verbessern, z. B. in Umgebungen mit mehr als 40 Partitionen. Ohne das globale Indexfeature müssen die Vorlage für dynamische Caches und die Verarbeitung der Abhängigkeitsinvalidierung Anforderungen ferner Agenten an alle Partitionen senden, was zu Leistungseinbußen führt. Wenn Sie einen globalen Index konfigurieren, werden Invalidierungsagenten nur an Partitionen gesendet, die Cacheeinträge enthalten, die sich auf die Vorlage oder die Abhängigkeits-ID beziehen. Die potenzielle Leistungsverbesserung ist in Umgebungen mit sehr vielen konfigurierten Partitionen höher. Sie können einen globalen Index mit den Indizes für Abhängigkeits-IDs und Vorlagen-IDs konfigurieren, die in den Beispiel-ObjectGrid-XML-Deskriptordateien für den dynamischen Cache verfügbar sind. Weitere Informationen finden Sie unter „Unternehmensdatengrid für dynamisches Caching in einer eigenständigen Umgebung konfigurieren” auf Seite 292.

Sicherheit

Wenn ein Cache in einer fernen Topologie ausgeführt wird, kann ein eigenständiger extreme-Scale-Client eine Verbindung zum Cache herstellen und den Inhalt der dynamischen Cacheinstanz beeinflussen. Deshalb ist es wichtig, dass sich Server von WebSphere eXtreme Scale, die die dynamischen Cacheinstanzen enthalten, in einem internen Netz hinter der so genannten Netz-DMZ (Demilitarized Zone) befinden.

Lesen Sie in der Dokumentation zu eXtreme Scale die Informationen zur „Übersicht über die Sicherheit” auf Seite 148, wenn SSL- oder Clientauthentifizierung erforderlich ist.

Naher Cache

Eine dynamische Cacheinstanz kann so konfiguriert werden, dass ein naher Cache erstellt wird, der sich lokal in der JVM des Anwendungsservers befindet und ein Subset der Einträge enthält, die in der fernen dynamischen Cacheinstanz enthalten sind. Sie können die nahe Cacheinstanz mit einer Datei namens "dynacache-nearCache-ObjectGrid.xml" konfigurieren. Weitere Informationen finden Sie unter „Unternehmensdatengrid für dynamisches Caching in einer eigenständigen Umgebung konfigurieren” auf Seite 292. Außerdem gibt es angepasste Eigenschaften für die Optimierung des nahen Caches. Weitere Informationen hierzu finden Sie unter

Angepasste Eigenschaften des dynamischen Caches.

Weitere Informationen

- Redbook zum dynamischen Cache
- Dokumentation zum dynamischen Cache
  - WebSphere Application Server 7.0
- Dokumentation zum Datenreplikationsservice
  - WebSphere Application Server 7.0

Umgebungskapazität planen

Wenn Sie eine anfängliche Dateigruppengröße und eine geplante Dateigruppengröße haben, können Sie die für die Ausführung von WebSphere eXtreme Scale erforderliche Kapazität planen. Mit diesen Planungsübungen können Sie WebSphere eXtreme Scale effizient für künftige Änderungen planen und die Elastizität des Datengrids maximieren. Diese Möglichkeit haben Sie in einem anderen Szenario, wie z. B. einer speicherinternen Datenbank oder einem anderen Typ von Datenbank, nicht.

Unternehmensdatengrid für dynamisches Caching in einer eigenständigen Umgebung konfigurieren

Kopieren und ändern Sie diese Implementierungs- und ObjectGrid-Deskriptordateien nacheinander, um ein Unternehmensgrid für dynamisches Caching zu konfigurieren. Diese Dateien werden zum Starten eines Unternehmensdatengrids verwendet.

Informationen zu diesem Vorgang

Wenn WebSphere eXtreme Scale als Provider für eine Instanz des dynamischen Caches von WebSphere Application Server angegeben wird, werden die Server von WebSphere eXtreme Scale in einer eigenständigen Umgebung oder in einer Umgebung von WebSphere Application Server gestartet. Weitere Informationen hierzu finden Sie unter Eigenständige Server starten und stoppen. Dieser Prozess setzt die Verwendung von Implementierungs- und ObjectGrid-Deskriptordateien voraus, die zum Konfigurieren des Unternehmensdatengrids verwendet werden. Dynamisches Caching erfordert eine spezielle Konfiguration. Deshalb werden mehrere XML-Dateien mit WebSphere eXtreme Scale bereitgestellt, die kopiert, (bei Bedarf) geändert und zum Starten des Unternehmensdatengrids verwendet werden müssen. Diese Dateien können unverändert verwendet werden, aber Änderungen an diesen Dateien sind vorbehalten. Deshalb sollten Sie diese Dateien an eine separate Position kopieren, bevor sie geändert oder verwendet werden.

Anmerkung: Abhängig davon, wie Sie WebSphere eXtreme Scale installiert haben, befinden sich diese Dateien entweder im Verzeichnis `WAS-Stammverzeichnis/optionalLibraries/ObjectGrid/dynacache/etc` (für Installationen mit WebSphere Application Server) oder im Verzeichnis `WXS-Installationsstammverzeichnis/ObjectGrid/dynacache/etc` (für Installationen in einer eigenständigen Umgebung).

Wichtig: Es wird dringend empfohlen, diese Dateien an eine andere Position zu kopieren, bevor sie bearbeitet oder verwendet werden.

Deskriptordatei für den dynamischen Cache (dynacache-remote-deployment.xml)

Diese Datei ist die Implementierungsdeskriptordatei zum Starten eines Container-Servers für dynamisches Caching. Weitere Informationen finden

Sie unter XML-Deskriptordatei für Implementierungsrichtlinie. Obwohl diese Datei unverändert verwendet werden kann, werden die folgenden Elemente oder Attribute gelegentlich geändert bzw. sind von entscheidender Bedeutung:

- **mapSet name und map ref**

  Das Attribut **name** in "mapSet" und der definierte Wert für "map ref" entsprechen nicht direkt dem Namen der Instanz des dynamischen Caches, der für WebSphere Application Server definiert ist, und werden gewöhnlich nicht geändert. Werden diese Werte jedoch geändert, müssen die entsprechenden angepassten Eigenschaften der Konfiguration der Instanz des dynamischen Caches hinzugefügt werden. Weitere Informationen finden Sie unter Instanz des dynamischen Caches mit angepassten Eigenschaften anpassen..

- **numberOfPartitions**

  Dieses Attribut kann geändert werden, um die richtige Anzahl von Partitionen für Ihre Konfiguration anzugeben. Weitere Informationen finden Sie unter „Umgebungskapazität planen” auf Seite 292.

- **maxAsyncReplicas**

  Dieses Attribut kann geändert werden. Ein dynamischer Cache wird gewöhnlich in einem Nebencachemodell mit einer Datenbank oder einer anderen Quelle als Aufzeichnungssystem für die Daten verwendet. Deshalb wird beim Definieren des Werts OPTIMISTIC oder NONE für diese Einstellung die Verarbeitung des nahen Caches ausgelöst, wenn der Transporttyp "eXtreme I/O (XIO)" verwendet wird, und aufgrund der Kompromisse, die bezüglich des Plattenspeicherplatzes und der Leistung erforderlich sind, um die Daten hoch verfügbar zu machen, wird von der Verwendung der Replikation abgeraten. In manchen Fällen ist die hohe Verfügbarkeit jedoch wichtig.

- **numInitialContainers**

  Dieses Attribut muss auf die Anzahl der Container für den Erststart des Unternehmensdatensgrids gesetzt werden. Die Definition des richtigen Werts für dieses Attribut hilft bei der Platzierung und Verteilung von Partitionen im Datengrid.

ObjectGrid-XML-Deskriptordatei für den dynamischen Cache (dynacache-remote-objectgrid.xml)

Diese Datei ist die empfohlene ObjectGrid-Deskriptordatei zum Starten eines Container-Servers für dynamisches Caching. Weitere Informationen finden Sie unter ObjectGrid-XML-Deskriptordatei. Sie ist für die Ausführung mit dem Transporttyp "eXtreme I/O" (XIO) unter Verwendung von eXtreme Data Formatting (XDF) konfiguriert. Außerdem sind die Indizes für die Abhängigkeits-ID und die Vorlagen-ID für die Verwendung eines globalen Index konfiguriert, was die Invalidierungsleistung verbessert. Obwohl diese Datei unverändert verwendet werden kann, werden die folgenden Elemente oder Attribute gelegentlich geändert bzw. sind von entscheidender Bedeutung.

- **objectGrid name und backingMap name**

  Die **name**-Attribute in den Elementen "objectGrid" und "backingMap" entsprechen nicht direkt dem Namen der Instanz des dynamischen Caches, der für die Cacheinstanz von WebSphere Application Server konfiguriert ist, und müssen gewöhnlich nicht geändert werden. Werden diese Attribute jedoch geändert, müssen die entsprechenden angepassten Eigenschaften der Konfiguration der Instanz des dynamischen Caches hinzu-

gefügt werden. Weitere Informationen finden Sie unter Instanz des dynamischen Caches mit angepassten Eigenschaften anpassen..

- **copyMode**

  Setzen Sie dieses Attribut auf COPY_TO_BYTES. Dieser Wert aktiviert eXtreme Data Format (XDF), wenn der Transporttyp "eXtreme I/O" (XIO) verwendet wird. Wenn Sie einen anderen Wert als "copyMode" festlegen, wird XDF inaktiviert, und Sie müssen das Kommentarzeichen für die ObjectTransformer-Plug-in-Bean entfernen.

- **lockStrategy**

  Setzen Sie dieses Attribut auf PESSIMISTIC. Wenn Sie das Attribut auf OPTIMISTIC oder NONE setzen, wird die Verarbeitung des nahen Caches ausgelöst. Außerdem müssen zugehörige Eigenschaften in der Datei "dynamic-nearcache-objectgrid.xml" definiert sein.

- **backingMapPluginCollections**

  Dieses Element ist erforderlich. Die untergeordneten Elemente "Evictor plug-in" und "MapIndex plug-in" sind für dynamisches Caching beide erforderlich und dürfen nicht entfernt werden.

- **GlobalIndexEnabled**

  DEPENDENCY_ID_INDEX und TEMPLATE_INDEX enthalten beide eine Eigenschaft "GlobalIndexEnabled", die auf "true" gesetzt ist. Wenn Sie diese Einstellung auf "false" setzen, wird das globale Indexfeature für diese Indizes inaktiviert. Es wird empfohlen, diese globalen Indizes aktiviert zu lassen, es sei denn, Sie arbeiten mit einer geringen Anzahl von Partitionen, z. B. mit weniger 40 Partitionen.

- **objectTransformer**

  Da diese ObjectGrid-Deskriptordatei für die Ausführung in eXtreme Data Format (XDF) bestimmt ist, ist sie auf Kommentar gesetzt. Wenn Sie XDF (durch Änderung des copyMode-Werts) inaktivieren möchten, müssen Sie das Kommentarzeichen für dieses Plug-in entfernen.

ObjectGrid-Deskriptordatei für den dynamischen nahen Cache (dynacache-nearCache-ObjectGrid.xml)

Diese Datei ist die empfohlene ObjectGrid-Deskriptordatei für das Starten von Grid-Container-Servern für dynamisches Caching, wenn ein naher Cache erwünscht ist. Sie ist für die Ausführung mit dem Transporttyp "eXtreme I/O" (XIO) unter Verwendung von eXtreme Data Formatting (XDF) konfiguriert. Außerdem sind die Indizes für die Abhängigkeits-ID und die Vorlagen-ID für die Verwendung eines globalen Index konfiguriert, was die Invalidierungsleistung verbessert. Die nahe Cachefunktion für dynamisches Caching setzt die Verwendung des Transporttyps "eXtreme I/O (XIO)" voraus.

Obwohl diese Datei unverändert verwendet werden kann, werden die folgenden Elemente oder Attribute gelegentlich geändert bzw. sind von entscheidender Bedeutung:

- **objectGrid name und backingMap name**

  Diese Werte in dieser Datei entsprechen dem Namen der dynamischen Cacheinstanz, die für die Cacheinstanz von WebSphere Application Server konfiguriert ist, nicht direkt und müssen gewöhnlich nicht geändert werden. Werden diese Werte jedoch geändert, müssen die entsprechenden angepassten Eigenschaften der Konfiguration der Instanz des dynamischen Caches hinzugefügt werden.

- **lockStrategy**

Diese Eigenschaft muss auf OPTIMISTIC oder NONE gesetzt werden, um einen nahen Cache zu aktivieren. Der nahe Cache wird von keiner anderen Sperrstrategie unterstützt.

- **nearCacheInvalidationEnabled**

  Diese Eigenschaft muss auf "true" gesetzt werden, um einen nahen Cache für dynamisches Caching zu aktivieren. Dieses Feature verwendet Publish/Subscribe, um Invalidierungen aus dem entfernten Cache in die nahen Cacheinstanzen zu übertragen und sie damit zu synchronisieren.

- **nearCacheLastAccessTTLSyncEnabled**

  Diese Eigenschaft muss auf "true" gesetzt werden, um einen nahen Cache für dynamisches Caching zu aktivieren. Dieses Feature verwendet Publish/Subscribe, um TTL-Bereinigungen aus dem entfernten Cache in die nahen Cacheinstanzen zu übertragen und sie damit zu synchronisieren.

- **copyMode**

  Diese BackingMap-Eigenschaft ist auf "COPY_TO_BYTES" gesetzt. Dieser Wert aktiviert eXtreme Data Format (XDF), wenn der Transporttyp "eXtreme I/O" (XIO) verwendet wird. Wenn Sie einen anderen Wert als "copyMode" festlegen, wird XDF inaktiviert, und Sie müssen das Kommentarzeichen für die ObjectTransformer-Plug-in-Bean entfernen.

- **backingMapPluginCollections**

  MapIndexPlugins und Evictor sind verbindliche Einträge für dynamisches Caching und dürfen nicht entfernt werden.

- **GlobalIndexEnabled**

  DEPENDENCY_ID_INDEX und TEMPLATE_INDEX enthalten beide eine Eigenschaft "GlobalIndexEnabled", die auf "true" gesetzt ist. Wenn Sie diese Einstellung auf "false" setzen, wird das globale Indexfeature für diese Indizes inaktiviert. Es wird empfohlen(diese globalen Indizes aktiviert zu lassen, es sei denn, Sie arbeiten mit einer geringen Anzahl von Partitionen, z. B. mit weniger 40 Partitionen.

- **ObjectTransformer**

  Da diese Datei für die Ausführung in eXtreme Data Format (XDF) bestimmt ist, ist dieses Plug-in auf Kommentar gesetzt. Wenn XDF (durch Ändern des copyMode-Werts) inaktiviert werden soll, muss das Kommentarzeichen für dieses Plug-in entfernt werden.

Traditionelle ObjectGrid-Deskriptordatei für dynamisches Caching (dynacache-legacy85-ObjectGrid.xml)

Diese Datei ist die empfohlene ObjectGrid-Deskriptordatei für das Starten eines Container-Servers für dynamisches Caching, wenn Sie einen nahen Cache ausgewählt haben. Obwohl diese Datei unverändert verwendet werden kann, werden die folgenden Elemente oder Attribute gelegentlich geändert bzw. sind von entscheidender Bedeutung:

- **objectGrid name und backingMap name**

  Diese Werte in dieser Datei entsprechen dem Namen der dynamischen Cacheinstanz, die für die Cacheinstanz von WebSphere Application Server konfiguriert ist, nicht direkt und müssen gewöhnlich nicht geändert werden. Werden diese Werte jedoch geändert, müssen die entsprechenden angepassten Eigenschaften der Konfiguration der Instanz des dynamischen Caches hinzugefügt werden.

- **copyMode**

Diese BackingMap-Eigenschaft ist auf "COPY_ON_READ_AND_COMMIT" gesetzt. Dieser Wert sollte nicht geändert werden.

- **lockStrategy**

  Diese BackingMap-Eigenschaft ist auf "PESSIMISTIC" gesetzt. Dieser Wert sollte nicht geändert werden.

- **backingMapPluginCollections**

  MapIndexPlugins, Evictor und Object Transformer sind verbindliche Einträge für dynamisches Caching und dürfen nicht entfernt werden.

Unternehmensdatengrid für dynamisches Caching mit einem Liberty-Profil konfigurieren

Ein Liberty-Profilserver kann ein Datengrid hosten, in dem Daten für Anwendungen zwischengespeichert werden, in denen der dynamische Cache aktiviert ist.

Vorbereitende Schritte

- Installieren Sie das Liberty-Profil. Weitere Informationen finden Sie unter Liberty-Profil installieren.
- Erstellt eine Anwendung, die den dynamischen Cache verwendet. Weitere Informationen finden Sie unter Standardinstanz des dynamischen Caches (baseCache) konfigurieren.

Informationen zu diesem Vorgang

Das Liberty-Profil hostet das Datengrid, das Anwendungen unterstützt, in denen der dynamische Cache aktiviert ist. Das bedeutet, dass die Anwendung in einer traditionellen Installation von WebSphere Application Server ausgeführt wird. Damit diese Anwendungen von der eXtreme-Scale-Laufzeitumgebung zwischengespeichert werden, müssen Sie WebSphere Application Server so konfigurieren, dass die Katalogservicedomäne und die Servereigenschaften verwendet werden, die Sie im Liberty-Profil angeben.

Vorgehensweise

1. Aktivieren Sie das Feature für den dynamischen Cache in WebSphere eXtreme Scale.
   a. Fügen Sie das Feature für den dynamischen Cache der Datei `server.xml` des Liberty-Profils hinzu. Ihre Datei `server.xml` gleicht beispielsweise der folgenden Codezeilengruppe:

   ```
   <featureManager>
   <feature>eXtremeScale.server-1.1</feature>
   <feature>eXtremeScale.dynacacheGrid-1.1</feature>
   </featureManager>
   ```

2. Optional: Legen Sie Eigenschaften im Element `xsDynacacheGrid` in der Datei `server.xml` fest. Sie können die folgenden Eigenschaften ändern, aber es wird empfohlen, die Standardwerte zu akzeptieren.

   `globalIndexDisabled`
   Die Invalidierung mithilfe eines globalen Index verbessert die Invalidierungsleistung in großen, partitionierten Umgebungen, z. B. in Umgebungen mit mehr als 40 Partitionen. Weitere Informationen finden Sie unter „Dateninvalidierung” auf Seite 70. Standardwert: `false`

   `objectGridName`
   Eine Zeichenfolge, die den Namen des Datengrids angibt. Standardwert: `DYNACACHE_REMOTE`

ojectGridTxTimeout
Gibt die Zeit in Sekunden an, die einer Transaktion für die Ausführung zugestanden wird. Wenn eine Transaktion nicht innerhalb dieser Zeit abgeschlossen wird, wird sie für Rollback markiert, und es wird eine Ausnahme des Typs `TransactionTimeoutException` ausgelöst. Standardwert: `30` (in Sekunden)

backingMapLockStrategy
Gibt an, ob der interne Sperrenmanager verwendet wird, wenn eine Transaktion auf einen Mapeintrag zugreift. Setzen Sie dieses Attribut auf einen der folgenden drei Werte: `OPTIMISTIC`, `PESSIMISTIC` oder `NONE`. Standardwert: `PESSIMISTIC`

backingMapCopyMode
Gibt an, ob eine get-Operation eines Eintrags in der BackingMap-Instanz den tatsächlichen Wert, eine Kopie des Werts oder einen Proxy für den Wert zurückgibt. Wenn Sie eXtreme Data Format (XDF) verwenden, sodass Java und .NET auf dasselbe Datengrid zugreifen können, ist der Standard- und erforderliche Kopiermodus "COPY_TO_BYTES". Andernfalls wird der Kopiermodus "COPY_ON_READ_AND_COMMIT" verwendet. Setzen Sie das Attribut "CopyMode" auf einen der folgenden fünf Werte:

COPY_ON_READ_AND_COMMIT
Der Standardwert ist `COPY_ON_READ_AND_COMMIT`. Setzen Sie das Attribut auf `COPY_ON_READ_AND_COMMIT`, um sicherzustellen, dass eine Anwendung keine Referenz auf das Wertobjekt verwendet, das in der BackingMap-Instanz enthalten ist. Stattdessen arbeitet die Anwendung immer mit einer Kopie des Werts, der in der BackingMap-Instanz enthalten ist. (Optional)

COPY_ON_READ
Setzen Sie das Attribut auf `COPY_ON_READ`, um eine im Vergleich mit dem Wert `COPY_ON_READ_AND_COMMIT` höhere Leistung zu erzielen, indem der Kopiervorgang bei der Festschreibung einer Transaktion umgangen wird. Zur Gewährleistung der Integrität der BackingMap-Daten stellt die Anwendung sicher, dass jede Referenz auf einen Eintrag nach der Festschreibung der Transaktion gelöscht wird. Durch die Festlegung dieses Werts wird eine Methode "ObjectMap.get" aufgerufen, die eine Kopie des Werts an Stelle einer Referenz auf den Wert zurückgibt, was sicherstellt, dass Änderungen, die von der Anwendung am Wert vorgenommen werden, erst dann für das BackingMap-Element wirksam werden, wenn die Transaktion festgeschrieben wird.

COPY_ON_WRITE
Setzen Sie das Attribut auf `COPY_ON_WRITE`, um eine im Vergleich mit dem Wert `COPY_ON_READ_AND_COMMIT` bessere Leistung zu erzielen, in dem der Kopiervorgang beim ersten Aufruf der Methode "ObjectMap.get" für einen bestimmten Schlüssel durch eine Transaktion umgangen wird. Stattdessen gibt die Methode "ObjectMap.get" einen Proxy für den Wert an Stelle einer Referenz auf das Wertobjekt zurück. Der Proxy stellt sicher, dass keine Kopie des Werts erstellt wird, sofern die Anwendung nicht eine Methode "set" für die Wertschnittstelle aufruft.

NO_COPY
Setzen Sie das Attribut auf den Wert `NO_COPY`, um einer Anwendung zu ermöglichen, ein Wertobjekt, das mit einer Methode "ObjectMap.get" abgerufen wird, nie zu ändern, um so Leistungsver-

besserungen zu erzielen. Setzen Sie das Attribut für Maps, die Entitäten der EntityManager-API zugeordnet sind, auf NO_COPY.

COPY_TO_BYTES
Setzen Sie das Attribut auf COPY_TO_BYTES, um den Speicherbedarf für komplexe Objekttypen zu verringern und die Leistung zu verbessern, wenn das Kopieren des Objekts durch Serialisierung vorgenommen werden soll. Wenn ein Objekt nicht klonbar ist oder kein angepasstes ObjectTransformer-Plug-in mit einer effizienten Methode "copyValue" bereitgestellt wird, wird der Standardkopiermechanismus verwendet, um das Objekt zu serialisieren und zu dekomprimieren und die Kopie zu erstellen. Bei der Einstellung COPY_TO_BYTES wird die Dekomprimierung bei einer reinen Leseoperation und die Serialisierung bei einer reinen Festschreibungsoperation durchgeführt.

Standardwert: COPY_ON_READ_AND_COMMIT

backingMapNearCacheEnabled
Setzen Sie diese Einstellung auf true, um den lokalen Client-Cache zu aktivieren. Zur Verwendung eines nahen Caches muss das Attribut **lockStrategy** auf NONE oder OPTIMISTIC gesetzt werden. Standardwert: false

mapSetNumberOfPartitions
Gibt die Anzahl der Partitionen für das Element "mapSet" an. Standardwert: 47

mapSetMinSyncReplicas
Gibt die Mindestanzahl synchroner Replikate für jede Partition im MapSet an. Es werden erst Shards verteilt, wenn die Domäne die Mindestanzahl synchroner Replikate unterstützen kann. Für die Unterstützung des minSyncReplicas-Werts benötigen Sie einen Container-Server mehr, als der **minSyncReplicas**-Wert angibt. Wenn die Anzahl synchroner Replikate unter den Wert von **minSyncReplicas** fällt, werden keine Schreiboperationen für diese Partition mehr zugelassen. Standardwert: 0

mapSetMaxSyncReplicas
Gibt die maximale Anzahl synchroner Replikate für jede Partition im MapSet an. Es werden keine weiteren synchronen Replikate für eine Partition verteilt, wenn eine Domäne diese Anzahl synchroner Replikate für diese bestimmte Partition erreicht. Das Hinzufügen von Container-Servern, die dieses ObjectGrid unterstützen, kann zu einer höheren Anzahl synchroner Replikate führen, wenn der Wert von **maxSyncReplicas** noch nicht erreicht ist. Standardwert: 0

mapSetNumInitialContainers
Gibt die Anzahl der Container-Server an, die erforderlich sind, bevor eine Erstverteilung für die Shards in diesem mapSet-Element vorgenommen wird. Mit diesem Attribut kann Verarbeitungszeit und Netzbandbreite eingespart werden, wenn eine ObjectGrid-Instanz über einen Kaltstart online gebracht wird. Standardwert: 1

mapSetDevelopmentMode
Mit diesem Attribut können Sie die Verteilung eines Shards in Relation zu dessen Peer-Shards beeinflussen. Wenn Sie das Attribut "developmentMode" auf false setzen, werden nie zwei Shards derselben Partition an dieselbe Maschine verteilt. Denn Sie das Attribut "developmentMode" auf true setzen, können Shards derselben Partition an dieselbe Maschine verteilt

werden. Für beide Fälle gilt jedoch, dass zwei Shards derselben Partition nie an denselben Container-Server verteilt werden. Standardwert: `false`

`mapSetReplicaReadEnabled`
Wenn Sie dieses Attribut auf `true` setzen, werden Leseanforderungen an die primären Shards und die Replikate einer Partition verteilt. Wenn Sie das Attribut "replicaReadEnabled" auf `false` setzen, werden Leseanforderungen nur an das primäre Shard weitergeleitet. Standardwert: `false`

.

3. Konfigurieren Sie WebSphere Application Server so, dass auf das Liberty-Profil verwiesen wird.

   Sie können Container und Webanwendungen von WebSphere eXtreme Scale, in denen der dynamische Cache aktiviert ist, mit einer Katalogservicedomäne verbinden, die in einer anderen Zelle von WebSphere Application Server oder als eigenständiger Prozess ausgeführt wird. Da über Fernzugriff konfigurierte Katalogserver in der Zelle nicht automatisch gestartet werden, müssen Sie diese manuell starten.

   Wenn Sie eine ferne Katalogservicedomäne konfigurieren, muss der Domänenname mit dem Domänennamen übereinstimmen, den Sie beim Starten der fernen Katalogserver angegeben haben. Standardmäßig wird als Katalogservicedomänenname für eigenständige Katalogserver `DefaultDomain` verwendet. Geben Sie mit dem Parameter **`-domain`** des Befehls **`startOgServer`** oder **`startXsServer`**, in einer Servereigenschaftendatei oder mit der integrierten Server-API den Namen einer Katalogservicedomäne an. Sie müssen jeden fernen Katalogserverprozess in der fernen Domäne mit demselben Domänennamen starten. Weitere Informationen zum Starten von Katalogservern finden Sie unter Eigenständigen Katalogservice starten, der den ORB Transport verwendet.

Instanzen des dynamischen Caches konfigurieren

Der dynamische Cache-Service von WebSphere unterstützt die Erstellung einer Standardcacheinstanz (Basiscache) sowie die Erstellung weiterer Servlet- und Objektcacheinstanzen.

Informationen zu diesem Vorgang

Die Standardcacheinstanz (Basiscache) war ursprünglich die einzige Instanz des dynamischen Caches, die von WebSphere Application Server unterstützt wurde, und ist momentan die sofort einsatzfähige Instanz des dynamischen Caches, die von WebSphere Commerce Suite verwendet wird. Die zusätzlichen Servlet- und Objektcacheinstanzen wurden in neueren Releases von WebSphere Application Server hinzugefügt und werden in einem separaten Abschnitt "Cacheinstanz" in der WebSphere-Administrationskonsole konfiguriert.

Kapitel 4. Beispiele

Es sind mehrere Lernprogramme und Beispiele für WebSphere eXtreme Scale verfügbar.

Beispiele

In den folgenden Abschnitten werden die Schlüsselfunktionen von WebSphere eXtreme Scale veranschaulicht.

- Beispiel für die DataGrid-APIs
- Lokale Implementierungen konfigurieren

Community-Beispiele

Die folgenden Beispiele aus der WebSphere eXtreme ScaleBeispielsammlung veranschaulichen, wie WebSphere eXtreme Scale in verschiedenen Umgebungen für die Bereitstellung verschiedener Features des Produkts verwendet wird.

Tabelle 14. Verfügbare Beispiele

Beispiel	
Asynchronous Service Framework	Das Beispiel "Asynchronous Service Framework" stellt eine skalierbare und fehlertolerante Verarbeitungsstruktur für die asynchrone Verarbeitung von Nachrichten bereit. Weitere Informationen, einschließlich Anweisungen zum Herunterladen des Beispiels, finden Sie auf der Webseite Samples Gallery: Asynchronous Service Framework sample .
Binary JSON (BSON) Serializer	Dieses Beispiel veranschaulicht, wie eine Serialisierungsmethode für eXtreme Scale geschrieben und für die Verwendung mit eXtreme Scale konfiguriert wird. Die mit diesem Beispiel bereitgestellte Serialisierungsmethode verwendet Binary JSON (BSON), um Objekte zu beschreiben und zu serialisieren. Weitere Informationen, einschließlich Anweisungen zum Herunterladen des Beispiels, finden Sie auf der Webseite Samples Gallery: BSON serializer sample.
Client Authentication Security	Dieses Beispiel beschreibt, wie die Authentifizierung so konfiguriert wird, dass der Client gültige Berechtigungsnachweise bereitstellen muss, bevor der Server den Zugriff auf Grids erteilt. Weitere Informationen, einschließlich Informationen zum Download des Beispiels, finden Sie auf der Webseite Samples Gallery: Client authentication security.

Tabelle 14. Verfügbare Beispiele (Forts.)

Beispiel	
Creating Dynamic Maps	Dieses Beispiel veranschaulicht, wie Maps erstellt werden, nachdem das Grid initialisiert wurde. Für eXtreme Scale 7.0 und höher können Sie Schablonen für das Abrufen von Maps verwenden. Weitere Informationen, einschließlich Anweisungen zum Herunterladen des Beispiels, finden Sie auf der Webseite Samples Gallery: Creating dynamic maps after grid initialization.
Einführung in das Produkt	Die Einführungsbeispiele werden für eine schnelle Einführung in WebSphere eXtreme Scale und die Basisoperationen in einer Umgebung von WebSphere Application Server bereitgestellt. Weitere Informationen, einschließlich Informationen zum Herunterladen des Beispiels, finden Sie unter Samples Gallery: WebSphere eXtreme Scale getting started web application sample.
Einführung in Spring	Das Beispiel für die Einführung ín Spring wird für eine schnelle Einführung in die Integration des Spring-Frameworks bereitgestellt. Das Beispiel setzt sich aus Shell- und Stapelscripts zusammen, mit denen ein einfaches Grid ohne umfassende Anpassungen gestartet werden kann. Weitere Informationen, einschließlich Informationen zum Herunterladen des Beispiels, finden Sie unter Samples Gallery: Getting started with Spring sample.
Google Protocol Buffer Serializer	Dieses Beispiel veranschaulicht, wie eine Serialisierungsmethode für eXtreme Scale geschrieben und für die Verwendung mit eXtreme Scale konfiguriert wird. Die mit diesem Beispiel bereitgestellte Serialisierungsmethode verwendet die Google-Protokollpuffer, um Objekte zu beschreiben und zu serialisieren. Weitere Informationen, einschließlich Informationen zum Download des Beispiels, finden Sie auf der Webseite Samples Gallery: Google protocol buffer serializer sample.

Tabelle 14. Verfügbare Beispiele (Forts.)

Beispiel	
Liberty-Profil	Das Beispiel "Airport" wird als Einführung in eine eXtreme-Scale-Installation in der Liberty-Profilumgebung von WebSphere Application Server Version 8.5 bereitgestellt. Liberty ist ein schlanker Anwendungsserver mit geringen JVM-Speicheranforderungen, der in weniger als fünf Sekunden startet. Der Liberty-Profilserver führt einfache Erstellungs-, Lese-, Aktualisierungs- und Löschfunktionen im eXtreme-Scale-Grid in Millisekunden aus. Das Beispiel veranschaulicht, wie große Datenmengen (in diesem Fall Informationen über Tausende von Flughäfen weltweit) über das Liberty-Profil von WebSphere Application Server mit WebSphere eXtreme Scale gespeichert werden können. Weitere Informationen, einschließlich Informationen zum Herunterladen des Beispiels, finden Sie unter Samples Gallery: Liberty profile airport sample.
Multimasterreplikation	Das Einführungsbeispiel in die Multimasterreplikation wird als Schnelleinführung in die Multimasterreplikation bereitgestellt. Weitere Informationen, einschließlich Anweisungen zum Herunterladen des Beispiels, finden Sie auf der Webseite Samples Gallery: Multimaster Replication sample.
OSGi-Framework	Das OSGi-Beispiel wird als Unterstützung bei der Installation und Ausführung eines Datengrids von WebSphere eXtreme Scale im Eclipse-Equinox-OSGi-Framework bereitgestellt. Das Beispiel enthält mehrere Plug-in-Bundles, die die bewährten Verfahren für die Entwicklung dynamischer eXtreme-Scale-Plug-in-Bundles veranschaulichen, sodass die eXtreme-Scale-Server ohne einen kostenaufwendigen Neustart aktualisiert werden können. Weitere Informationen, einschließlich Informationen zum Herunterladen des Beispiels, finden Sie unter Samples Gallery: WebSphere eXtreme Scale OSGi sample.
Abfragen mit der Entity-Manager-API	Dieses Beispiel veranschaulicht, wie Abfragen in einer verteilten partitionierten Map mit der API "EntityManager" verwendet werden. Weitere Informationen, einschließlich Anweisungen zum Herunterladen des Beispiels, finden Sie auf der Webseite Samples Gallery: Running Queries in a partitioned grid using Entity Manager API.

Tabelle 14. Verfügbare Beispiele (Forts.)

Beispiel	
Parallel queries with a ReduceGridAgent implementation	Dieses Beispiel veranschaulicht, wie die Datengrid-API verwendet wird, um eine Abfrage über alle Partitionen im Grid ausgeführt wird. Weitere Informationen, einschließlich Anweisungen zum Herunterladen des Beispiels, finden Sie auf der Webseite Samples Gallery: Running Queries in Parallel using a ReduceGridAgent .
Ressourcenadapter	Stellen Sie eine Verbindung zu einem eXtreme-Scale-Datengrid in Ihrer Anwendung mit dem WebSphere-eXtreme-Scale-Ressourcenadapter über die Java Transaction API (JTA) her. Dieses Beispiel hilft Ihnen bei der Installation des Ressourcenadapters, bei der Konfiguration einer Verbindungsfactory und stellt Codebeispiele für das Herstellen von Verbindungen zu Ihrem Datengrid und zum Abrufen von Objekten aus dem Datengrid bereit. Weitere Informationen, einschließlich Informationen zum Herunterladen des Beispiels, finden Sie unter Samples Gallery: Using the WebSphere eXtreme Scale resource adapter.

Artikel mit Lernprogrammen und Beispielen

Tabelle 15. Verfügbare Artikel nach Feature

Artikel	**Features**
WebSphere eXtreme Scale Tutorial	Partitionierung, Replikation, Shards, Zonen, Programmierungs-APIs, Leistungsoptimierung
Building grid-ready applications	ObjectMap-API, EntityManager-API, Abfragen, Agenten, Java SE und EE, Statistiken, Partitionierung, Verwaltung und Operationen, Eclipse
Skalierbare Datenverarbeitung mit Grids	EntityManager-API, Agenten
Skalierbare, flexible und leistungsfähige Datenbankalternative erstellen	ObjectMap-API, Replikation, Partitionierung, Verwaltung und Operationen, Eclipse
xsadmin für WebSphere eXtreme Scale erweitern	Verwaltung
Redbook: User's Guide	Alle Abschnitte

Kostenlose Testversion

Um sich mit der Verwendung von WebSphere eXtreme Scale vertraut zu machen, laden Sie eine kostenlose Testversion herunter. Sie können innovative Hochleistungsanwendungen entwickeln, indem Sie das Daten-Caching-Konzept mit erweiterten Features erweitern.

Probedownload

Sie können eine kostenlose Testversion von WebSphere eXtreme Scale von der Downloadsite mit der Testversion von eXtreme Scale herunterladen.

Nach dem Download und Entpacken der Testversion von eXtreme Scale navigieren Sie zum Verzeichnis `gettingstarted`, und lesen die Datei `GETTINGSTARTED_README.txt`. Dieses Lernprogramm ist eine Einführung in die Verwendung von eXtreme Scale. Sie erstellen ein Datengrid in mehreren Servern und führen verschiedene einfache Anwendungen zum Speichern und Abrufen von Daten in einem Grid aus. Vor der Implementierung von eXtreme Scale in einer Produktionsumgebung müssen verschiedene Optionen berücksichtigt werden, z. B. die Anzahl der zu verwendenden Server, die Speicherkapazität jedes Servers und die synchrone oder asynchrone Replikation.

Beispieleigenschaftendateien

Servereigenschaftendateien enthalten Einstellungen für die Ausführung der Katalogserver und Container-Server. Sie können eine Servereigenschaftendatei für eine eigenständige Konfiguration oder eine Konfiguration von WebSphere Application Server angeben. Clienteigenschaftendateien enthalten Einstellungen für Ihren Client.

Sie können die folgende Beispieleigenschaftendateien im Verzeichnis *WXS-Installationsstammverzeichnis*`\properties` verwenden, um Ihre Eigenschaftendatei zu erstellen:

- `sampleServer.properties`
- `sampleClient.properties`

Beispiel: Dienstprogramm `xsadmin`

Mit dem Dienstprogramm **`xsadmin`** können Sie Textinformationen zu Ihrer Topologie von WebSphere eXtreme Scale formatieren und anzeigen. Das Beispieldienstprogramm stellt eine Methode für die Syntaxanalyse und die Erkennung aktuelle Implementierungsdaten bereit und kann als Grundlage für das Schreiben angepasster Dienstprogramme verwendet werden.

Vorbereitende Schritte

- Das Dienstprogramm **`xsadmin`** wird als Beispiel bereitgestellt, das veranschaulicht, wie Sie angepasste Dienstprogramme für Ihre Implementierung erstellen können. Das Dienstprogramm **`xscmd`** wird als unterstütztes Dienstprogramm für die Überwachung und Verwaltung Ihrer Umgebung bereitgestellt. Weitere Informationen finden Sie im Abschnitt Verwaltung mit dem Dienstprogramm **`xscmd`** .
- Damit das Dienstprogramm **`xsadmin`** Ergebnisse anzeigt, müssen Sie Ihre Datengridtopologie erstellt haben. Ihre Katalogserver und Container-Server müssen gestartet sein. Weitere Informationen finden Sie unter Eigenständige Server starten und stoppen.
- Vergewissern Sie sich, dass die Umgebungsvariable *JAVA_HOME* so gesetzt ist, dass die mit dem Produkte installierte Laufzeitumgebung verwendet wird. Wenn Sie die Testversion dews Produkts verwenden, müssen Sie die Umgebungsvariable *JAVA_HOME* setzen.

Informationen zu diesem Vorgang

Das Beispieldienstprogramm **xsadmin** verwendet eine Implementierung von Managed Beans (MBeans). Diese Beispielüberwachungsanwendung aktiviert schnell integrierte Überwachungsfunktionen, die Sie mit den Schnittstellen aus dem Paket com.ibm.websphere.objectgrid.management erweitern können. Sie finden Quellcode der Beispielanwendung **xsadmin** in der Datei *WXS-Ausgangsverzeichnis*/samples/xsadmin.jar in einer eigenständigen Installation bzw. in der Datei *WXS-Ausgangsverzeichnis*/xsadmin.jar in einer Installation mit WebSphere Application Server.

Sie können das Beispieldienstprogramm **xsadmin** verwenden, um das aktuelle Layout und den jeweiligen Status des Datengrids, z. B. den Mapinhalt, anzuzeigen. In diesem Beispiel setzt sich das Layout des Datengrids in dieser Aufgabe aus einem einzigen Datengrid (*ObjectGridA*) mit einer einzigen Map (*MapA*) zusammen, die zum MapSet *MapSetA* gehört. Dieses Beispiel veranschaulicht, wie Sie alle aktiven Container in einem Datengrid anzeigen und gefilterte Metriken zur Mapgröße der Map *MapA* ausgeben. Zum Anzeigen aller möglichen Befehlsoptionen führen Sie das Dienstprogramm **xsadmin** ohne Argumente oder mit der Option **-help** aus.

Vorgehensweise

1. Wechseln Sie in das Verzeichnis bin.

   ```
   cd WXS-Ausgangsverzeichnis/bin
   ```

2. Führen Sie das Dienstprogramm **xsadmin** aus.
   - Zum Anzeigen der Onlinehilfe führen Sie den folgenden Befehl aus:

     UNIX

     ```
     xsadmin.sh
     ```

     Windows

     ```
     xsadmin.bat
     ```

     Sie dürfen nur eine einzige der aufgelisteten Optionen übergeben, damit das Dienstprogramm funktioniert. Wenn Sie keine Option **-g** oder **-m** angeben, gibt das Dienstprogramm **xsadmin** Informationen zu jedem Grid in der Topologie aus.
   - Führen Sie den folgenden Befehl aus, um Statistiken für alle Server zu aktivieren:

     UNIX

     ```
     xsadmin.sh -g ObjectGridA -setstatsspec ALL=enabled
     ```

     Windows

     ```
     xsadmin.bat -g ObjectGridA -setstatsspec ALL=enabled
     ```
   - Wenn Sie alle Onlinecontainer für ein Grid anzeigen möchten, führen Sie den folgenden Befehl aus:

     UNIX

     ```
     xsadmin.sh -g ObjectGridA -m MapSetA -containers
     ```

     Windows

     ```
     xsadmin.bat -g ObjectGridA -m MapSetA -containers
     ```

     Alle Containerinformationen werden angezeigt. Es folgt ein Beispiel für die Ausgabe:

```
Connecting to Catalog service at localhost:1099


*** Show all online containers for grid - ObjectGridA & mapset - MapSetA

Host: 192.168.0.186
Container: server1_C-0, Server:server1, Zone:DefaultZone
Partition Shard Type
        0 Primary

Num containers matching = 1
Total known containers = 1
Total known hosts = 1
```

Achtung: Zum Abrufen dieser Informationen, wenn Transport Layer Security/Secure Sockets Layer (TLS/SSL) aktiviert ist, müssen Sie die Katalog- und Container-Server mit definiertem JMX-Service-Port starten. Für die Definition des JMX-Service-Ports können Sie die Option **-JMXServicePort** im Script **startOgServer** verwenden oder die Methode setJMXServicePort in der Schnittstelle ServerProperties aufrufen.

- Führen Sie den folgenden Befehl aus, um die Verbindung zum Katalogservice herzustellen und Informationen zu MapA anzuzeigen:

  UNIX

  ```
  xsadmin.sh -g ObjectGridA -m MapSetA -mapsizes -fm MapA
  ```

  Windows

  ```
  xsadmin.bat -g ObjectGridA -m MapSetA -mapsizes -fm MapA
  ```

  Die Größe der angegebenen Map wird angezeigt. Es folgt ein Beispiel für die Ausgabe:

  ```
  Connecting to Catalog service at localhost:1099

  ****Displaying Results for Grid - ObjectGridA, MapSet - MapSetA*****


  *** Listing Maps for server1 ***
   Map Name Partition Map Size Used Bytes (B) Shard Type
   MapA     0         0        0              Primary
  ```

- Führen Sie den folgenden Befehl aus, um eine Verbindung zum Katalogservice unter Verwendung eines bestimmten JMX-Ports herzustellen und Informationen zu MapA anzuzeigen: UNIX

  ```
  xsadmin.sh -g ObjectGridA -m MapSetA -mapsizes -fm MapA
    -ch CatalogMachine -p 6645
  ```

  Windows

  ```
  xsadmin.bat -g ObjectGridA -m MapSetA -mapsizes -fm MapA
    -ch CatalogMachine -p 6645
  ```

  Das Beispieldienstprogramm **xsadmin** stellt eine Verbindung zum MBean-Server her, der in einem Katalogserver ausgeführt wird. Ein Katalogserver kann als eigenständiger Prozess, als Prozess von WebSphere Application Server oder integriert in einem angepassten Anwendungsprozess ausgeführt werden. Verwenden Sie die Option **-ch**, um den Hostnamen des Katalogservice anzugeben, und die Option **-p**, um den Port des Katalogservice anzugeben.

  Die Größe der angegebenen Map wird angezeigt. Es folgt ein Beispiel für die Ausgabe:

```
Connecting to Catalog service at CatalogMachine:6645

*****Displaying Results for Grid - ObjectGridA, MapSet - MapSetA*****


*** Listing Maps for server1 ***
Map Name: MapA  Partition #: 0  Map Size: 0  Shard Type: Primary
Server Total: 0
```

- Wenn Sie eine Verbindung zu einem Katalogservice in einem Prozess von WebSphere Application Server herzustellen, führen Sie die folgenden Schritte aus:

  Die Option **-dmgr** ist erforderlich, wenn eine Verbindung zu einem Katalogservice in einem Prozess oder Prozesscluster von WebSphere Application Server hergestellt werden soll. Verwenden Sie die Option **-ch**, um den Hostnamen anzugeben, falls dieser nicht `localhost` ist, und die Option **-p**, um den Bootstrap-Port des Katalogservice zu überschreiben, der den mit BOOTSTRAP_ADDRESS angegebenen Prozess verwendet. Die Option **-p** ist nur erforderlich, wenn BOOTSTRAP_ADDRESS nicht auf den Standardwert von 9809 gesetzt ist.

  Anmerkung: Die eigenständige Version von WebSphere eXtreme Scale kann nicht verwendet werden, um eine Verbindung zu einem Katalogservice in einem Prozess von WebSphere Application Server herzustellen. Verwenden Sie das Script **xsadmin** im Verzeichnis *WAS-Stammverzeichnis*/bin, das verfügbar ist, wenn Sie WebSphere eXtreme Scale in WebSphere Application Server oder WebSphere Application Server Network Deployment installieren.

  a. Navigieren Sie zum Verzeichnis "bin" von WebSphere Application Server.
     ```
     cd WAS-Stammverzeichnis/bin
     ```
  b. Starten Sie das Dienstprogramm **xsadmin** mit dem folgenden Befehl:

     UNIX
     ```
     xsadmin.sh -g ObjectGridA -m MapSetA -mapsizes -fm MapA -dmgr
     ```

     Windows
     ```
     xsadmin.bat -g ObjectGridA -m MapSetA -mapsizes -fm MapA -dmgr
     ```

  Die Größe der angegebenen Map wird angezeigt.

  ```
  Connecting to Catalog service at localhost:9809

  ****Displaying Results for Grid - ObjectGridA, MapSet - MapSetA*****


  *** Listing Maps for server1 ***
  Map Name: MapA  Partition #: 0  Map Size: 0  Shard Type: Primary
  Server Total: 0
  ```

- Führen Sie einen der folgenden Befehle aus, um die konfigurierte und die Laufzeitverteilung Ihrer Konfiguration anzuzeigen:

  ```
  xsadmin -placementStatus
  xsadmin -placementStatus -g myOG -m myMapSet
  xsadmin -placementStatus -m myMapSet
  xsadmin -placementStatus -g myOG
  ```

  Sie können den Befehl so ausführen, dass Verteilungsinformationen für die gesamte Konfiguration, für ein einzelnes Datengrid, für ein einzelnes MapSet oder für eine Kombination aus einem Datengrid und einem MapSet angezeigt werden. Es folgt ein Beispiel für die Ausgabe:

  ```
  **********Printing Placement Status for Grid - Grid, MapSet - mapSet**********

  <objectGrid name="Grid" mapSetName="mapSet">
  ```

```
  <configuration>
    <attribute name="placementStrategy" value="FIXED_PARTITIONS"/>
    <attribute name="numInitialContainers" value="3"/>
    <attribute name="minSyncReplicas" value="0"/>
    <attribute name="developmentMode" value="true"/>
  </configuration>
  <runtime>
    <attribute name="numContainers" value="3"/>
    <attribute name="numMachines" value="1"/>
    <attribute name="numOutstandingWorkItems" value="0"/>
  </runtime>
</objectGrid>
```

Konfigurationsprofil für das Dienstprogramm `xsadmin` erstellen

Sie können Parameter, die Sie häufig für das Dienstprogramm **xsadmin** angeben, in einer Eigenschaftendatei speichern. Auf diese Weise werden die Aufrufe des Dienstprogramms **xsadmin** kürzer.

Vorbereitende Schritte

Erstellen Sie eine Basisimplementierung von WebSphere eXtreme Scale, die mindestens einen Katalogserver und mindestens einen Container-Server enthält. Weitere Informationen finden Sie unter Script **startOgServer** (ORB).

Informationen zu diesem Vorgang

Eine Liste der Eigenschaften, die Sie in einem Konfigurationsprofil für das Dienstprogramm **xsadmin** definieren können, finden Sie im Abschnitt „Referenzinformationen zum Dienstprogramm **xsadmin**". Wenn Sie eine Eigenschaftendatei und einen entsprechenden Parameter als Befehlszeilenargument angeben, überschreibt das Befehlszeilenargument den Wert in der Eigenschaftendatei.

Vorgehensweise

1. Erstellen Sie eine Eigenschaftendatei für das Konfigurationsprofil. Diese Eigenschaftendatei sollte alle globalen Eigenschaften enthalten, die Sie in allen Befehlsaufrufen von **xsadmin** verwenden möchten.

   Speichern Sie die Eigenschaftendatei unter einem Namen Ihrer Wahl. Sie können die Datei im folgenden Pfad ablegen: `/opt/ibm/WebSphere/wxs71/ObjectGrid/security/<my.properties>`.

   Ersetzen Sie `<my.properties>` durch den Namen Ihrer Datei. Sie können beispielsweise die folgenden Eigenschaften in Ihrer Datei festlegen:

   - `XSADMIN_TRUST_TYPE=jks`
   - `XSADMIN_TRUST_PATH=/opt/ibm/WebSphere/wxs71/ObjectGrid/bin/security/key.jks`
   - `XSADMIN_USERNAME=ogadmin`

2. Führen Sie das Dienstprogramm "xsadmin" mit der Eigenschaftendatei aus, die Sie erstellt haben. Verwenden Sie den Parameter **-profile**, um die Position Ihrer Eigenschaftendatei anzugeben. Sie können auch den Parameter **-v** verwenden, um eine ausführliche Ausgabe anzuzeigen.

   ```
   ./xsadmin.sh -l -v -password xsadmin -ssl -trustPass ogpass -profile
   /opt/ibm/WebSphere/wxs71/ObjectGrid/security/<my.properties>
   ```

Referenzinformationen zum Dienstprogramm `xsadmin`

Sie können Argumente mit zwei verschiedenen Methoden an das Dienstprogramm **xsadmin** übergeben: mit einem Befehlszeilenargument oder mit einer Eigenschaftendatei.

Argumente für `xsadmin`

Sie können eine Eigenschaftendatei für das Dienstprogramm **xsadmin** mit Version 7.1 Fix 1 oder höher definieren. Durch die Erstellung einer Eigenschaftendatei können Sie sich die Eingabe häufig verwendeter Argumente, wie z. B. des Benutzernamens, sparen. Die Eigenschaften, die Sie einer Eigenschaftendatei hinzufügen können, sind in der folgenden Tabelle aufgelistet. Wenn Sie eine Eigenschaft in einer Eigenschaftendatei und mit dem entsprechenden Befehlszeilenargument angeben, überschreibt der Wert des Befehlszeilenarguments den Wert in der Eigenschaftendatei.

Weitere Informationen zum Definieren einer Eigenschaftendatei für das Dienstprogramm **xsadmin** finden Sie unter „Konfigurationsprofil für das Dienstprogramm **xsadmin** erstellen" auf Seite 309.

Tabelle 16. Argumente für das Dienstprogramm **xsadmin**

Befehlszeilenargument	Entsprechender Eigenschaftsname in der Eigenschaftendatei	Beschreibung und gültige Werte
-bp	Nicht zutreffend	Gibt den Listener-Port an. **Standardwert:** `2809`
-ch	Nicht zutreffend	Gibt den JMX-Hostnamen für den Katalogserver an. **Standardwert:** `localhost`
-clear	Nicht zutreffend	Bereinigt die angegebene Map. **Die folgenden Filter sind zulässig:** `-fm`
-containers	Nicht zutreffend	Zeigt für jedes Datengrid und jedes MapSet eine Liste von Container-Servern an. **Die folgenden Filter sind zulässig:** `-fnp`
-continuous	Nicht zutreffend	Geben Sie dieses Flag an, wenn Sie fortlaufende Ergebnisse zur Mapgröße anzeigen möchten, um das Datengid zu überwachen. Wenn Sie diesen Befehl mit dem Argument **-mapsizes** ausführen, wird die Mapgröße alle 20 Sekunden angezeigt.
-coregroups	Nicht zutreffend	Zeigt alle Stammgruppen für den Katalogserver an. Dieses Argument wird für die erweiterte Diagnose verwendet.
-dismissLink *<Katalogservicedomäne>*	Nicht zutreffend	Entfernt eine Verbindung zwischen zwei Katalogservicedomänen. Geben Sie den Namen der fremden Katalogservicedomäne, zu der Sie zuvor mit dem Argument **-establishLink** eine Verbindung hergestellt haben.
-dmgr	Nicht zutreffend	Gibt an, ob Sie eine Verbindung zu einem in WebSphere Application Server gehosteten Katalogservice herstellen. **Standardwert:** `false`
-empties	Nicht zutreffend	Geben Sie dieses Flag an, wenn Sie leere Container in der Ausgabe anzeigen möchten.
-establishLink *<Name_der_fremden_Domäne> <Host1:Port1,Host2:Port2...>*	Nicht zutreffend	Verbindet die Katalogservicedomäne mit einer fremden Katalogservicedomäne. Verwenden Sie das folgende Format: `-establishLink <Name_der_fremden_Domäne> <Host1:Port1,Host2:Port2...>`. *Name_der_fremden_Domäne* steht für den Namen der fremden Katalogservicedomäne, und *Host1:Port1,Host2:Port2...* steht für eine durch Kommas getrennte Liste mit den Hostnamen der Katalogserver und den ORB-Ports, auf bzw. an denen diese Katalogservicedomäne ausgeführt wird.
-fc	Nicht zutreffend	Filtert nur den angegebenen Container. Wenn Sie Container-Server in einer Umgebung von WebSphere Application Server Network Deployment filtern möchten, verwenden Sie das folgende Format: *<Zellenname>/<Knotenname>/<Servername_Containersuffix>* **Verwenden Sie diesen Filter mit den folgenden Argumenten: `-mapsizes`, `-teardown`,`-revisions`,`-getTraceSpec`,`-setTraceSpec`,`-getStatsSpec`,`-setStatsSpec`**
-fh	Nicht zutreffend	Filtert nur den angegebenen Host. **Verwenden Sie diesen Filter mit den folgenden Argumenten: `-mapsizes`, `-teardown`,`-revisions`,`-getTraceSpec`,`-setTraceSpec`,`-getStatsSpec`,`-setStatsSpec`,`-routetable`**

Tabelle 16. Argumente für das Dienstprogramm **xsadmin** *(Forts.)*

Befehlszeilenargument	Entsprechender Eigenschaftsname in der Eigenschaftendatei	Beschreibung und gültige Werte
-fm	Nicht zutreffend	Filtert nur die angegebene Map. **Verwenden Sie diesen Filter mit den folgenden Argumenten: -clear, -mapsizes**
-fnp	Nicht zutreffend	Filtert Server, die keine primären Shards haben. **Verwenden Sie diesen Filter mit den folgenden Argumenten: -containers**
-fp	Nicht zutreffend	Filtert nur die angegebene Partition. **Verwenden Sie diesen Filter mit den folgenden Argumenten: -mapsizes, -teardown,-revisions,-getTraceSpec,-setTraceSpec,-getStatsSpec,-setStatsSpec,-routetable**
-fs	Nicht zutreffend	Filtert nur den angegebenen Server. Wenn Sie Anwendungsserver in einer Umgebung von WebSphere Application Server Network Deployment filtern möchten, verwenden Sie das folgende Format: *<Zellenname>/<Knotenname>/<Servername>* **Verwenden Sie diesen Filter mit den folgenden Argumenten: -mapsizes, -teardown,-revisions,-getTraceSpec,-setTraceSpec,-getStatsSpec,-setStatsSpec**
-fst	Nicht zutreffend	Filtert nur den angegebene Shard-Typ. Geben Sie P für primäre Shards, A für asynchrone Replikat-Shards und S für synchrone Replikat-Shards an. **Verwenden Sie diesen Filter mit den folgenden Argumenten: -mapsizes, -teardown,-revisions,-getTraceSpec,-setTraceSpec,-getStatsSpec,-setStatsSpec**
-fz	Nicht zutreffend	Filtert nur die angegebene Zone. **Verwenden Sie diesen Filter mit den folgenden Argumenten: -mapsizes, -teardown,-revisions,-getTraceSpec,-setTraceSpec,-getStatsSpec,-setStatsSpec,-routetable**
-force	Nicht zutreffend	Erzwingt die mit dem Befehl angegebene Aktion und inaktiviert damit alle präventiven Aufforderungen. Dieses Argument ist für die Ausführung von Befehlen in Stapeln hilfreich.
-g	Nicht zutreffend	Gibt den ObjectGrid-Namen an.
-getstatsspec	Nicht zutreffend	Zeigt die aktuelle Statistikspezifikation an. Sie können die Statistikspezifikation mit dem Argument **-setstatsspec** festlegen. **Die folgenden Filter sind zulässig:** `-fst -fc -fz -fs -fh -fp`
-getTraceSpec	Nicht zutreffend	Zeigt die aktuelle Tracespezifikation an. Sie können die Tracespezifikation mit dem Argument **-settracespec** festlegen. **Die folgenden Filter sind zulässig:** `-fst -fc -fz -fs -fh -fp`
-h	Nicht zutreffend	Zeigt die Hilfe für das Dienstprogramm **xsadmin** an, die eine Liste von Argumenten enthält.
-hosts	Nicht zutreffend	Zeigt alle Hosts in der Konfiguration an.
-jmxUrl	XSADMIN_JMX_URL	Gibt die Adresse eines JMX-API-Connector-Servers im folgenden Format an: `service:jmx:`*`Protokoll`*`:`*`sap`*. Die Variablendefinitionen für *protocol* und *sap* folgen: *Protokoll* Gibt das Transportprotokoll an, das für die Herstellung der Verbindung zum Connector-Server verwendet werden soll. *sap* Gibt die Adresse an, unter der der Connector-Server zu erreichen ist. Weitere Informationen zum Format des JMX-Service-URL finden Sie unter Class JMXServiceURL (Java 2 Platform SE 5.0).
-l	Nicht zutreffend	Zeigt alle bekannten Datengrids und MapSets an.
-m	Nicht zutreffend	Gibt den Namen des MapSets an.
-mapsizes	Nicht zutreffend	Zeigt die Größe jeder Map im Katalogserver an. Mithilfe dieser Informationen können Sie prüfen, ob die Schlüsselverteilung auf die Shards einheitlich erfolgt ist. **Die folgenden Filter sind zulässig:** `-fm -fst -fc -fz -fs -fh -fp`
-mbeanservers	Nicht zutreffend	Zeigt eine Liste aller MBean-Server-Endpunkte an.
-overridequorum	Nicht zutreffend	Überschreibt die Quorumeinstellung, sodass Container-Server-Ereignisse bei einem Ausfall im Rechenzentrum nicht ignoriert werden.

Tabelle 16. Argumente für das Dienstprogramm **xsadmin** *(Forts.)*

Befehlszeilenargument	Entsprechender Eigenschaftsname in der Eigenschaftendatei	Beschreibung und gültige Werte
-password	XSADMIN_PASSWORD	Gibt das Kennwort für die Anmeldung beim Dienstprogramm **xsadmin** an. Geben Sie das Kennwort nicht in der Eigenschaftendatei an, wenn Sie möchten, dass Ihr Kennwort sicher bleibt.
-p	Nicht zutreffend	Gibt den JMX-Port für den Host des Katalogservers an. **Standardwert:** 1099 oder 9809 für einen Host von WebSphere Application Server, 1099 für eigenständige Konfigurationen.
-placementStatus	Nicht zutreffend	Zeigt die konfigurierte Verteilung und die Laufzeitverteilung Ihrer Konfiguration an. Sie können eine Ausgabe für eine Kombination von Datengrids und MapSets oder für die gesamte Konfiguration erstellen: • Gesamte Konfiguration: `-placementStatus` • Bestimmtes Datengrid: `-placementStatus -g` *`my_grid`* • Bestimmtes MapSet: `-placementStatus -m` *`my_mapset`* • Bestimmtes Datengrid und bestimmtes MapSet: `-placementStatus -g` *`my_grid`* `-m` *`my_mapset`*
-primaries	Nicht zutreffend	Zeigt eine Liste der primären Shards an.
-profile	Nicht zutreffend	Gibt einen vollständig qualifizierten Pfad zur Eigenschaftendatei für das Dienstprogramm **xsadmin** an.
-quorumstatus	Nicht zutreffend	Zeigt den Status des Quorums für den Katalogservice an.
-releaseShard *<Name_des_Container-Servers> <ObjectGrid-Name> <MapSet-Name> <Partitionsname>*	Nicht zutreffend	Wird zusammen mit dem Argument **-reserveShard** verwendet. Das Argument **-releaseShard** muss nach der Reservierung oder Verteilung eines Shards aufgerufen werden. Das Argument **-releaseShard** ruft die Methode ContainerMBean.release() auf.
-reserved	Nicht zutreffend	Wird mit dem Argument **-containers** verwendet, um nur die Shards anzuzeigen, die mit dem Argument **-reserveShard** reserviert wurden.
-reserveShard *<Name_des_Container-Servers> <ObjectGrid-Name> <MapSet-Name> <Partitionsname>*	Nicht zutreffend	Verschiebt ein primäres Shard in den angegebenen Container-Server. Die Methode ContainerMBean.reserve() wird von diesem Argument aufgerufen.
-resumeBalancing *<ObjectGrid-Name> <MapSet-Name>*	Nicht zutreffend	Versucht, Anforderungen gleichmäßig zu verteilen. Ermöglicht weitere Neuverteilungsversuche für das angegebene ObjectGrid und das angegebene MapSet.
-revisions	Nicht zutreffend	Zeigt Revisions-IDs für eine Katalogservicedomäne an, einschließlich Datengrids, Partitionsnummer, Partitionstyp (primär oder Replikat), Katalogservicedomäne, Lebensdauer-ID und Anzahl der Datenüberarbeitungen für jedes Shard. Sie können dieses Argument verwenden, um festzustellen, ob Sie ein asynchrones Replikat oder eine verbundene Domäne haben. Dieses Argument ruft die Methode ObjectGridMBean.getKnownRevisions() auf. **Die folgenden Filter sind zulässig:** `-fst -fc -fz -fs -fh -fp`
-routetable	Nicht zutreffend	Zeigt den aktuellen Status des Datengrids aus der Perspektive des Client-Servers an. Die Routentabelle enthält die Informationen, die ein ObjectGrid-Client-Server verwendet, um mit dem Datengrid zu kommunizieren. Verwenden Sie die Routentabelle als Diagnosehilfe, wenn Sie versuchen, Verbindungsprobleme oder Ausnahmen des Typs `TargetNotAvailable` zu ermitteln. **Erforderliche Argumente:** In einer eigenständigen Umgebung müssen Sie die Parameter **-bp** und **-p** mit diesem Argument angeben, wenn Sie die Standardwerte für den Bootstrap-Listener-Port und den JMX-Port für den Host des Katalogservers nicht verwenden. **Die folgenden Filter sind zulässig:** `-fz -fh -fp`

Tabelle 16. Argumente für das Dienstprogramm **xsadmin** *(Forts.)*

Befehlszeilenargument	Entsprechender Eigenschaftsname in der Eigenschaftendatei	Beschreibung und gültige Werte
-settracespec *<Tracezeichenfolge>*	Nicht zutreffend	Aktiviert die Traceerstellung in Servern zur Laufzeit. Sehen Sie sich das folgende Beispiel an: `-setTraceSpec "ObjectGridReplication=all=enabled"` Weitere Informationen zu den Tracezeichenfolgen, die Sie angeben können, finden Sie unter Trace erfassen und Traceoptionen für Server. **Die folgenden Filter sind zulässig:** `-fst -fc -fz -fs -fh -fp`
-swapShardWithPrimary *<Name_des_Container-Servers> <ObjectGrid-Name> <MapSet-Name> <Partitionsname>*	Nicht zutreffend	Tauscht das angegebene Replikat-Shard aus dem angegebenen Container-Server mit dem primären Shard aus. Mit diesem Befehl können Sie primäre Shards bei Bedarf manuell verteilten.
-setstatsspec *<Statistikspezifikation>*	Nicht zutreffend	Aktiviert die Statistikerfassung. Dieses Argument ruft die Methoden DynamicServerMBean.setStatsSpec und DynamicServerMBean.getStatsSpec auf. Weitere Informationen finden Sie unter Statistiken aktivieren. **Die folgenden Filter sind zulässig:** `-fm -fst -fc -fz -fs -fh -fp`
-suspendBalancing *<ObjectGrid-Name> <MapSet-Name>*	Nicht zutreffend	Verhindert künftige Versuche, das angegebene ObjectGrid und das angegebene MapSet zu verteilen.
-ssl	Nicht zutreffend	Gibt an, dass Secure Sockets Layer (SSL) aktiviert wird.
-teardown	Nicht zutreffend	Stoppt eine Liste oder Gruppe von Katalog- und Container-Servern. **Die folgenden Filter sind zulässig:** `-fst -fc -fz -fs -fh -fp` **Format für die Angabe der Serverliste:** *Servername_1*`,`*Servername_2* `...` **Zum Stoppen aller Server in einer Zone schließen Sie das Argument -fz ein:** `–fz` *<Zonenname>* **Zum Stoppen aller Server auf einem Host schließen Sie das Argument -fh ein:** `–fh` *<Hostname>*
-triggerPlacement	Nicht zutreffend	Erzwingt die Durchführung der Shard-Verteilung, wobei der für **numInitialContainers** konfigurierte Wert in der XML-Implementierungsdatei ignoriert wird. Sie können dieses Argument verwenden, wenn Sie Wartungsarbeiten an den Servern vornehmen, um die Fortsetzung der Shard-Verteilung zuzulassen, selbst wenn der Wert von **numInitialContainers** kleiner ist als der konfigurierte Wert.
-trustPass	XSADMIN_TRUST_PASS	Gibt das Kennwort für den angegebenen Truststore an.
-trustPath	XSADMIN_TRUST_PATH	Gibt einen Pfad zur Truststore-Datei an. Beispiel: `etc/test/security/server.public`
-trustType	XSADMIN_TRUST_TYPE	Gibt den Typ des Truststores an. Die gültigen Werte sind JKS, JCEK, PKCS12 usw.
-unassigned	Nicht zutreffend	Zeigt eine Liste mit Shards an, die nicht im Datengrid verteilt werden können. Shards können nicht verteilt werden, wenn der Verteilungsservice eine Einschränkung aufweist, die die Verteilung verhindert.
-username	XSADMIN_USERNAME	Gibt den Benutzernamen für die Anmeldung beim Dienstprogramm **xsadmin** an.
-v	Nicht zutreffend	Aktiviert die ausführliche Befehlszeilenaktion. Verwenden Sie dieses Flag, wenn Sie Umgebungsvariablen und/oder eine Eigenschaftendatei verwenden, um bestimmte Befehlszeilenargumente anzugeben und deren Werte anzuzeigen. Weitere Informationen finden Sie unter „Option "verbose" des Dienstprogramms **xsadmin**” auf Seite 314.
-xml	Nicht zutreffend	Gibt die ungefilterte Ausgabe der Methode PlacementServiceMBean.listObjectGridPlacement() an. Die anderen **xsadmin**-Argumente filtern die Ausgabe dieser Methode und organisieren die Daten in einem benutzerfreundlicheren Format.

Option "verbose" des Dienstprogramms xsadmin

Sie können die Option "verbose" von **xsadmin** verwenden, um Probleme zu beheben. Führen Sie den Befehl `xsadmin -v` aus, um alle konfigurierten Parameter aufzulisten. Die Option "verbose" zeigt alle Werte in allen Geltungsbereichen an, einschließlich Befehlszeilenargumenten, Argumenten in Eigenschaftendateien und umgebungsdefinierten Argumenten. Der Abschnitt `Effective arguments` enthält die Einstellungen, die in der Umgebung verwendet werden, wenn Sie dieselbe Eigenschaft mit mehreren Geltungsbereichen angegeben haben.

Beispiel für die Verwendung der Option "verbose"

Argument des Befehls xsadmin:

Der folgende Text ist ein Beispielausgabe, die angezeigt wird, wenn Sie die Option "verbose" in der Befehlszeile verwenden, nachdem Sie den folgenden Befehl mit einem angegebenen Eigenschaftswert ausgeführt haben:

```
./xsadmin -l -v -username xsadmin -password xsadmin -ssl -trustPass ogpass
-profile /opt/ibm/WebSphere/wxs71/ObjectGrid/security/my.properties
```

Argumente in der Eigenschaftendatei:

Im Folgenden sehen Sie den Inhalt der Datei `/opt/ibm/WebSphere/wxs71/ObjectGrid/security/my.properties properties`:

```
XSADMIN_TRUST_PASS=ogpass
XSADMIN_TRUST_TYPE=jks
XSADMIN_TRUST_PATH=/opt/ibm/WebSphere/wxs71/ObjectGrid/bin/security/key.jks
XSADMIN_USERNAME=ogadmin
XSADMIN_PASSWORD=ogpass
```

Befehlsergebnisse:

In der folgenden Ausgabe des vorherigen Befehls **xsadmin** steht der Text in ***fetter Kursivschrift*** für Eigenschaften und Werte, die in der Befehlszeile und in der Eigenschaftendatei angegeben werden. Im Abschnitt `Effective command line arguments` können Sie sehen, dass die in der Befehlszeile angegebenen Argumente die Werte in der Eigenschaftendatei überschreiben.

```
Command line specified arguments
*********************************
XSADMIN_USERNAME=xsadmin
XSADMIN_PASSWORD=xsadmin
XSADMIN_TRUST_PATH=<unspecified>
XSADMIN_TRUST_TYPE=<unspecified>
XSADMIN_TRUST_PASS=ogpass
XSADMIN_PROFILE=/opt/ibm/WebSphere/wxs71/ObjectGrid/security/my.properties
XSADMIN_JMX_URL=<unspecified>
*********************************
Properties file specified arguments
*********************************
XSADMIN_USERNAME=ogadmin
XSADMIN_PASSWORD=ogpass
XSADMIN_TRUST_PATH=/opt/ibm/WebSphere/wxs71/ObjectGrid/bin/security/key.jks
XSADMIN_TRUST_TYPE=jks
XSADMIN_TRUST_PASS=ogproppass
XSADMIN_JMX_URL=<unspecified>
*********************************
Environment-specified arguments
*********************************
XSADMIN_USERNAME=<unspecified>
XSADMIN_PASSWORD=<unspecified>
XSADMIN_TRUST_PATH=<unspecified>
```

```
XSADMIN_TRUST_TYPE=<unspecified>
XSADMIN_TRUST_PASS=<unspecified>
XSADMIN_JMX_URL=<unspecified>
*********************************
Effective arguments
*********************************
XSADMIN_USERNAME=xsadmin
XSADMIN_PASSWORD=xsadmin
XSADMIN_TRUST_PATH=/opt/ibm/WebSphere/wxs71/ObjectGrid/bin/security/key.jks
XSADMIN_TRUST_TYPE=jks
XSADMIN_TRUST_PASS=ogpass
XSADMIN_PROFILE=/opt/ibm/WebSphere/wxs71/ObjectGrid/security/my.properties
XSADMIN_JMX_URL=<unspecified>
SSL authentication enabled: true
*********************************
Connecting to Catalog service at localhost:1099
*** Show all 'objectGrid:mapset' names
Grid Name  MapSet Name
accounting defaultMapSet
```

Achtung: Die Eigenschaft XSADMIN_PROFILE ist trotz Anzeige in der ausführlichen Ausgabe kein gültiger Schlüssel, den Sie in einer Eigenschaftendatei angeben können. Der Wert dieser Eigenschaft in der ausführlichen Ausgabe gibt den verwendeten Eigenschaftswert an, der mit dem Befehlszeilenarguent **-profile** angegeben wird.

Ausgabe ohne die Option "verbose"

Im Folgenden sehen Sie ein Beispiel für dieselbe Befehlsausgabe mit aktivierter Option "verbose":

```
./xsadmin -l -username xsadmin -password xsadmin -ssl -trustPass ogpass
-profile /opt/ibm/WebSphere/wxs71/ObjectGrid/security/my.properties
```

```
Connecting to Catalog service at localhost:1099
*** Show all 'objectGrid:mapset' names
Grid Name  MapSet Name
accounting defaultMapSet
```

Bemerkungen

Hinweise auf IBM Produkte, Programme und Services in dieser Veröffentlichung bedeuten nicht, dass IBM diese in allen Ländern, in denen IBM vertreten ist, anbietet. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Services von IBM verwendet werden können. Anstelle der IBM Produkte, Programme oder Services können auch andere, ihnen äquivalente Produkte, Programme oder Services verwendet werden, solange diese keine gewerblichen oder anderen Schutzrechte von IBM verletzen. Die Verantwortung für den Betrieb der Produkte, Programme oder Fremdservices in Verbindung mit Fremdprodukten und Fremdservices liegt beim Kunden, soweit solche Verbindungen nicht ausdrücklich von IBM bestätigt sind.

Für in diesem Handbuch beschriebene Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieses Handbuchs ist keine Lizenzierung dieser Patente verbunden. Lizenzanforderungen sind schriftlich an folgende Adresse zu richten (Anfragen an diese Adresse müssen auf Englisch formuliert werden):

IBM Director of Licensing
IBM Europe, Middle East & Africa
Tour Descartes
2, avenue Gambetta
92066 Paris La Defense
France

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängigen, erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

IBM Corporation
Mail Station P300
522 South Road
Poughkeepsie, NY 12601-5400
USA
Attention: Information Requests

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Marken

IBM, das IBM Logo und ibm.com sind Marken oder eingetragene Marken der IBM Corporation in den USA und/oder anderen Ländern. Weitere Produkt- und Servicenamen können Marken von IBM oder anderen Unternehmen sein. Eine aktuelle Liste der IBM Marken finden Sie auf der Webseite „Copyright and trademark information“ unter ibm.com/legal/copytrade.shtml

Java und alle auf Java basierenden Marken und Logos sind Marken oder eingetragene Marken der Oracle Corporation und/oder ihrer verbundenen Unternehmen.

Linux i st eine Marke von Linus Torvalds in den USA und/oder anderen Ländern.

Microsoft, Windows, Windows NT und das Windows-Logo sind Marken der Microsoft Corporation in den USA und/oder anderen Ländern.

UNIX ist eine eingetragene Marke von The Open Group in den USA und anderen Ländern.

Weitere Unternehmens-, Produkt- oder Servicenamen können Marken anderer Hersteller sein.

Index

N

O

P

Q

R

S

T

U

V

W

X

Z