WebSphere® eXtreme Scale Versione 7.1
Panoramica del prodotto

WebSphere eXtreme Scale Panoramica del prodotto

IBM

Questa edizione si riferisce alla versione 7, release 1, di WebSphere eXtreme Scale e a tutte le release e modifiche successive se non diversamente indicato nelle nuove edizioni.

Indice

Figure v

Tabelle vii

Informazioni sul manuale *Panoramica sul prodotto* ix

Capitolo 1. WebSphere eXtreme Scale Panoramica 1
Funzioni nuove e obsolete in questa release 4
Funzionamento di WebSphere eXtreme Scale . . . 5
Panoramica sulla distribuzione delle applicazioni . . 7
Integrazione con altri prodotti WebSphere Application Server 7
Modifiche al nome del prodotto 8
Prova gratuita 8
Guida alla programmazione e alla gestione 8

Capitolo 2. Panoramica sulla memorizzazione nella cache 9
Architettura di memorizzazione nella cache: mappe, contenitori, client e cataloghi 9
- Mappe 9
- Contenitori, partizioni e frammenti 10
- Client 12
- Servizi catalogo (Server di catalogo) 13

Topologia della memorizzazione nella cache: memorizzazione nella cache distribuita ed in memoria 14
- Cache in memoria locale 15
- Cache in memoria locale replicata tra peer . . . 16
- Cache distribuita 18
- Topologie della replica della griglia multi-master (AP) 22

Integrazione del database: Cache write-behind, in-line e side 31
- Cache limitata e cache completa 32
- Side cache e in-line cache 32
- Cache in linea 34
- Cache write-behind 36
- Programmi di caricamento 40
- Warm-up e precaricamento dei dati 44
- Precaricamento della mappa 45
- Tecniche di sincronizzazione del database . . . 50
- Invalidazione dei dati della cache obsoleti . . . 51
- Indicizzazione 52

Concetto di memorizzazione nella cache di oggetti Java 54
- Programma di caricamento classe e considerazioni sul percorso di classe. 55
- Gestione delle relazioni 55
- Considerazioni sulla chiave della cache 57
- Prestazione della serializzazione 57
- Inserimento di dati per fusi orari diversi . . . 59

Capitolo 3. Panoramica relativa all'integrazione della cache: JPA, sessioni e memorizzazione nella cache dinamica 61
Programmi di caricamento JPA 61
Plug-in della cache JPA 63
Gestione della sessione HTTP 67
Gestore replica di sessione basata su Listener . . . 70
Provider di cache dinamica 72
Pianificazione della capacità e alta disponibilità (cache dinamica). 84

Capitolo 4. Panoramica sui concetti di scalabilità 89
Scalabilità 89
Griglie, partizioni e frammenti 89
Partizionamento 91
Posizionamento e partizioni 93
Transazioni su partizione singola e sulle partizioni della griglia 97
Scalabilità nelle unità o nei pod 103

Capitolo 5. Panoramica sulla disponibilità 107
Alta disponibilità 107
- Replica per disponibilità. 109
- Tipi di rilevamento failover. 115
- Servizio catalogo ad alta disponibilità 118
- Quorum del server di catalogo 120

Repliche e frammenti. 129
- Allocazione dei frammenti (shard): primario e replica. 131
- Lettura dalle repliche. 133
- Bilanciamento del carico tra repliche 134
- Eventi del ciclo di vita, di ripristino e di errore 134

Instradamento a zone preferite 140
Topologie della replica della griglia multi-master (AP) 144
JMS per la distribuzione delle modifiche della transazione 153
Serie di mappe per la replica 154

Capitolo 6. Panoramica sull'elaborazione della transazione . . 157
Sessioni ed elaborazione della transazione. 157
Transazioni 157
Attributo CopyMode 159
Blocco della voce della mappa. 160
Strategie di blocco. 162
JMS per la distribuzione delle modifiche della transazione 166
Transazioni su partizione singola e sulle partizioni della griglia 167

Capitolo 7. Panoramica sulla sicurezza 175

Capitolo 8. Panoramica servizi dati REST 179

Capitolo 9. Panoramica sull'integrazione Spring framework . . 183

Capitolo 10. Supporti didattici, esempi e campioni 185
Supporto didattico gestore entità: panoramica . . 185
Supporto didattico gestore entità: creazione di una classe di entità 186
Supporto didattico gestore entità: creazione relazioni di entità 187
Supporto didattico gestore entità: schema entità Order 189
Supporto didattico gestore entità: aggiornamento elementi 192
Supporto didattico gestore entità: aggiornamento e rimozione elementi con un indice 193
Supporto didattico gestore entità: aggiornamento e rimozione elementi tramite query 193
Supporto didattico ObjectQuery 194
Supporto didattico ObjectQuery - passo 1 . . . 195
Supporto didattico ObjectQuery - passo 2 . . . 196
Supporto didattico ObjectQuery - passo 3 . . . 197
Supporto didattico ObjectQuery - passo 4 . . . 199
Supporto didattico sulla sicurezza Java SE: panoramica 201
Supporto didattico sulla sicurezza - Passo 1 Java SE 202
Java SE supporto didattico sulla sicurezza - Passo 2 205
Java SE Supporto didattico - Passo 3 212
Java SE supporto didattico - Passo 4. 215
Esempio e supporto didattico del servizio dati REST 219
Convenzioni sulle directory 220
Abilitazione del servizio dati REST 221
Configurazione dei server delle applicazioni per il servizio dati REST 230
Utilizzo di un browser con i servizi dati REST 237
Utilizzo di un client Java con i servizi dati REST 239
Visual Studio 2008 WCF client con servizio dati REST 241

Informazioni particolari 245

Marchi 247

Indice analitico. 249

Figure

1. Topologia ad alto livello. 2
2. Mappa 9
3. Serie di mappe 10
4. Contenitore 11
5. Partizione 11
6. Frammento 12
7. ObjectGrid 12
8. Topologie possibili 13
9. Servizio catalogo 13
10. Dominio del servizio catalogo 14
11. Scenario della cache in memoria locale . . . 15
12. Cache replicata tra peer con modifiche che vengono propagate con JMS 16
13. La cache replicata tra peer con modifiche propagate con il gestore HA (High Availability) 17
14. Cache distribuita 19
15. Cache locale 19
16. Cache incorporata 21
17. ObjectGrid come un buffer del database 31
18. ObjectGrid come una side cache 32
19. Side cache 33
20. In-line cache 34
21. Memorizzazione nella cache read-through 35
22. Memorizzazione nella cache write-through 35
23. Memorizzazione nella cache write-behind 36
24. Memorizzazione nella cache write-behind 37
25. Programma di caricamento 41
26. Plug-in Loader 44
27. Programma di caricamento del client 45
28. Aggiornamento periodico 50
29. Architettura del programma di caricamento JPA 62
30. Topologia incorporata JPA 64
31. Topologia incorporata JPA suddivisa in partizioni 65
32. Topologia JPA remota 66
33. Topologia della gestione sessioni HTTP con una configurazione remota del contenitore . . 69
34. Percorso di comunicazione tra frammenti primari e di replica 130
35. Posizionamento di una mapSet ObjectGrid con una politica di distribuzione di 3 partizioni con il valore 1 per minSyncReplicas, il valore 1 per maxSyncReplicas e il valore 1 per maxAsyncReplicas.. 133
36. Esempio di posizionamento di una serie di mappe ObjectGrid per la partizione partition0. La politica di distribuzione ha il valore minSyncReplicas uguale a 1, il valore maxSyncReplicas uguale a 2 ed il valore maxAsyncReplicas uguale a 1. 137
37. Malfunzionamento del contenitore per il frammento primario 137
38. Il frammento di replica sincrona sul contenitore 2 ObjectGrid diventa il frammento primario 138
39. La macchina B contiene il frammento primario. In base al modo in cui è impostata la modalità di correzione automatica ed alla disponibilità dei contenitori, su una macchina potrebbe essere posizionato un nuovo frammento di replica sincrona.. 138
40. Elementi primari e repliche nelle zone 142
41. WCF Data Services di Microsoft 179
42. Servizio dati REST WebSphere eXtreme Scale 180
43. Schema entità Order 189
44. Introduzione alla topologia di esempio 219
45. Diagramma dello schema di esempio di Microsoft SQL Server Northwind 222
46. Diagramma dello schema dell'entità Customer e Order 223
47. Diagramma dello schema dell'entità Category e Product 224
48. Diagramma dello schema entità Customer e Order 225

Tabelle

1. Nuove funzioni in WebSphere eXtreme Scale Versione 7.1 4
2. Funzioni obsolete 5
3. Approcci di arbitraggio 26
4. Valore di stato e risposta 47
5. Sequenza commit sul frammento primario 48
6. Elaborazione del commit sincrono 48
7. Confronto delle funzioni 75
8. Integrazione tecnologia seamless 76
9. Interfacce di programmazione 77
10. Valore di stato e risposta 112
11. Sequenza commit sul frammento primario 113
12. Elaborazione del commit sincrono. 113
13. Rilevamento dell'errore e riepilogo del recupero 118
14. Approcci di arbitraggio 148
15. Articoli disponibili per funzione 185
16. Archivio a repository 234
17. Valori di installazione 235

Informazioni sul manuale *Panoramica sul prodotto*

L'insieme della documentazione WebSphere eXtreme Scale comprende tre volumi che forniscono le informazioni necessarie per utilizzare, programmare e gestire il prodotto WebSphere eXtreme Scale.

Libreria WebSphere eXtreme Scale

La libreria WebSphere eXtreme Scale contiene i seguenti manuali:

- Il manuale *Guida alla gestione* contiene informazioni necessarie per gli amministratori di sistema, che comprendono informazioni su come pianificare le distribuzioni dell'applicazione, pianificare la capacità, installare e configurare il prodotto, avviare ed arrestare i server, monitorare l'ambiente e renderlo sicuro.
- Il manuale *Guida alla programmazione* contiene informazioni per gli sviluppatori di applicazioni su come sviluppare le applicazioni per WebSphere eXtreme Scale utilizzando le informazioni API incluse.
- Il manuale *Panoramica sul prodotto* contiene una vista di livello superiore dei concetti WebSphere eXtreme Scale, che comprendono l'impiego di scenari d'utilizzo e di supporti didattici.

Per scaricare i manuali, andare alla WebSphere eXtreme Scale pagina della libreria.

È possibile inoltre accedere alle stesse informazioni presenti in questa libreria nel WebSphere eXtreme Scale centro informazioni.

A chi è rivolto questo manuale

Questo manuale è rivolto a chiunque sia interessato nell'acquisire informazioni su WebSphere eXtreme Scale.

Struttura di questo manuale

Il manuale contiene informazioni sui seguenti argomenti principali:

- Il **capitolo 1** contiene una panoramica di WebSphere eXtreme Scale
- Il **capitolo 2** contiene informazioni su concetti relativi alla cache nel prodotto.
- Il **capitolo 3** contiene informazioni sull'integrazione della cache.
- Il **capitolo 4** contiene informazioni sulla scalabilità.
- Il **capitolo 5** contiene informazioni sulla disponibilità.
- Il **capitolo 6** contiene informazioni sulla sicurezza.
- Il **capitolo 7** contiene informazioni sull'elaborazione delle transazioni.
- Il **capitolo 8** contiene supporti didattici con concetti base sul prodotto.
- Il **capitolo 9** contiene il glossario del prodotto.

Come ottenere aggiornamenti a questo manuale

È possibile ottenere aggiornamenti a questo manuale scaricando la versione più recente dalla WebSphere eXtreme Scale pagina della libreria.

Come inviare i commenti

Contattare il team della documentazione. Sono state trovate le informazioni richieste? Le informazioni erano accurate e complete? Inviare commenti relativi a questa documentazione via e-mail all'indirizzo wasdoc@us.ibm.com.

Capitolo 1. WebSphere eXtreme Scale Panoramica

WebSphere eXtreme Scale è una griglia di dati in memoria adattabile, scalabile. Essa dinamicamente memorizza in cache, partizioni, repliche e gestisce dati dell'applicazione e logiche applicative tra più server. WebSphere eXtreme Scale esegue volumi considerevoli di elaborazioni della transazione con elevata efficienza e scalabilità lineare. Con WebSphere eXtreme Scale, è possibile anche ottenere qualità del servizio come l'integrità transazionale, elevata disponibilità e tempi di risposta prevedibili.

La scalabilità adattabile è possibile mediante l'utilizzo di memorizzazioni in cache di oggetti distribuiti. Con la scalabilità adattabile, la griglia di dati monitora e gestisce essa stessa. Essa può eseguire un aumento di scala (scale-out) o una riduzione di scala (scale-in) della topologia aggiungendo o rimuovendo server il che incrementa o decrementa la memoria, la velocità di trasmissione della rete e la capacità di elaborazione in base alle necessità. Quando viene inizializzato un processo di scale-out, la capacità viene aggiunta alla griglia di dati mentre è in esecuzione senza richiedere il riavvio. Al contrario, un processo di scale-in rimuove immediatamente la capacità. La griglia di dati è anche capace di risolvere autonomamente, mediante il recupero automatico, gli errori.

WebSphere eXtreme Scalepuò essere utilizzato in diversi modi. Può essere utilizzato come una cache molto potente oppure come un modulo di uno spazio di elaborazione di database in memoria per gestire lo stato dell'applicazione o come una piattaforma per eseguire il build di potenti applicazioni XTP (Extreme Transaction Processing).

È importante notare, tuttavia, che eXtreme Scale non può essere considerato un effettivo database in memoria, molto di ciò è dovuto al fatto che il più avanzato è troppo semplice per gestire situazioni complesse come quelle che può gestire eXtreme Scale. Si dovrebbero avere gli stessi benefici con entrambi gli scenari perché essi sono ciascuno in memoria, ma se un database in memoria ha una macchina che non funziona, non è capace di correggere immediatamente il problema. Un tale evento potrebbe essere particolarmente grave se l'intero ambiente è su quell'unica macchina.

Per affrontare un problema di questo tipo, eXtreme Scale suddivide il data set dato in partizioni che sono equivalenti agli schemi della struttura ad albero vincolati. Gli schemi della strutta ad albero vincolati descrivono la relazione tra le entità. Quando si utilizzano le partizioni, le relazioni dell'entità devono modellare una struttura dati ad albero, in cui la cima della struttura ad albero è l'entità root ed è la sola entità che è partizionata. Tutti gli altri child dell'entità root sono memorizzati nella stessa partizione dell'entità root. Ciascuna partizione esiste come copia dell'elemento primario o frammento. Una partizione contiene anche frammenti di replica per il backup dei dati. Un database in memoria non può fornire questo tipo di funzionalità perché non è strutturato e non è dinamico in questo modo, richiedendo all'utente di eseguire manualmente ciò che eXtreme Scale esegue automaticamente. Inoltre, poiché un database in memoria, è un database può consentire operazioni SQL e funziona con notevole miglioramento in termini di velocità di elaborazione rispetto ai database non in memoria. WebSphere eXtreme Scaledispone di un proprio linguaggio di query rispetto al supporto SQL, ma è significativamente più adattabile, consente il partizionamento dei dati e fornisce un recupero affidabile degli errori.

Con la funzione di cache write-behind, WebSphere eXtreme Scale può essere utilizzato come una cache di front-end per un database. Utilizzando questa cache di front-end, si incrementa il livello della prestazione mentre si riduce il carico del database e il conflitto. WebSphere eXtreme Scale fornisce scalabilità in riduzione (scaling-in) e scalabilità in aumento (scalinng-out) prevedibili al costo di elaborazione prevedibile.

La seguente immagine mostra che in un ambiente di cache ben connesso, distribuito i client eXtreme Scale inviano e ricevono dati dalla griglia di dati che possono essere automaticamente sincronizzati con l'archivio dati di backend. La cache è ben connessa perché tutti i client vedono gli stessi dati nella cache. Ciascuna parte di dati viene memorizzata esattamente su un server scrivibile nella cache, evitando copie onerose di record che potrebbero potenzialmente contenere diverse versioni di dati. Una cache ben connessa conserva tanti più dati quanti più server vengono aggiunti alla griglia dati e scala in modo lineare adattandosi alla crescita della dimensione della griglia di dati. I dati possono anche essere facoltativamente replicati per la tolleranza di errori aggiuntivi.

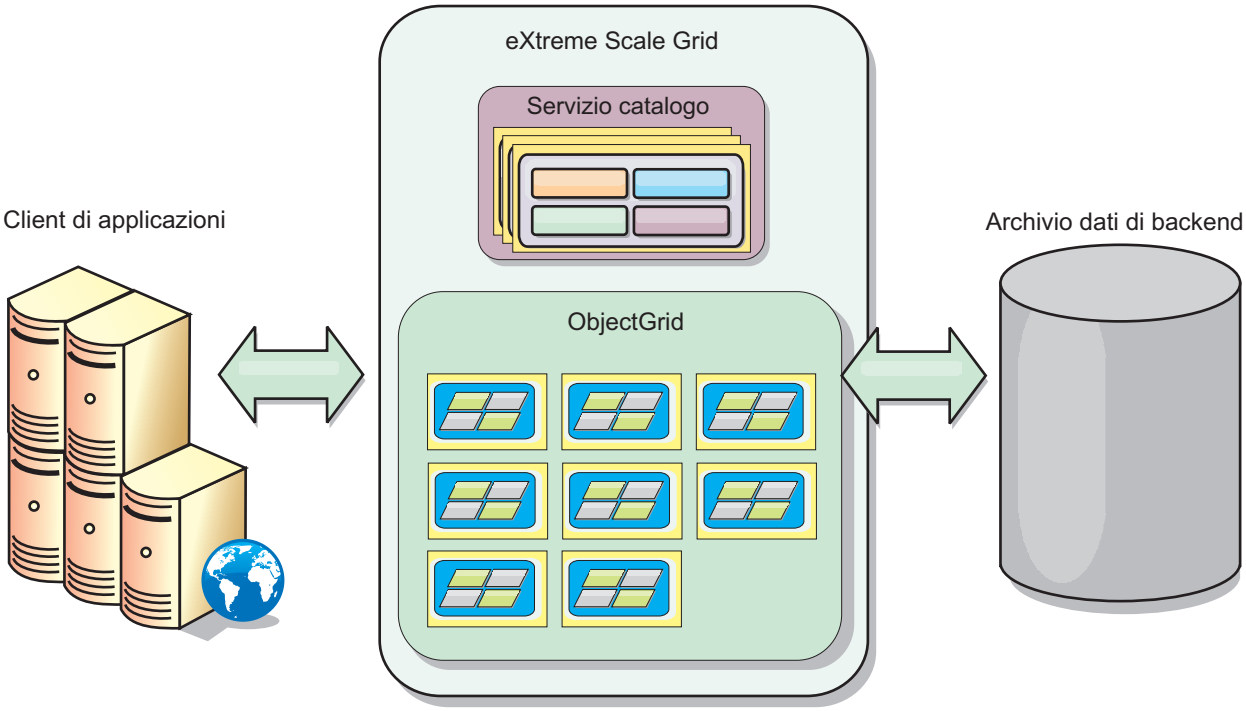


Figura 1. Topologia ad alto livello

WebSphere eXtreme Scaledispone di server che forniscono la propria griglia di dati in memoria. Questi server possono essere in esecuzione all'interno di WebSphere Application Server, o su una semplice Java™ Standard Edition (J2SE) Java virtual machine, consentendo più di una di queste per macchina fisica. Perciò, la griglia di dati in memoria può essere abbastanza larga. La griglia di dati non è limitata dalla memoria e non influisce sulla memoria o lo spazio di indirizzo dell'applicazione o del server delle applicazioni. La memoria può essere la somma della memoria di diverse centinaia o migliaia di Java virtual machine, in esecuzione su differenti macchine.

Come in uno spazio di elaborazione del database in memoria, WebSphere eXtreme Scale può ancora essere eseguito dal disco, dal database o da entrambi.

Mentre eXtreme Scale fornisce diverse API Java, molti utilizzi non richiedono la programmazione utente, ma piuttosto solo la configurazione e la distribuzione della propria infrastruttura WebSphere.

Paradigma di base

Il paradigma fondamentale della griglia di dati è una coppia chiave-valore in cui la griglia di dati memorizza i valori (oggetti Java) utilizzando una chiave associata (un altro oggetto Java). La chiave viene successivamente utilizzata per recuperare il valore. In eXtreme Scale, una mappa è costituita da voci di tali coppie chiave-valore.

WebSphere eXtreme Scaleoffre un numero di configurazioni di griglie di dati da una singola, semplice cache locale ad una larga cache distribuita che utilizza più Java virtual macchine o server.

In aggiunta per memorizzare semplici oggetti Java, è possibile memorizzare gli oggetti con le relazioni. È possibile utilizzare un linguaggio di query che è simile all'SQL con istruzioni `SELECT ... FROM ... WHERE` per recuperare questi oggetti. Ad esempio, un oggetto Order, potrebbe avere un oggetto Customer e più oggetti dell'elemento ad esso associato. WebSphere eXtreme Scale supporta relazioni uno-a-uno, uno-a-molti, molti-a-uno e molti-a-molti.

WebSphere eXtreme Scalesupporta anche un'interfaccia di programmazione EntityManager per memorizzare le entità nella cache. Questa interfaccia di programmazione è molto simile alle entità di Java Enterprise Edition. Le relazioni di entità possono essere automaticamente rilevate da un file XML descrittore dell'entità o da annotazioni nelle classi Java. Perciò, un'entità può essere recuperata dalla cache mediante la chiave primaria che utilizza il metodo find nell'interfaccia EntityManager. Le entità possono essere persistite o rimosse dalla griglia di dati entro i confini della transazione.

WebSphere eXtreme Scalefornisce capability XTP (Extreme Transaction Processing) che garantiscono infrastrutture di applicazioni più intelligenti per supportare la maggior parte delle esigenze delle applicazioni critiche di business. È possibile superare i limiti delle prestazioni IT tradizionali per generare livelli di scalabilità globale, efficienze nel processo e intelligence business necessari per risultati più intelligenti e per un considerevole vantaggio di business competitivo.

Utilizzando il supporto per WebSphereReal Time, l'industria leader nell'offerta Java real-time, WebSphere eXtreme Scale abilita le applicazioni XTP per avere più congruenza e tempi di risposta prevedibili. Consultare le informazioni relative al supporto Real Time Support in *Guida alla gestione*.

Prima della distribuzione di eXtreme Scale in un ambiente di produzione, esistono diverse opzioni da considerare che comprendono il numero di server da utilizzare, la quantità di memoria su ciascun server e la replica sincrona o asincrona.

Considerare un esempio distribuito dove la chiave è un semplice nome alfabetico. La cache potrebbe essere suddivisa in 4 partizioni per chiave: partizione 1 per le chiavi che cominciano con A-E, partizione 2 per le chiavi che cominciano con F-L e così via. Per disponibilità, una partizione ha (è memorizzata in) un frammento dell'elemento primario e un frammento replica. Le modifiche ai dati cache sono apportate al frammento dell'elemento primario e replicate al frammento secondario. Per una cache distribuita (o una griglia di dati o ObjectGrid nel vocabolario eXtreme Scale), è possibile configurare il numero di server di eXtreme

Scale che conterranno la griglia di dati e eXtreme Scale distribuisce i dati tra i frammenti sulle istanze di questi server. Per la disponibilità, i frammenti di replica vengono posizionati in macchine separate dai frammenti degli elementi primari.

WebSphere eXtreme Scaleutilizza un servizio catalogo per individuare il frammento dell'elemento primario per ciascuna chiave. Esso gestisce i frammenti in movimento tra i server eXtreme Scale qualora essi o le macchine fisiche che li contengono, dovessero essere in errore e conseguentemente recuperarli. Ad esempio, se il server contenente un frammento di replica è in errore, eXtreme Scale alloca un nuovo frammento di replica. Se un server contenente un frammento dell'elemento primario è in errore, il frammento di replica è promosso a divenire il frammento dell'elemento primario e come prima viene costruito un nuovo frammento di replica.

La più semplice interfaccia di programmazione eXtreme Scale è ObjectMap, che è una semplice interfaccia di mappa: un metodo map.put(chiave,valore) per inserire un valore nella cache e un metodo map.get(chiave) per recuperare di conseguenza il valore.

Per una discussione relativa alle migliori pratiche che si possono utilizzare quando si progettano le applicazioni WebSphere eXtreme Scale, leggere il seguente articolo su developerWorks: Principi e migliore pratiche per la creazione di applicazioni ad elevate prestazioni ed elevata adattabilità WebSphere eXtreme Scale.

Funzioni nuove e obsolete in questa release

WebSphere eXtreme Scale contiene molte funzioni nuove nella versione 7.1, comprese l'integrazione con la cache dinamica, le mappe array di byte e altro.

Novità in WebSphere eXtreme Scale Versione 7.1

Tabella 1. Nuove funzioni in WebSphere eXtreme Scale Versione 7.1

Funzione	Descrizione
Integrazione di informazioni client DB2	Integrare i plug-in del programma di caricamento JPA eXtreme Scale con DB2, cosicché quando DB2 viene utilizzato come database di backend, le informazioni WebSphere eXtreme Scale (nome utente, nome workstation, nome applicazione e informazioni sull'account) possono essere rese disponibili in DB2 Performance Monitor. Questa funzione consente di abilitare e disabilitare la configurazione di informazioni client per DB2. Per impostazione predefinita questa funzione è disabilitata. Per ulteriori informazioni, vedere “Programmi di caricamento” a pagina 40.
Configurazione del dominio del servizio catalogo	I domini del servizio catalogo possono essere configurati utilizzando la console di gestione WebSphere Application Server o le attività di gestione. I domini del servizio catalogo definiscono un gruppo. Per ulteriori informazioni, consultare le informazioni sulla creazione dei domini del servizio catalogo in *Guida alla gestione*.
Replica multi-master	È possibile collegare in modo asincrono più centri dati, consentendo l'accesso locale del centro dati ai dati, e gestire l'alta disponibilità. Per ulteriori informazioni, consultare “Topologie della replica della griglia multi-master (AP)” a pagina 22.
Programma di eliminazione TTL dell'ultimo tempo di aggiornamento	Il programma di eliminazione TTL è stato aggiornato per tenere traccia del tempo di aggiornamento di una voce, estendendo sul programma di eliminazione TTL (Time-To-Live). Per ulteriori informazioni, consultare le informazioni sul programma di eliminazione TTL (TimeToLive) in *Guida alla programmazione*
Statistiche usedBytes per griglie in memoria	È possibile tenere traccia della quantità di memoria utilizzata dalle voci di cache in una BackingMap utilizzando tutti i provider statistici. Per ulteriori informazioni, consultare le informazioni sul dimensionamento del consumo di memoria cache in *Guida alla gestione*.
Statistiche dinamiche	Le statistiche possono ora essere abilitate e disabilitate su richiesta. Per ulteriori informazioni, consultare le informazioni sul monitoraggio con MBeans in *Guida alla gestione*.
Console di monitoraggio	La console grafica di monitoraggio fornisce viste correnti e cronologiche nelle statistiche del server WebSphere eXtreme Scale. Per ulteriori informazioni, consultare le informazioni sulla console Web in *Guida alla gestione*.
Gestore sessioni HTTP migliorato	La configurazione del gestore sessioni HTTP è stata semplificata. È ora possibile configurare il gestore sessioni HTTP nella console di gestione WebSphere Application Server. Per ulteriori informazioni, consultare le informazioni sulla configurazione del gestore sessioni HTTP in *Guida alla gestione*.
Supporto client a più home	I client possono essere configurati per utilizzare uno specifico adattatore di rete. Per ulteriori informazioni, consultare le informazioni sul file delle proprietà del client in *Guida alla gestione*.

Tabella 1. Nuove funzioni in WebSphere eXtreme Scale Versione 7.1 (Continua)

Funzione	Descrizione
ISA Lite	IBM® Support Assistant Lite per WebSphere eXtreme Scale consente la raccolta di dati automatici e il supporto dell'analisi di sintomi per scenari relativi alla determinazione di problemi. Per ulteriori informazioni, consultare le informazioni relative a IBM Support Assistant per WebSphere eXtreme Scale in *Guida alla gestione*.
REST	Il servizio dati REST fornisce ai client non Java l'accesso ai dati eXtreme Scale, supportando OData (Open Data Protocol) e fornendo piena compatibilità con Microsoft® WCF Data Services. Per ulteriori informazioni, vedere Capitolo 8, "Panoramica servizi dati REST", a pagina 179.
Installazione solo client	I client WebSphere eXtreme Scale possono ora essere installati in modo indipendente, riducendo lo spazio occupato dall'installazione per applicazioni WebSphere eXtreme Scale. Per ulteriori informazioni, consultare le informazioni sull'installazione e sulla distribuzione di WebSphere eXtreme Scale in *Guida alla gestione*.

Funzioni obsolete

Tabella 2. Funzioni obsolete

Obsoleta	Azione di migrazione consigliata
	Creare un dominio del servizio catalogo nella console di gestione WebSphere Application Server, che crea la stessa configurazione di utilizzo della proprietà personalizzata. Per ulteriori informazioni, consultare le informazioni sulla creazione dei domini del servizio catalogo in *Guida alla gestione*.
	Utilizzare invece CatalogServiceManagementMBean.
Funzione di partizione (WPF): la funzione di partizione consiste in una serie di API di programmazione che consentono alle applicazioni Java EE di supportare il clustering assimetrico.	Le funzionalità di WPF possono essere alternativamente realizzate in WebSphere eXtreme Scale.
StreamQuery: una query costante su dati trasmessi memorizzati nelle mappe ObjectGrid.	Nessuna
Configurazione griglia statica: una topologia statica basata su cluster che utilizza il file XML di distribuzione del cluster.	Sostituita con la topologia di distribuzione dinamica migliorata per la gestione di griglie di dati di grandi dimensioni.
Proprietà di sistema obsolete: le proprietà di sistema utilizzate per specificare i file delle proprietà del server e del client sono obsolete.	È ancora possibile utilizzare questi argomenti, ma è necessario modificare le proprietà di sistema con valori nuovi. Per ulteriori informazioni, consultare le informazioni sul file delle proprietà in *Guida alla gestione*.

Funzionamento di WebSphere eXtreme Scale

WebSphere eXtreme Scaleè una griglia di dati in memoria adattabile, scalabile. Essa dinamicamente memorizza in cache, partizioni, repliche e gestisce dati dell'applicazione e logiche di business tra più server.

Poiché non è un database in memoria, è necessario considerare i requisiti di configurazione specifici per eXtreme Scale. La prima operazione per distribuire una griglia di dati eXtreme Scale è avviare un gruppo principale e un servizio catalogo che agirà come coordinatore per tutte le altre Java Virtual machine che partecipano

nella griglia e gestisce le informazioni di configurazione. Le elaborazioni WebSphere eXtreme Scale vengono avviate richiamando un semplice script del comando dalla riga comandi.

L'operazione successiva è avviare i processi del server di WebSphere eXtreme Scale per la griglia da memorizzare e recuperare i dati. Quando i server sono avviati, questi automaticamente registrano essi stessi con il gruppo principale e il servizio catalogo consentendo loro di collaborare per fornire i servizi della griglia. Più server incrementano sia la capacità della griglia che l'affidabilità.

Una griglia locale è una semplice griglia ad istanza singola in cui tutti i dati sono in una sola griglia. Per utilizzare effettivamente eXtreme Scale come uno spazio di elaborazione di database in memoria, è possibile configurare e distribuire una griglia distribuita. I dati nella griglia distribuita vengono suddivisi su vari server eXtreme Scale che li contengono così che ciascun server contiene solo alcuni dati denominata partizione.

Un parametro chiave di configurazione della griglia distribuita è il numero di partizioni nella griglia. I dati della griglia vengono partizionati in questo numero di serie secondarie ciascuna delle quali è denominata partizione. Il servizio catalogo individua la partizione per un determinato dato basato sulla sua chaive. Il numero di partizioni influisce direttamente sulla capacità e la scalabilità della griglia. Un server può contenere una o più partizioni della griglia. Perciò lo spazio di memoria del server limita la dimensione della partizione Al contrario, incrementando il numero di partizioni, si incrementa la capacità della griglia. La capacità massima di una griglia è il numero di partizioni moltiplicato per la dimensione della memoria utilizzabile di un server che può essere una JVM.

i dati della partizione vengono memorizzati in un frammento. Per la disponibilità, una griglia può essere configurata con repliche che possono essere sincrone o asincrone. Le modifiche ai dati della griglia sono apportate al frammento dell'elemento primario e replicate ai frammenti della replica. La memoria complessiva utilizzata/richiesta da una griglia è perciò la dimensione della griglia moltiplicato per (1 (per l'elemento primario) + il numero delle repliche).

WebSphere eXtreme Scale distribuisce i frammenti della griglia sul numero di server che contengono la griglia. Questi server possono essere sulla stessa e/o su macchine fisiche separate. Per la disponibilità, i frammenti di replica vengono posizionati su macchine separate dai frammenti dell'elemento primario.

WebSphere eXtreme Scale monitora lo stato dei suoi server e sposta i frammenti tra questi qualora i frammenti e/o le macchine fisiche che li contengono dovessero essere in errore e conseguentemente li recupera. Ad esempio, se il server che contiene un frammento di replica è in errore, eXtreme Scale allocherà un nuovo frammento di replica e replicherà i dati dall'elemento primario alla nuova replica. Qualora un server contenga un frammento di replica in errore, il frammento di replica è promosso ad essere il frammento dell'elemento primario e, come prima, viene costruito un nuovo frammento di replica. Se si avvia un server aggiuntivo per la griglia, i frammenti saranno distribuiti tra tutti i server in modo che il carico su ciascuno verrà bilanciato come è possibile. Questo viene chiamato scale-out. Analogamente, utilizzando la riduzione di scala (scale-in), è possibile arrestare uno dei server per ridurre le risorse utilizzate da una griglia e i frammenti verranno nuovamente bilanciati tra i server rimanenti solo nel caso di una situazione di errore.

Panoramica sulla distribuzione delle applicazioni

Prima di utilizzare WebSphere eXtreme Scale in un ambiente di produzione considerare i seguenti problemi per ottimizzare la distribuzione.

Pianificazione della distribuzione dell'applicazione

L'elenco che segue comprende gli elementi da considerare:

- numero di sistemi e di processori: quante macchine fisiche e processori sono necessari nell'ambiente?
- Numero di server: quanti server eXtreme Scale per ospitare le mappe eXtreme Scale?
- Numero di partizioni: La quantità di dati memorizzati nelle mappe è un fattore che determina il numero di partizioni necessarie.
- Numero di repliche: quante repliche sono richieste per l'elemento primario nel dominio?
- Replica sincrona o asincrona: I dati sono fondamentali? Perciò è richiesta la replica sincrona? Oppure la prestazione ha un'elevata priorità, rendendo la replica asincrona la scelta corretta?
- Dimensioni dell'Heap: quanti dati verranno memorizzati su ciscun server?

Integrazione con altri prodotti WebSphere Application Server

È possibile integrare WebSphere eXtreme Scale con altri prodotti server come WebSphere Application Server e WebSphere Application Server Community Edition.

Configurazione del gestore di sessione HTTP per lavorare con WebSphere Application Server Community Edition

WebSphere Application Server Community Edition può condividere lo stato della sessione, ma non in modo efficiente e scalabile. WebSphere eXtreme Scalefornisce un'elevata prestazione, un livello di persistenza distribuita che può essere utilizzata per replicare lo stato, ma non prontamente integrato con il server delle applicazioni al di fuori di WebSphere Application Server. È possibile integrare questi due prodotti per fornire una soluzione di gestione della sessione scalabile. Consultare *Guida alla gestione* per ulteriori dettagli.

Configurazione del gestore di sessione WebSphere eXtreme Scale per lavorare con WebSphere Application Server

Il gestore di sessione HTTP prima veniva spedito con WebSphere Extended Deployment DataGrid Versione 6.1.0.0. Le versioni successive alla Versione 6.1.0.5 non hanno modificato i metodi di utilizzo, poiché soddisfano la specifica di J2EE (Java 2 Enterprise Edition) per l'integrazione e il recupero della sessione, ma ci sono stati miglioramenti in QoS e nelle prestazioni in ciascun release. Per garantire l'ottenimento della migliore qualità del servizio, si suggerisce di applicare la Fix pack di WebSphere eXtreme Scale versione 6.1.0.5.

Consultare *Guida alla gestione* per ulteriori dettagli.

Modifiche al nome del prodotto

È necessario sapere che WebSphere eXtreme Scale era precedentemente conosciuto con altri nomi.

Modifiche al nome del prodotto

Quando si fa riferimento ad altra documentazione materiale marketing o presentazioni, tener presente che eXtreme Scale era precedentemente conosciuto con i seguenti nomi.

- ObjectGrid
- WebSphere Extended Deployment Data Grid

Sebbene il prodotto stesso è attualmente conosciuto come WebSphere eXtreme Scale, il termine ObjectGrid appare nella documentazione e altrove perché questo è il nome della risorsa che abilita la tecnologia della griglia.

Prova gratuita

Per iniziare l'utilizzo di WebSphere eXtreme Scale, scaricare una versione di prova gratuita È possibile sviluppare innovative applicazioni ad elevate prestazioni estendendo il concetto relativo di memorizzazione dei dati nella cache utilizzando le funzioni avanzate.

Scaricamento della versione di prova

È possibile scaricare una versione di prova gratuita di eXtreme Scale da Scaricare una versione di prova di eXtreme Scale.

Dopo aver scaricato e decompresso la versione di prova di eXtreme Scale, passare alla directory gettingstarted e leggere il file GETTINGSTARTED_README.txt. Questo supporto didattico guida l'utente utilizzando eXtreme Scale, nella creazione di una griglia su diversi server ed esegue alcune semplici applicazioni per memorizzare e recuperare i dati nella griglia. Prima di distribuire eXtreme Scale in un ambiente di produzione, esistono diverse opzioni da considerare, compreso il numero di server da utilizzare, la quantità di memoria su ciascun server e la replica sincrona o asincrona.

Guida alla programmazione e alla gestione

La *Panoramica sul prodotto* descrive i concetti fondamentali per la comprensione di WebSphere eXtreme Scale. Sono disponibili altre due guide che estendono i concetti descritti in questa guida.

Utilizzare *Guida alla gestione* per le attività generali di configurazione e gestione e *Guida alla programmazione* per le descrizioni delle API Java per accedere e configurare la griglia eXtreme Scale.

Capitolo 2. Panoramica sulla memorizzazione nella cache

WebSphere eXtreme Scale può operare come uno spazio di elaborazione del database in memoria, che può essere utilizzato per fornire la memorizzazione nella in-line cache per un back-end del database oppure per servire come side-cache. La memorizzazione nella in-line cache utilizza eXtreme Scale come mezzo principale per interagire con i dati. Quando eXtreme Scale viene utilizzato come side-cache, il back-end viene utilizzato insieme alla griglia dei dati. Questa sezione descrive i vari concetti e scenari di cache ed esamina le topologie disponibili per la distribuzione di una griglia di dati.

Architettura di memorizzazione nella cache: mappe, contenitori, client e cataloghi

Con WebSphere eXtreme Scale, la propria architettura può utilizzare la memorizzazione dei dati nella cache in memoria locale o la memorizzazione nella cache dei dati client-server.

Per il funzionamento, WebSphere eXtreme Scale richiede una minima infrastruttura aggiuntiva. L'infrastruttura è composta da script per l'installazione, l'avvio e l'arresto di un'applicazione Java Platform, Enterprise Edition su un server. I dati memorizzati nella cache sono memorizzati sul server eXtreme Scale ed i client eseguono la connessione in remoto al server.

Le cache distribuite forniscono migliori prestazioni, disponibilità e scalabilità e possono essere configurate utilizzando le topologie dinamiche, in cui i server vengono automaticamente bilanciati. È inoltre possibile aggiungere ulteriori server senza riavviare i server eXtreme Scale esistenti. È possibile creare distribuzioni semplici o distribuzioni di grandi dimensioni in cui sono necessari migliaia di server.

Mappe

Una mappa è un contenitore di coppie chiave-valore, che consente ad un'applicazione di memorizzare un valore indicizzato mediante una chiave. Le mappe supportano indici che è possibile aggiungere agli attributi dell'indice sulla chiave o sul valore. Tali indici sono automaticamente utilizzati dal runtime della query per determinare il modo più efficace per eseguire una query.

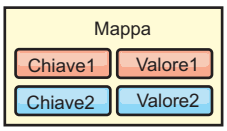


Figura 2. Mappa

Una serie di mappe è una raccolta di mappa con un algoritmo di partizionamento comune. I dati all'interno delle mappe vengono replicati in base alla politica definita sulla serie di mappe. La serie di mappe viene utilizzata solo per le topologie distribuite e non è necessaria per le topologie locali.

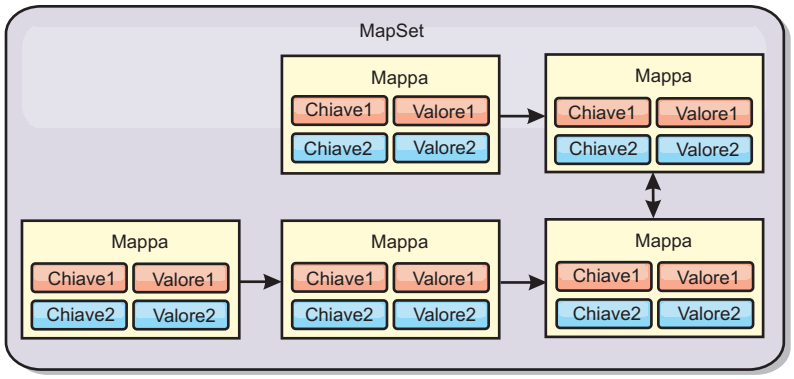


Figura 3. Serie di mappe

Ad una serie di mappe può essere associato uno schema. Uno schema è composto dai metadati che descrivono le relazioni tra ciascuna mappa quando vengono utilizzati entità o tipi di oggetti omogenei.

WebSphere eXtreme Scale può memorizzare oggetti Java serializzabili in ciascuna delle mappe utilizzando l'API ObjectMap. È possibile definire uno schema per le mappe per identificare la relazione tra gli oggetti nelle mappe in cui ciascuna mappa contiene oggetti di un singolo tipo. La definizione di uno schema per le mappe è necessaria per eseguire la query del contenuto degli oggetti della mappa. WebSphere eXtreme Scale può disporre di più schemi di mappa definiti. Per ulteriori dettagli, consultare le informazioni relative all'API ObjectMap in *Guida alla programmazione*.

WebSphere eXtreme Scale può memorizzare anche le entità utilizzando l'API EntityManager. Ciascuna entità è associata ad una mappa. La schema per una serie di mappe dell'entità viene rilevato automaticamente utilizzando un file XML descrittore entità o le classi Java annotate. Ciascuna entità dispone di una serie di attributi chiave e di una serie di attributi non chiave. Inoltre, un'entità può disporre di relazioni con altre entità. WebSphere eXtreme Scale supporta relazioni uno a uno, uno a molti, molti a uno e molti a molti. Ciascuna entità è fisicamente associata ad una singola mappa nella serie di mappe. Le entità consentono alle applicazioni di disporre di grafici oggetto complessi che includono più mappe. Una topologia distribuita può avere più schemi di entità. Per ulteriori dettagli, consultare le informazioni relative all'API EntityManager in *Guida alla programmazione*.

Contenitori, partizioni e frammenti

Il contenitore è un servizio che memorizza i dati dell'applicazione per la griglia. Tali dati sono generalmente suddivisi in parti, chiamate partizioni, e ospitati in più contenitori. Ciascun contenitore, a sua volta, ospita una serie secondaria dei dati completi. Una JVM potrebbe ospitare uno o più contenitori e ciascun contenitore può ospitare più frammenti.

Attenzione: Pianificare la dimensione heap dei contenitori, che ospitano tutti i dati. Configurare di conseguenza le impostazioni heap.

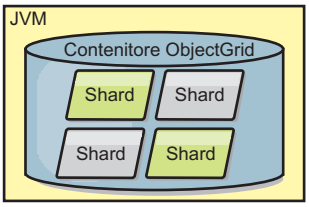


Figura 4. Contenitore

Le partizioni ospitano una serie secondaria dei dati nella griglia. WebSphere eXtreme Scale posiziona automaticamente più partizioni in un singolo contenitore e distribuisce le partizioni man mano che altri contenitori diventano disponibili.

Importante: scegliere attentamente il numero di partizioni prima della distribuzione finale, in quanto tale numero di partizioni non può essere modificato in modo dinamico. Per individuare le partizioni nella rete viene utilizzato un meccanismo hash e eXtreme Scale non è in grado di eseguire nuovamente l'hash dell'intera serie di dati dopo la distribuzione. Come regola generale, è possibile sovrastimare il numero di partizioni

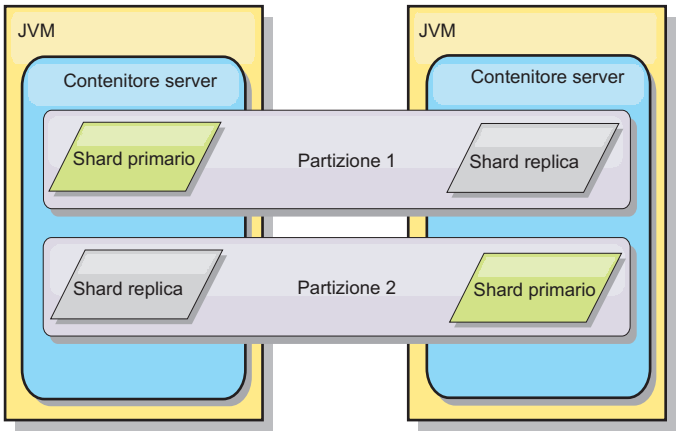


Figura 5. Partizione

I frammenti sono istanze di partizioni e possono avere uno dei due ruoli riportati di seguito: principale o di replica. Il frammento principale e le relative repliche costituiscono la manifestazione fisica della partizione. Ciascuna partizione dispone di diversi frammenti, ciascuno dei quali ospita tutti i dati contenuti in tale partizione. Un frammento è il frammento principale e gli altri sono repliche, ovvero copie ridondanti dei dati nel frammento principale. Il frammento principale è l'unica istanza della partizione che consente alle transazioni di scrivere nella cache. Un frammento di replica è un'istanza "speculare" della partizione. Esso riceve gli aggiornamenti in modo sincrono o asincrono dal frammento principale. Il frammento di replica consente alle transazioni solo operazioni di lettura dalla cache. I frammenti di replica non sono mai ospitati nello stesso contenitore che ospita il frammento principale e, generalmente, non sono ospitati sulla stessa macchina su cui è presente il frammento principale.

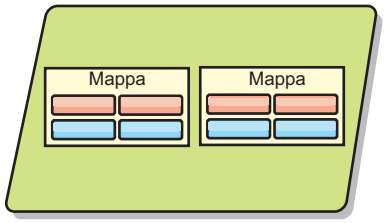


Figura 6. Frammento

Per incrementare la disponibilità dei dati o le garanzie di persistenza dei dati, eseguire la replica dei dati. Tuttavia, la replica rappresenta un costo per la transazione e comporta un peggioramento delle prestazioni in cambio della disponibilità. Con eXtreme Scale, è possibile controllare i costi, in quanto sono supportate le repliche sincrone ed asincrone e modelli di replica ibridi che utilizzano modalità di replica sincrona ed asincrona. Un frammento di replica sincrona riceve gli aggiornamenti come parte della transazione del frammento principale per garantire la congruenza dei dati. La replica sincrona può raddoppiare i tempi di risposta, perché la transazione deve eseguire il commit sul frammento principale e sul frammento di replica sincrona prima di essere completata. Il frammento di replica asincrona riceve gli aggiornamenti dopo il commit della transazione per limitare l'impatto sulle prestazioni, ma introduce la possibilità di perdita dei dati, perché la replica asincrona può essere eseguita dopo diverse transazioni.

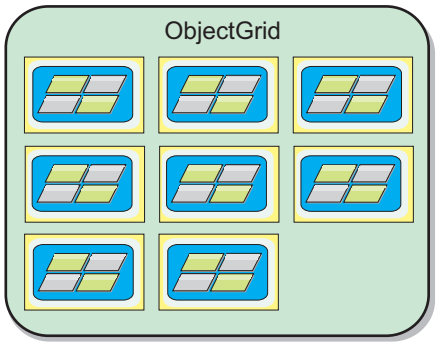


Figura 7. ObjectGrid

Client

I client eseguono la connessione ad un servizio catalogo, richiamano una descrizione della topologia del server e comunicano direttamente con ciascun server, in base alle necessità. Quando la topologia del server viene modificata perché vengono aggiunti nuovi server oppure in caso di malfunzionamento dei server esistenti, il servizio catalogo dinamico indirizza il client verso il server appropriato che ospita i dati. I client devono esaminare le chiavi dei dati dell'applicazione per determinare la partizione a cui indirizzare la richiesta. I client possono leggere i dati da più partizioni in una singola transazione. Tuttavia, i client possono aggiornare solo una singola partizione in una transazione. Dopo che il client ha aggiornato alcune voci, la transazione client devo utilizzare tale partizione per gli aggiornamenti.

Nell'elenco seguente sono riportate le combinazioni di distribuzione possibili:

- Il servizio catalogo esiste nella propria griglia di JVM (Java Virtual Machines). È possibile utilizzare un servizio catalogo singolo per gestire più server o client eXtreme Scale.
- Il contenitore può essere avviato in una JVM oppure caricato in una JVM arbitraria con altri contenitori per istanze ObjectGrid differenti.
- Il client può esistere in qualsiasi JVM e comunicare con una o più istanze ObjectGrid. Il client può esistere anche nella stessa JVM del contenitore.

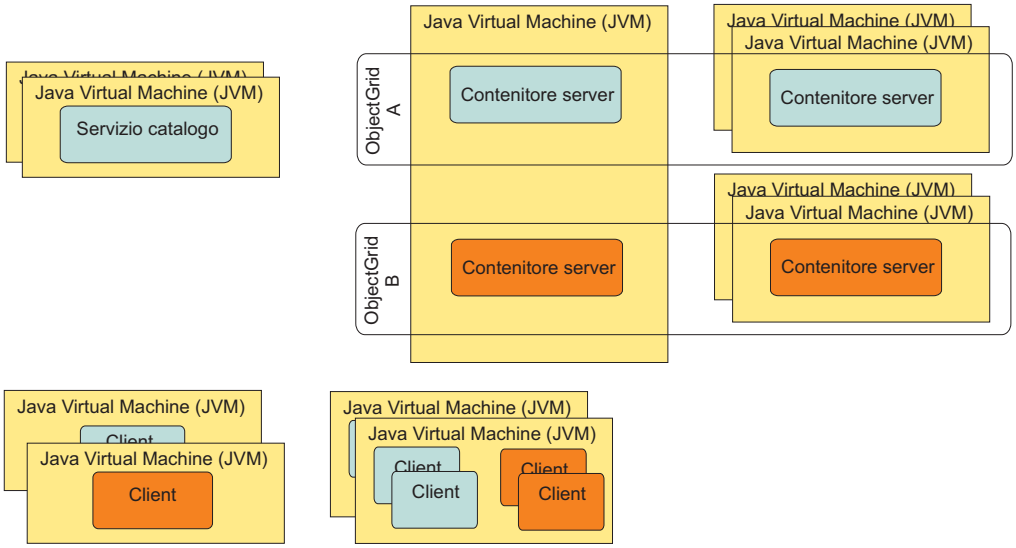


Figura 8. Topologie possibili

Servizi catalogo (Server di catalogo)

Il servizio catalogo contiene la logica che dovrebbe essere inattiva durante uno stato stabile ed influisce in modo non significativo sulla scalabilità. Il servizio catalogo è creato per servire centinaia di contenitori che diventano disponibili contemporaneamente ed esegue dei servizi per gestire i contenitori.

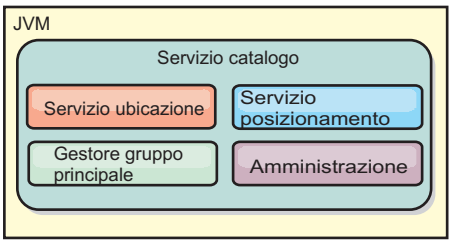


Figura 9. Servizio catalogo

Le responsabilità del catalogo sono rappresentate dai seguenti servizi:

Servizio di ubicazione

Il servizio di ubicazione fornisce l'ubicazione per i client che ricercano i contenitori che ospitano le applicazioni e per i contenitori che cercano di registrare le applicazioni ospitate con il servizio di posizionamento. Il servizio di ubicazione viene eseguito in tutti i membri della griglia per eseguire lo scale out di questa funzione.

Servizio di posizionamento
Il servizio di posizionamento rappresenta il sistema nervoso centrale della griglia ed è responsabile dell'assegnazione dei singoli frammenti ai relativi contenitori host. Il servizio di posizionamento viene eseguito come uno degli N servizi eletti nel cluster. Poiché si utilizza la politica Uno di N, in ogni momento vi sarà esattamente un'istanza del servizio di posizionamento in esecuzione. In caso di arresto di tale istanza, viene eseguito un altro processo. Tutti gli stati del servizio catalogo vengono replicati su tutti i server che ospitano il servizio catalogo per ridondanza.

Gestore del gruppo principale
Il gestore del gruppo principale gestisce il raggruppamento peer per il monitoraggio dello stato, raggruppa i contenitori in piccoli gruppi di server e rende automaticamente federati i gruppi di server. Quando contatta per la prima volta il servizio catalogo, il contenitore attende di essere assegnato ad un nuovo gruppo o ad un gruppo esistente di diverse JVM (Java virtual machine). Ciascun gruppo di JVM (Java virtual machine) monitora la disponibilità di ciascuno dei propri membri mediante la funzione di heartbeat. Uno di tali membri del gruppo trasmette le informazioni relative alla disponibilità al servizio catalogo per consentire la risposta agli errori mediante la riallocazione e l'instradamento.

Amministrazione
Le quattro fasi che costituiscono la gestione dell'ambiente WebSphere eXtreme Scale sono pianificazione, distribuzione, gestione e monitoraggio. Consultare *Guida alla gestione* per ulteriori informazioni relative a ciascuna fase.

Per la disponibilità, configurare un dominio del servizio catalogo. Un dominio del servizio catalogo è formato da più JVM (Java virtual machine), tra cui una JVM principale e diverse JVM (Java virtual machine) di backup.

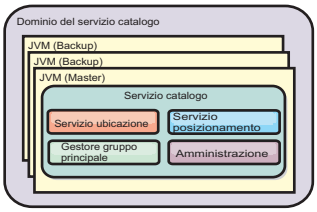


Figura 10. Dominio del servizio catalogo

Topologia della memorizzazione nella cache: memorizzazione nella cache distribuita ed in memoria

Con WebSphere eXtreme Scale, la propria architettura può utilizzare la memorizzazione dei dati nella cache in memoria locale o la memorizzazione nella cache dei dati client-server.

Per il funzionamento, WebSphere eXtreme Scale richiede una minima infrastruttura aggiuntiva. L'infrastruttura è composta da script per l'installazione, l'avvio e l'arresto di un'applicazione Java Platform, Enterprise Edition su un server. I dati memorizzati nella cache sono memorizzati sul server eXtreme Scale ed i client eseguono la connessione in remoto al server.

Le cache distribuite forniscono migliori prestazioni, disponibilità e scalabilità e possono essere configurate utilizzando le topologie dinamiche, in cui i server

vengono automaticamente bilanciati. È inoltre possibile aggiungere ulteriori server senza riavviare i server eXtreme Scale esistenti. È possibile creare distribuzioni semplici o distribuzioni di grandi dimensioni in cui sono necessari migliaia di server.

Cache in memoria locale

Nel caso più semplice, eXtreme Scale può essere utilizzato come una cache di griglia di dati in memoria locale (non distribuita). La cacche in locale può avvantaggiare soprattutto le applicazioni ad alta concorrenza in cui più thread necessitano di accedere e modificare dati transitori. I dati conservati in una griglia locale di eXtreme Scale possono essere indicizzati e recuperati utilizzando WebSphere eXtreme Scale il supporto della query. La possibilità di eseguire query sui dati può essere di grande aiuto per gli sviluppatori quando lavorano con una quantità considerevole di dati in memoria rispetto al supporto della struttura dati limitata fornita con il JVM (Java Virtual Machine), che è pronto per l'uso.

La topologia della cache in memoria locale per eXtreme Scale viene utilizzata per fornire accesso transazionale congruente ai dati temporanei all'interno di una singola Java virtual machine.

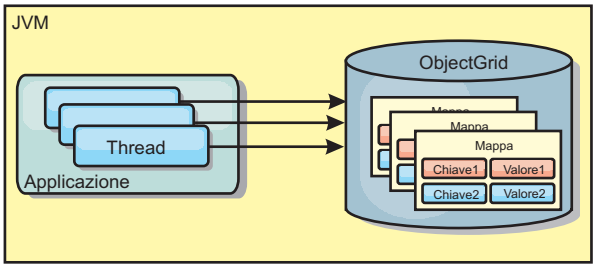


Figura 11. Scenario della cache in memoria locale

Vantaggi

- Impostazione semplice: un ObjectGrid può essere creato in modo programmatico o in modo dichiarato con il file XML descrittore di distribuzione oppure con altri framework come Spring.
- Veloce: ciascuna BackingMap può essere indipendentemente adattata per l'utilizzo ottimale della memoria e la concorrenza.
- Ideale per topologie per singole JVM (Java virtual machine) con piccoli dataset o per dati acceduti frequentemente per la memorizzazione in cache.
- Transazionale. Gli aggiornamenti di BackingMap possono essere raggruppati in una singola unità di lavoro e possono essere incorporati come ultimo partecipante nelle transazioni a 2 fasi come le transazioni JTA (Java Transaction Architecture).

Svantaggi

- Non è tollerato alcun errore.
- I dati non sono replicati. Le cache in memoria sono le migliori per dati di riferimento in sola lettura.
- Non scalabile. La quantità di memoria richiesta dal database potrebbe eccedere laJava virtual machine.
- Si verificano problemi quando si aggiungono Java virtual machine:
  - I dati non possono essere partizionati facilmente

– È necessario replicare manualmente la condizione tra le Java virtual machine o ciascuna istanza di cache potrebbe avere diverse versioni dello stesso dato.
– L'invalidazione è onerosa.
– Per ciascuna cache deve essere impostato il warm-up indipendentemente. Il warm-up è il tempo di caricamento di una serie di dati in modo che la cache possa popolarsi con i dati validi.

Quando utilizzarla

La topologia di distribuzione della cache in memoria locale dovrebbe essere utilizzata solo quando il quantitativo di dati da memorizzare in cache è piccolo (può rientrare in una singola Java virtual machine) ed è relativamente stabile. I dati non aggiornati devono essere tollerati con questo approccio. L'utilizzo di programmi di eliminazione per conservare i dati più frequentemente o recentemente utilizzati nella cache possono guidare a ridurre la dimensione della cache e ad incrementare la validità dei dati.

Cache in memoria locale replicata tra peer

Per una cache locale WebSphere eXtreme Scale, è necessario garantire che la cache sia sincronizzata se esistono più processi con istanze di cache indipendenti. Per fare ciò, abilitare una cache replicata tra peer con JMS.

WebSphere eXtreme Scale include due plug-in che automaticamente propagano le modifiche alla transazione tra le istanze ObjectGrid del peer. Il plug-in JMSObjectGridEventListener propaga automaticamente le modifiche eXtreme Scale utilizzando JMS (Java Messaging Service).

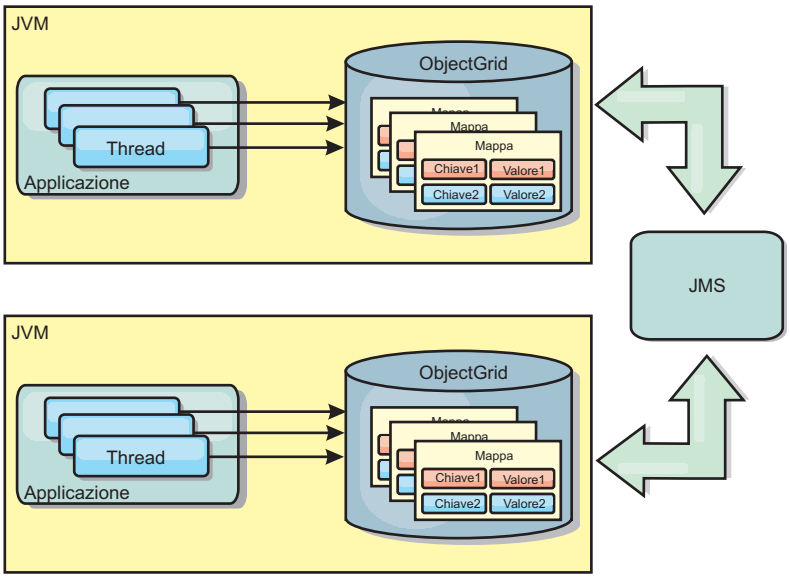


Figura 12. Cache replicata tra peer con modifiche che vengono propagate con JMS

Se si è in esecuzione in un ambiente WebSphere Application Server è disponibile anche il plug-in TranPropListener. Il plug-in TranPropListener utilizza il gestore HA (High Availability) per propagare le modifiche a ciascuna istanza della cache tra peer eXtreme Scale.

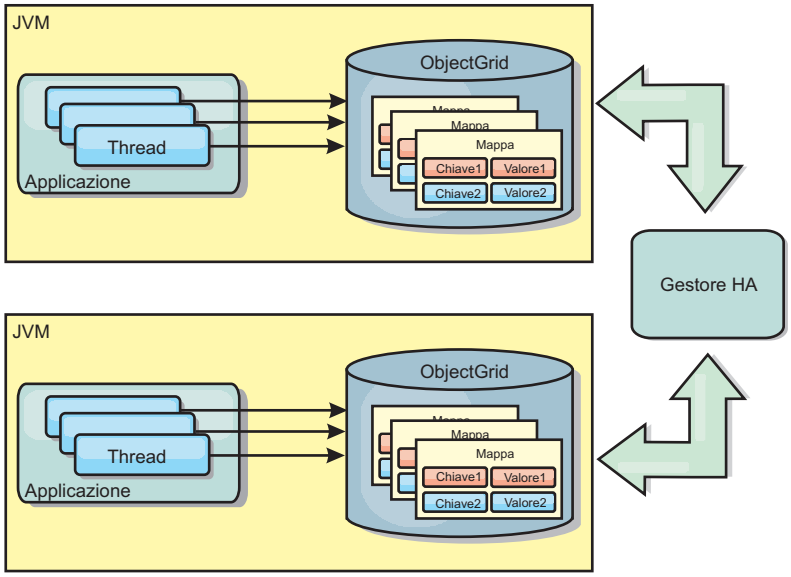


Figura 13. La cache replicata tra peer con modifiche propagate con il gestore HA (High Availability)

Vantaggi

- I dati sono più validi perchè vengono aggiornati più spesso.
- Con il plug-in TranPropListener come in ambiente locale, eXtreme Scale può essere creato in modo programmatico o in modo dichiarato con il file XML del descrittore di distribuzione eXtreme Scale o con gli altri framework come Spring. L'integrazione con il gestore High Availability viene eseguita automaticamente.
- Ciascuna BackingMap può essere indipendentemente sintonizzata per l'ottimale utilizzo della memoria e della concorrenza .
- Gli aggiornamenti di BackingMap possono essere raggruppati in una singola unità di lavoro e possono essere incorporati come l'ultimo partecipante nelle transazioni a due fasi come le transazioni JTA (Java Transaction Architecture).
- Ideale per le topologie few-JVM con un dataset ragionevolmente piccolo o per dati acceduti frequentemente per la memorizzazione in cache.
- Le modifiche a eXtreme Scale sono replicate su tutte le istanze dei eXtreme Scale. Le modifiche sono coerenti finché viene utilizzata una sottoscrizione durevole.

Svantaggi

- La configurazione e la manutenzione per il JMSObjectGridEventListener possono essere complesse. eXtreme Scalepuò essere creato in modo programmatico o in modo dichiarato con il file XML del descrittore di distribuzione eXtreme Scale o con altri framework come Spring.
- Non scalabile: la quantità di memoria richiesta dal database può sopraffare la JVM
- Funziona impropriamente quando si aggiungono Java virtual machine:
  - i dati non possono essere partizionati facilmente
  - l'invalidazione è onerosa.
  - Ciascuna cache deve essere caricata indipendentemente

Quando utilizzare

Questa topologia dovrebbe essere utilizzata solo quando la quantità di dati da memorizzare in cache è piccola (può rientrare in una sola JVM) ed è relativamente stabile.

Cache distribuita

WebSphere eXtreme Scale è più spesso utilizzata come una cache condivisa, per fornire accesso transazionale ai dati a più componenti laddove verrebbe invece utilizzato un database tradizionale. La cache condivisa elimina la necessità di configurare un database.

La cache è coerente in quanto tutti i client vedono gli stessi dati nella cache. Ogni porzione di dati viene memorizzata esattamente su un server nella cache, prevenendo uno dispendio di copie di record che potrebbero potenzialmente contenere versioni differenti di dati. Una cache coerente può anche contenere più dati quanti più server vengono aggiunti alla griglia, e scalare in modo lineare quando la dimensione della griglia cresce. Poiché i client accedono ai dati da questa griglia con chiamate procedurali remote, può anche essere conosciuta come cache remota (o cache lontana). Mediante il partizionamento dei dati, ogni processo comprende una serie secondaria univoca della serie totale dei dati. Le griglie più larghe possono contenere più dati e servizi e più richieste per quei dati. La coerenza elimina anche la necessità di spingere i dati di invalidazione intorno alla griglia poiché non ci sono dati obsoleti. La cache coerente contiene solo l'ultima copia di ogni porzione di dati.

Se si sta eseguendo un ambiente WebSphere Application Server, è disponibile anche il plug-in TranPropListener. Il plug-in TranPropListener utilizza il componente HA Manager (high availability) di WebSphere Application Server per propagare le modifiche ad ogni istanza della cache ObjectGrid peer.

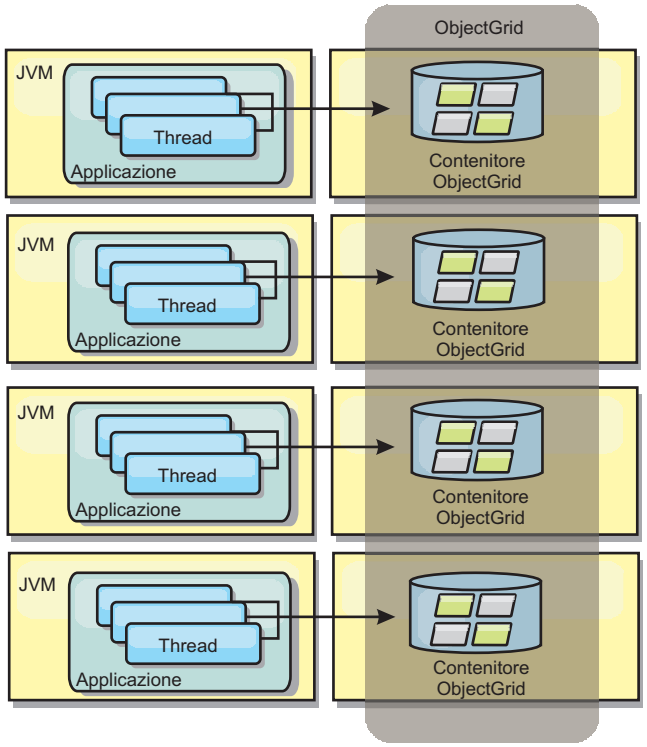


Figura 14. Cache distribuita

Cache locale

I client possono facoltativamente avere una cache locale, in linea, quando eXtreme Scale viene utilizzato in una topologia distribuita. Questa cache facoltativa viene denominata cache locale, un ObjectGrid indipendente su ciascun client, servendo come una cache per la cache in remoto lato server. Come impostazione predefinita la cache locale è abilitata quando il blocco è configurato come Ottimistico o Nessuno e non può essere utilizzata quando è configurato come pessimistico.

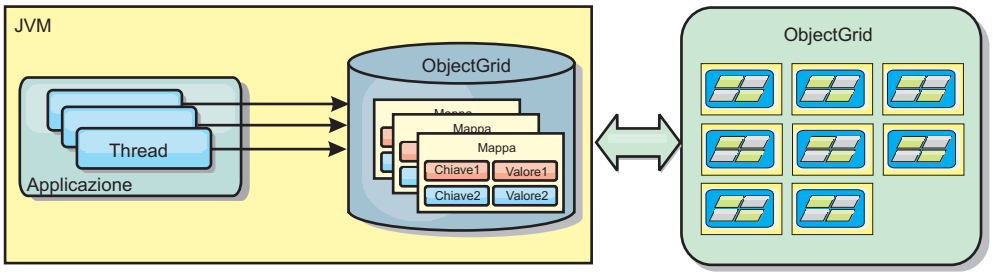


Figura 15. Cache locale

Una cache locale è molto veloce in quanto fornisce accesso in memoria ad una serie secondaria dell'intera serie di dati memorizzati nella cache in remoto sui server eXtreme Scale. La cache locale non è divisa in partizioni e contiene dati provenienti da qualsiasi partizione di eXtreme Scale in remoto. WebSphere eXtreme Scale può avere fino a tre livelli di cache come di seguito illustrato.

1. La cache del livello di transazione contiene tutte le modifiche di una singola transazione. La cache della transazione contiene una copia di lavoro dei dati fino al commit della transazione. Quando una transazione client richiede dati da un ObjectMap, viene prima controllata la transazione.

2. La cache locale nel livello client contiene una serie secondaria di dati provenienti dal livello server. Quando il livello di transazione non ha i dati, viene eseguito il fetch dei dati dalla cache locale se disponibili e vengono inseriti nella cache della transazione.
3. La griglia nel livello server contiene la maggioranza dei dati e viene condivisa tra tutti i client. Il livello del server può essere partizionato, il che consente la memorizzazione nella cache di un gran quantitativo di dati. Quando la cache locale del client non ha i dati, viene eseguito il fetch dal livello server e i dati vengono inseriti nella cache del client. Il livello server può anche avere un plug-in Loader. Quando la griglia non dispone dei dati richiesti, viene richiamato il programma di caricamento (Loader) e i dati che ne risultano vengono inseriti dall'archivio dati di backend nella griglia.

Per disabilitare la cache locale, impostare l'attributo numberOfBuckets su 0 nella configurazione del descrittore di eXtreme Scale per la sostituzione del client. Per i dettagli sulle strategie di blocco di eXtreme Scale, consultare l'argomento sul blocco delle voci della mappa. La cache locale può essere anche configurata in modo che abbia una politica di eliminazione separata e plug-in differenti che utilizzano una configurazione del descrittore eXtreme Scale per la sostituzione del client.

Vantaggi

- Tempo rapido di risposta, poiché tutti gli accessi ai dati sono in locale.

Svantaggi

- Aumenta la durata dei dati obsoleti.
- Bisogna utilizzare un programma di eliminazione per invalidare i dati così da evitare di rimanere senza memoria.

Quando utilizzarla

Usarla quando il tempo di risposta è importante e i dati obsoleti possono essere tollerati.

Cache incorporata

Le griglie eXtreme Scale possono funzionare all'interno di processi esistenti come server eXtreme Scale incorporati oppure possono essere gestiti come processi esterni. Le griglie incorporate sono utili quando si lavora in un server delle applicazioni come WebSphere Application Server. È possibile avviare i server eXtreme Scale non incorporati utilizzando gli script di riga comandi in un processo Java.

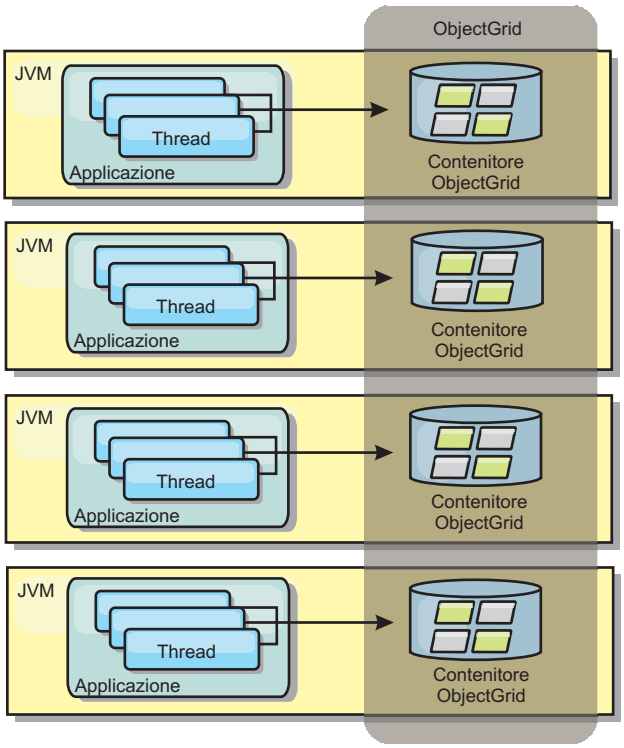


Figura 16. Cache incorporata

Vantaggi

- Amministrazione semplificata poiché ci sono meno processi da gestire.
- Sviluppo delle applicazioni semplificato poiché la griglia utilizza il classloader dell'applicazione client.
- Supporto al partizionamento e alta disponibilità.

Svantaggi

- Aumento dello spazio occupato in memoria nel processo client poiché tutti i dati sono collocati nel processo.
- Aumento dell'utilizzo della CPU per servire le richieste del client.
- Maggiore difficoltà per gestire gli aggiornamenti dell'applicazione poiché i client stanno utilizzando gli stessi file dell'applicazione dell'archivio Java che stanno utilizzando i server.
- Minore flessibilità. La scalabilità di client e di server di griglia non può aumentare alla stessa velocità. Quando i server vengono definiti esternamente, è possibile avere maggiore flessibilità nella gestione del numero dei processi.

Quando utilizzarla

Utilizzare le griglie incorporate quando c'è ampia disponibilità di memoria libera nel processo client per i dati della griglia e potenziali errori di dati.

Per ulteriori informazioni, consultare l'argomento relativo all'abilitazione del meccanismo di invalidazione del client contenuto in *Guida alla gestione*.

Topologie della replica della griglia multi-master (AP)

Utilizzando la funzione di replica asincrona multi-master, due o più griglie possono diventare una copia speculare dell'altra. Questo mirroring viene effettuato utilizzando repliche asincrone tra link che connettono insieme le griglie. Ciascuna griglia è ospitata all'interno di un "dominio" completamente indipendente che possiede un proprio servizio catalogo, server contenitori e un nome dominio univoco. Utilizzando la funzione di replica asincrona, è possibile utilizzare dei link per interconnettere una collezione di questi domini e successivamente sincronizzare i domini utilizzando la replica su questi link. eXtreme Scale abilita a costruire quasi qualsiasi topologia perché la definizione dei link tra i domini è lasciata all'utente.

Domini: Griglie con specifiche caratteristiche

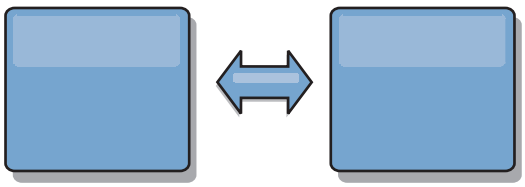
Le griglie utilizzate nelle topologie di replica multi-master sono conosciute come *domini*. Ciascun dominio deve avere le seguenti caratteristiche:

- deve avere un servizio catalogo privato con un nome dominio univoco
- deve avere lo stesso nome griglia delle altre griglie nel dominio
- deve avere lo stesso numero di partizioni delle altre griglie nel dominio
- Deve essere una griglia a FIXED_PARTITION (le griglie PER_CONTAINER non possono essere replicate)
- deve avere lo stesso numero di partizioni (potrebbe avere o meno lo stesso numero e gli stessi tipi di repliche)
- deve avere gli stessi tipi di dati che sono stati replicati dalle altre griglie nel dominio
- deve avere lo stesso nome mapset, i nomi mappe e i template della mappa dinamica delle altre griglie nel dominio.

Dopo aver avviato i domini nella topologia, verrà replicata ogni griglia avente le caratteristiche precedentemente indicate. Considerare che la relativa politica di replica sarà ignorata.

Link di connessione ai domini

Un'infrastruttura di griglia di replica è un grafico connesso di domini con link bidirezionali tra di essi. Un link consente due domini per scambiarsi le modifiche dei dati. Ad esempio, la topologia più semplice è una coppia di domini con un singolo link tra di essi. I domini sono denominati partendo dalla lettera “A” e poi proseguendo con la lettera “B” e così via procedendo da sinistra. Il link potrebbe incrociare una WAN (Wide area network) coprendo grandi distanze. Se il link viene interrotto, le modifiche ai dati possono ancora essere effettuate in entrambi i domini. Le modifiche vengono riconciliate successivamente, quando il link riconnette i domini. I link automaticamente tentano di eseguire nuovamente la connessione se la connessione di rete viene interrotta.

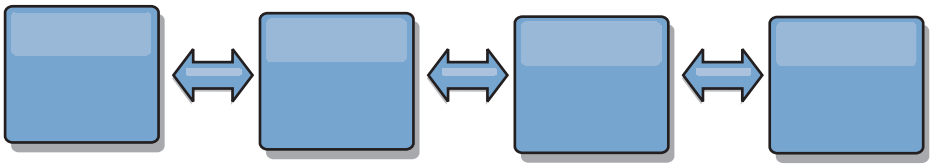


Dopo aver impostato i link eXtreme Scale tenterà prima di rendere ogni dominio identico e successivamente tenterà di mantenere le condizioni di identicità quando le modifiche avvengono in qualsiasi dominio. L'obiettivo di eXtreme Scale è per

ciascun dominio di essere un esatto specchio di ogni altro dominio connesso dai link. I link di replica tra i domini aiutano a garantire che qualsiasi modifica effettuata in un dominio venga copiata negli altri domini.

Topologie lineari

Sebbene è tra le più semplici topologie, un topologia lineare dimostra alcune qualità dei link. Prima, non è necessario per un dominio essere connesso direttamente ad ogni altro dominio per poter ricevere le modifiche. Il dominio B trarrà le modifiche dal dominio A. Il dominio C riceve le modifiche dal dominio A attraverso il dominio B che connette i domini A e C. Analogamente, il dominio D riceve le modifiche dagli altri domini attraverso il dominio C. Questa funzionalità diffonde il carico della distribuzione delle modifiche lontano dall'origine delle modifiche.



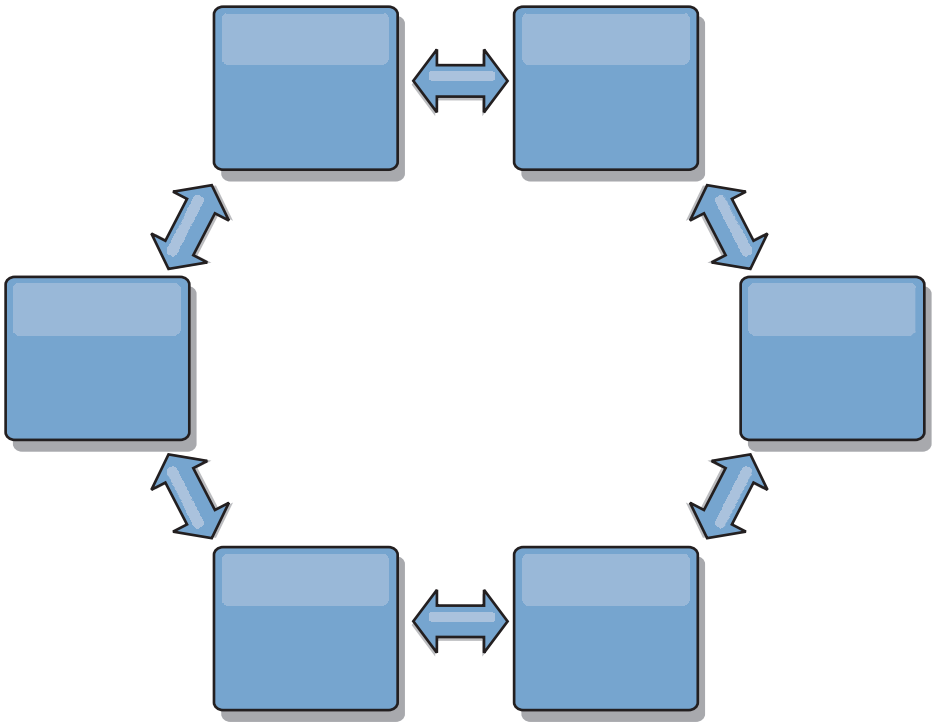
Considerare che se il dominio C è in errore, si verificano i seguenti eventi.

1. Il dominio D dovrebbe essere orfano fino a quando il dominio C non è riavviato.
2. Il dominio C dovrebbe sincronizzare se stesso con il dominio B, che è una copia del dominio A
3. Il dominio D dovrebbe utilizzare il dominio C per sincronizzare se stesso con le modifiche avvenute nei domini A e B mentre il dominio D è rimasto orfano (mentre il dominio C era giù)

Alla fine, i domini A, B, C e D dovrebbero diventare nuovamente tutti identici l'uno con gli altri.

Topologie ad anello

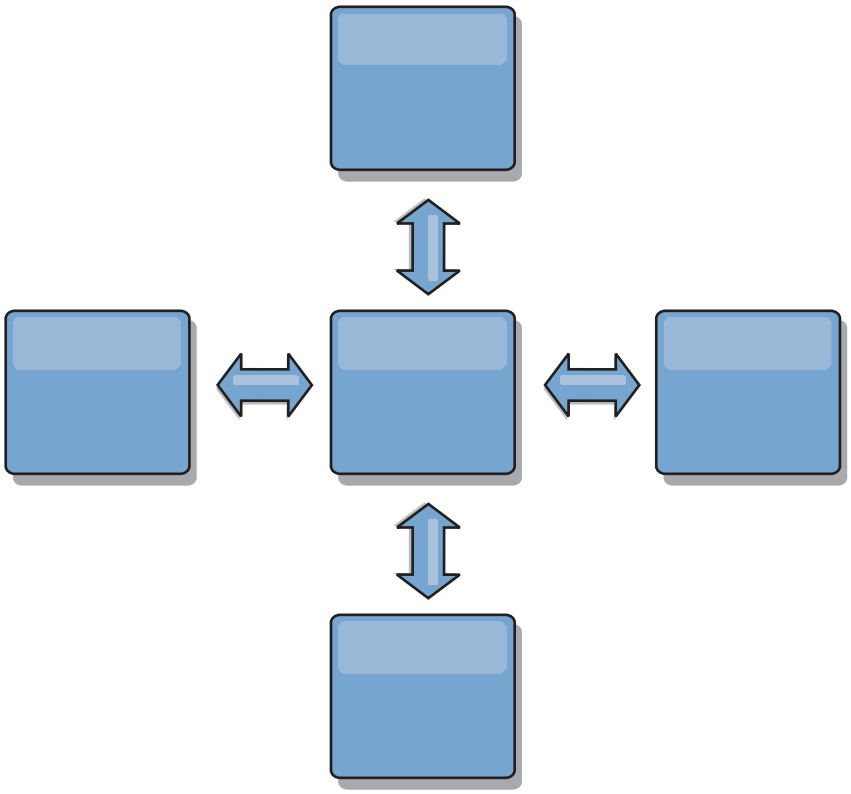
Le topologie ad anello sono un esempio di una topologia più adattabile. Sebbene un dominio o un singolo link può essere in errore, i domini superstiti possono ancora ricevere modifiche spostandosi intorno all'anello lontano dall'errore. Ciascun dominio dispone di due link agli altri domini. Ciascun dominio ha al massimo due link, non importa quanto grande sia la topologia ad anello. La latenza nel propagare le modifiche può essere considerevole poiché le modifiche da un particolare dominio potrebbero avere la necessità di spostarsi attraverso diversi domini prima che tutti i domini vedano tutte le modifiche. Una topologia lineare presenta lo stesso problema.



Immaginare una topologia ad anello più sofisticata con un dominio root al centro dell'anello. Il dominio root agisce come una clearing house centrale mentre gli altri domini agiscono come clearing house remote per le modifiche che si verificano nel dominio root. Il dominio root può arbitrare le modifiche tra i domini. Se una topologia ad anello contiene più di un anello intorno ad un dominio root, il dominio root può solo arbitrare le modifiche tra i domini nell'anello più interno. Tuttavia, i risultati dell'arbitraggio si estendono ai domini negli altri anelli.

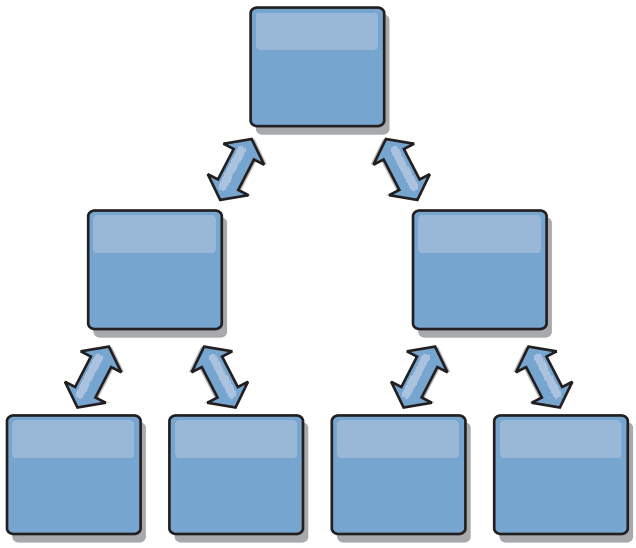
Topologie Hub e spoke

Una topologia hub e spoke ha una migliore latenza il che significa che le modifiche viaggiano attraverso al massimo un dominio intermedio (l'hub), ma presenta altri problemi. Essa ha un dominio centrale che agisce come un hub. Questo "dominio hub" è connesso ad ogni "dominio spoke" utilizzando un link. Chiaramente, l'onere della distribuzione delle modifiche tra i domini resta a carico dell'hub. L'hub agisce come una clearing house per i conflitti, un'impostazione che può essere importante in determinati scenari. In un ambiente con un elevato tasso di aggiornamento, l'hub, per essere adeguato, potrebbe avere la necessità di essere eseguito su più hardware rispetto agli spoke. eXtreme Scale è progettato per scalare in modo lineare, il che significa che è possibile ingrandire l'hub come richiesto senza difficoltà. Tuttavia, se l'hub va in errore, le modifiche non verranno distribuite fino a quando l'hub non viene riavviato. Alcune modifiche nei domini spoke verranno distribuite dopo che l'hub sia stato riconnesso.



Topologie a struttura ad albero

Un ultimo esempio di topologia è una struttura ad albero diretta non ciclica. Non ciclica significa che non esistono cicli o loop. Diretta significa che i link esistono solo tra parent e child. Questa configurazione può essere utile per topologie con così tanti domini che non è agevole avere un hub centrale connesso a ciascun spoke, o nei casi in cui si ha la necessità di avere la possibilità di aggiungere dei domini child senza aggiornare il dominio root.



Questa topologia può ancora avere una clearing house nel dominio root, ma il secondo livello può agire come clearing house remote per le modifiche che si verificano al di sotto del dominio stesso. Il dominio root può arbitrare le modifiche

tra i domini solo al secondo livello. Sono possibili anche strutture ad albero N-ary. Una struttura ad albero N-ari ha N child a ciascun livello. Ciascun dominio ha un'estensione di N.

Considerazioni sull'arbitraggio nella progettazione della topologia

I conflitti di modifica potrebbero verificarsi se gli stessi record possono essere modificati contemporaneamente in due posti. Impostare ciascun dominio in modo da avere circa la stessa quantità di CPU, memoria, risorse di rete. Si potrebbe osservare che i domini che eseguono la gestione del conflitto di modifica (arbitraggio) utilizzano più risorse degli altri domini. I conflitti vengono rilevati automaticamente. Essi vengono risolti utilizzando uno dei due meccanismi:

- **Arbitro del conflitto predefinito.** Il protocollo predefinito prevede di utilizzare le modifiche provenienti dal dominio con il nome lessicale più basso. Ad esempio, se il dominio A e il dominio B generano un conflitto per un record, le modifiche dal dominio B saranno ignorate. Il dominio A conserva la propria versione e il record nel dominio B viene modificato per corrispondere al record del dominio A. Questo si applica anche alle applicazioni in cui gli utenti o le sessioni sono normalmente legati o presentano affinità con una delle griglie.
- **Arbitro del conflitto personalizzato.** Le applicazioni possono fornire un arbitro personalizzato. Quando un dominio rileva un conflitto, esso richiama l'arbitro. Per informazioni relative allo sviluppo di un 'buon' arbitro personalizzato, consultare Sviluppo di arbitri personalizzati per la replica multi-master..

Per topologie in cui sono possibili i conflitti, considerare di utilizzare una topologia hub e spoke oppure una topologia a struttura ad albero. Queste due topologie sono idonee ad evitare continui continui il che può accadere quando:

1. più domini sperimentano un conflitto
2. ciascun dominio risolve il conflitto localmente, apportando delle correzioni
3. le correzioni entrano in conflitto producendo correzioni delle correzioni
4. e così via, poiché le correzioni si propagano tra i vari domini, tentando di realizzare la sincronia.

Per evitare continui conflitti, scegliere un dominio specifico – un *dominio di arbitraggio* – come gestore del conflitto per un sottoinsieme di domini. Ad esempio, una topologia hub e spoke potrebbe utilizzare l'hub come gestore del conflitto. Il gestore del conflitt spoke ignora qualsiasi conflitto rilevato dai domini spoke. Il dominio hub creerà le correzioni, evitando il controllo delle correzioni del conflitto. Il dominio designato a gestire i conflitti deve avere un link a tutti i domini per cui è responsabile di risolvere i conflitti. In una topologia a struttura ad albero, ogni dominio interno parent risolve i conflitti per i propri child diretti. Al contrario, se si utilizza una topologia ad anello, non è possibile designare un dominio nell'anello come dominio risolutore dei conflitti.

La tabella di seguito riportata riepiloga gli approcci di arbitraggio che sono maggiormente compatibili con le varie topologie. .

Tabella 3. Approcci di arbitraggio. Questa tabella stabilisce se l'arbitraggio dell'applicazione è compatibile con le varie tecnologie.

Topologia	Arbitraggio dell'applicazione	Note
Una linea di due domini	Sì	Scegliere un dominio come arbitro.

Tabella 3. Approcci di arbitraggio (Continua). Questa tabella stabilisce se l'arbitraggio dell'applicazione è compatibile con le varie tecnologie.

Topologia	Arbitraggio dell'applicazione	Note
Una linea di tre domini	Sì	Il dominio in mezzo deve essere l'arbitro. Pensare al dominio in mezzo come l'hub in una semplice topologia hub e spoke.
Una linea di più di tre domini	Sì	L'arbitraggio dell'applicazione non è supportato.
Un hub con N spoke	Sì	L'hub con i link a tutti gli spoke deve essere il dominio di arbitraggio.
Un anello di N domini	Sì	L'arbitraggio dell'applicazione non è supportato
Una struttura ad albero diretta, non ciclica (struttura ad albero N-ari)	Sì	Tutti i nodi root devono aribitrare solo i loro discendenti diretti.

Considerazioni sui link nella progettazione della topologia

Idealmente, una topologia comprende il minimo numero di link mentre l'ottimizzazione cerca un compromesso tra la latenza della modifica, la tolleranza di errori e le caratteristiche delle prestazioni.

- **Latenza di modifica**

  La latenza di modifica viene determinata dal numero di domini intermedi che una modifica deve attraversare prima di arrivare ad un dominio specifico.

  Una topologia ha la migliore latenza di modifica quando essa elimina i domini intermedi creando dei link per ciascun dominio ad ogni altro dominio. Tuttavia, un dominio deve eseguire l'attività di replica in proporzione al relativo numero di link. Per topologie larghe, il picco di link da definire potrebbe comportare un onere amministrativo.

  La velocità con cui una modifica viene copiata negli altri domini dipende da ulteriori fattori tra i quali:

  – CPU e larghezza di banda della rete nel dominio origine
  – il numero di domini intermedi e i link tra il dominio origine e il dominio di destinazione
  – la CPU e le risorse di rete disponibili nei domini origine, di destinazione e intermedi.

- **Tolleranza degli errori**

  La tolleranza degli errori è determinata dal numero di percorsi esistenti tra i due domini per la replica della modifica.

  Se esiste un singolo link tra i domini, se il link va in errore, le modifiche non verranno propagate. Se il singolo link da un dominio ad un altro dominio passa attraverso dei domini intermedi, le modifiche non verranno propagate se qualsiasi dominio intermedio è giù.

  Considerare la topologia lineare con tre domini A, B e C:

  ```
  A <-> B <-> C
  ```

  Se perdura qualsiasi di queste condizioni, il dominio C non vedrà le modifiche del dominio A:

  – Il dominio A è su e il dominio B è giù

– Il link tra A e B è giù
– Il link tra B e C è giù

Al contrario, una topologia ad anello consente a ciascun dominio di trarre le modifiche da altre direzioni.

```
A <-> B <-> C <-> back to A
```

Ad esempio, se il dominio B è giù, il dominio C ancora può trarre le modifiche direttamente dal dominio A.

Una progettazione hub e spoke è suscettibile di far andare giù l'hub poiché sono forzate a passare attraverso l'hub stesso. Tuttavia, è meritevole ricordare che un singolo dominio è ancora una griglia completamente tollerante agli errori grossolani che potrebbero presentarsi come problemi del centro dati fisico o WAN.

- **Prestazioni**

  Il numero di link definiti in un dominio influisce sulle prestazioni. Più link utilizzano più risorse e il risultato potrebbe essere un calo delle prestazioni. La possibilità di trarre le modifiche per un dominio A attraverso gli altri domini effettivamente riduce il carico del dominio A nelle repliche delle relative transazioni ovunque. *Il carico di distribuzione della modifica in un dominio è limitato al numero di link che esso utilizza. Esso non ha niente a che fare con quanti domini sono presenti nella topologia.* Questa proprietà fornisce la scalabilità e consente all'onere della distribuzione della modifica di essere condiviso tra i domini nella topologia, piuttosto che caricare l'onere su un singolo dominio.

  Un dominio può trarre le modifiche indirettamente attraverso altri domini. Considerare una topologia lineare con cinque domini.

  ```
  A <=> B <=> C <=> D <=> E
  ```

  – A trae le modifiche da B, C, D ed E attraverso B
  – B trae direttamente le modifiche da A e C e le modifiche da D ed E attraverso C.
  – C trae direttamente le modifiche da B e D e le modifiche da A attraverso B ed E attraverso D.
  – D trae direttamente le modifiche da C ed E e le modifiche da A e B attraverso C.
  – E trae direttamente le modifiche da D e le modifiche da A, B e C attraverso D.

  Il carico di distribuzione nei domini A ed E è più basso perché ciascuno di essi ha un solo link ad un singolo dominio. Il carico di distribuzione nei domini B, C, e D è doppio del carico nei domini A ed E perché ciascuno dei domini B, C e D hanno un link a due domini. Questa distribuzione di carichi potrebbe rimanere costante anche se la linea contenesse 1000 domini perché il carico dipende dal numero di link di ciascun dominio e non dal numero complessivo di domini nella topologia.

Considerazioni sulla prestazione

Considerare le limitazioni di seguito riportate quando si utilizzano le topologie di replica multi-master.

- **Ottimizzazione della distribuzione della modifica** (trattata in precedenza)
- **Latenza della replica** (trattata in precedenza)
- **Le prestazioni del link di replica** eXtreme Scalecreano un singolo socket TCP/IP tra ogni coppia di JVM. Tutto il traffico tra queste JVM si verifica sopra quel socket compreso la replica multi-master. Poiché i domini vengono ospitati

su almeno N JVM contenitori fornendo almeno N link TCP per domini peer, i domini con il maggior numero di contenitori avrà livelli di prestazioni della replica più elevati. Più contenitori significa più CPU e risorse di rete.

- **Il supporto dell'ottimizzazione di 'sliding window ' TCP e l'** Abilitazione di RFC 1323 su entrambi gli estremi di un link consente di avere più dati in andata e ritorno (round trip) risultando in un livello di prestazioni più elevato. La tecnica espande la capacità della finestra di un fattore pari a circa 16,000.

  Ricordarsi che i socket TCP utilizzano il meccanismo 'sliding window' per controllare il flusso di masse di dati il che tipicamente limita il socket a 64 KB per un intervallo di andata e ritorno (round trip). Se l'intervallo di andata e ritorno è 100 ms, la larghezza di banda è limitata a 640 KB al secondo senza ulteriore ottimizzazione. L'utilizzo completo della larghezza di banda disponibile su un link potrebbe richiedere un'ottimizzazione specifica in base al sistema operativo. La maggior parte di sistemi operativi prevede i parametri di ottimizzazione incluso le opzioni RFC 1323 per migliorare la velocità di trasmissione per i link ad elevata latenza.

  Diversi fattori possono influire sulle prestazioni della replica

  – La velocità a cui eXtreme Scale può trarre le modifiche.
  – La velocità a cui eXtreme Scale può soddisfare le richieste di replica.
  – La capacità 'sliding window'
  – L'ottimizzazione del buffer di rete su entrambi i lati di un link per consentire a eXtreme Scale di trarre le modifiche sopra i socket in modo più veloce possibile.
- **Serializzazione dell'oggetto** Tutti i dati devono essere serializzabili. Se un dominio non utilizza COPY_TO_BYTES, il dominio deve utilizzare la serializzazione Java oppure ObjectTransformers per ottimizzare le prestazioni della serializzazione.
- **Compressione** eXtreme Scale per impostazione predefinita, comprime tutti i dati inviati tra i domini. Non esiste un'opzione per disabilitare la compressione nell'attuale release.
- **Ottimizzazione della memoria** *L'utilizzo della memoria per una topologia di replica multi-master è largamente indipendente dal numero di domini nella topologia.*

  L'abilitazione della replica multi-master aggiunge un sovraccarico fisso per la voce della mappa per gestire il controllo di versioni. Ciascun contenitore tiene anche traccia di un quantitativo fisso di dati per ciascun dominio nella topologia. Una topologia con due domini utilizza approssimativamente la stessa memoria di una topologia con 50 domini. eXtreme Scale non utilizza log di ripetizione o code simili nella sua implementazione il che significa che se un link di replica non è disponibile per un intervallo considerevole di tempo, non si verifica la crescita della dimensione della struttura dati, piuttosto resta in attesa di riprendere la replica quando il link si riavvia.

Più centri dati con FIXED_PARTITION

Ora è possibile utilizzare una griglia FIXED_PARTITION tra due o più centri dati. Ciascun centro dati ha la necessità di avere il proprio dominio, in termini di replica multi-master. Ciascun centro dati può leggere e scrivere i dati rispetto al dominio locale. Queste modifiche si propagheranno agli altri centri dati utilizzando i link che sono stati definiti.

Client completamente replicati

Questa variazione nella topologia interessa una coppia di server eXtreme Scale che sono in esecuzione come hub. Ciascun client crea una singola griglia contenitore autonoma con un catalogo nella JVM client. Un client utilizza la propria griglia per connettersi al catalogo hub causando che il client si sincronizza con l'hub appena ottiene una connessione all'hub.

Alcune modifiche effettuate dal client sono locali al client e vengono replicate in modo asincrono all'hub. L'hub agisce come dominio di arbitraggio distribuendo le modifiche a tutti i client connessi. La topologia dei client completamente replicati fornisce una buona cache L2 per il programma di definizione relazionale dell'oggetto come OpenJPA. Le modifiche verranno distribuite velocemente tra le JVM client attraverso l'hub. Finché la dimensione della cache può essere contenuta entro lo spazio dell'heap disponibile dei client, questa topologia costituisce una buona architettura per questo stile L2.

Utilizzare, se necessario, più partizioni per scalare il dominio hub su più JVM. Poiché tutti i dati ancora devono adattarsi ad una singola JVM client , utilizzando più partizioni si incrementa la capacità dell'hub di distribuire ed arbitrare le modifiche, ma esso non modifica la capacità di un singolo dominio.

Limitazioni

Considerare le seguenti limitazioni quando si decide se e in che modo utilizzare le topologie di replica multi-master.

- **Considerare la configurazione dei programmi di caricamento della classe con più domini.**

  I domini devono avere accesso a tutte le classi che vengono utilizzate come chiavi e valori. Qualsiasi dipendenza deve essere riflessa in tutti i percorsi di classe per le JVM del contenitore della griglia per tutti i domini. Se un plug-in CollisionArbiter recupera il valore per una voce di cache, le classi per i valori devono essere presenti nel dominio che sta richiamando l'arbitro.

- **L'utilizzo dei programmi di caricamento non è consigliato**

  I programmi di caricamento possono essere utilizzati per interfacciare le modifiche tra una griglia e un database. >È improbabile che tutte le griglie (domini) in una topologia siano collocate geograficamente con lo stesso database. La latenza WAN e gli altri fattori potrebbero rendere questo utilizzo non desiderabile.

  Il precaricamento della griglia è un altro problema che richiede una progettazione attenta. Generalmente, quando una griglia è riavviata, essa viene nuovamente precaricata. Il precaricamento non è necessario o addiruttura desiderabile allorché si utilizza una replica multi-master. Appena un dominio è in linea, esso automaticamente ricarica se stesso con i contenuti del dominio a cui è collegato. Di conseguenza, non è necessario inizializzare un precaricamento manuale per una griglia che è un dominio in una topologia di replica multi-master.

  I programmi di caricamento generalmente rispettano le regole di inserimento ed aggiornamento. Utilizzando la replica multi-master, gli inserimenti devono essere trattati come unioni. Quando i dati sono tratti in remoto dopo che un dominio è stato riavviato, i dati esistenti saranno ‘inseriti' nel dominio locale. Poiché questi dati potrebbero già essere presenti nel database locale, un tipico inserimento potrebbe andare in errore con un'eccezione di chiave duplicata nel database. Invece devono essere utilizzate le semantiche di unione.

eXtreme Scale può essere configurato per eseguire un precaricamento basato sul frammento utilizzando i metodi di precaricamento nei plug-in Loader. Non utilizzare questa tecnica nella topologia di replica multi-master. Invece, utilizzare un precaricamento basato sul client quando la topologia è avviata (inizialmente). Consentire alla topologia multi-master di aggiornare qualsiasi dominio riavviato con una copia corrente di quanto è stato memorizzato in altri domini nella topologia. Dopo che i domini sono stati avviati, la topologia multi-master è responsabile di mantenerli in sincronia.

- **Non è supportato EntityManager**

  Una serie di mappe contenenti un'entità mappa non è replicata tra i domini.
- **Le mappe array di byte non sono supportate**

  Una serie di mappe contenenti una mappa che è configurata con COPY_TO_BYTES non è replicata tra i domini.
- **Write-behind non è supportato**

  Una serie di mappe contenenti una mappa che è configurata con il supporto write-behind non è replicata tra i domini.

Integrazione del database: Cache write-behind, in-line e side

WebSphere eXtreme Scale viene utilizzato per far fronte ad un database tradizionale e per eliminare attività di lettura inviate normalmente al database. Una cache coerente può essere utilizzata con un'applicazione che utilizza direttamente o indirettamente un programma di associazione relazionale di oggetti. La cache coerente può scaricare il database o il backend dalle letture. In uno scenario leggermente più complesso, come un accesso transazionale ad una serie di dati in cui alcuni dati richiedono garanzie di tradizionale persistenza, l'uso di filtri può essere utilizzato per scaricare anche le transazioni in scrittura.

È possibile configurare eXtreme Scale in modo che funzioni come uno spazio di elaborazione del database in memoria altamente flessibile. Tuttavia, eXtreme Scale non è un ORM (object relational mapper). Non sa da dove provengono i dati in eXtreme Scale. Un'applicazione o un ORM può collocare dati in un server eXtreme Scale. È responsabilità del sorgente dati assicurarsi che sia coerente con il database da cui hanno origine i dati. Questo vuol dire che eXtreme Scale non può invalidare i dati presi automaticamente da un database. Questa applicazione o programma di associazione deve fornire questa funzione e gestire i dati memorizzati in eXtreme Scale.

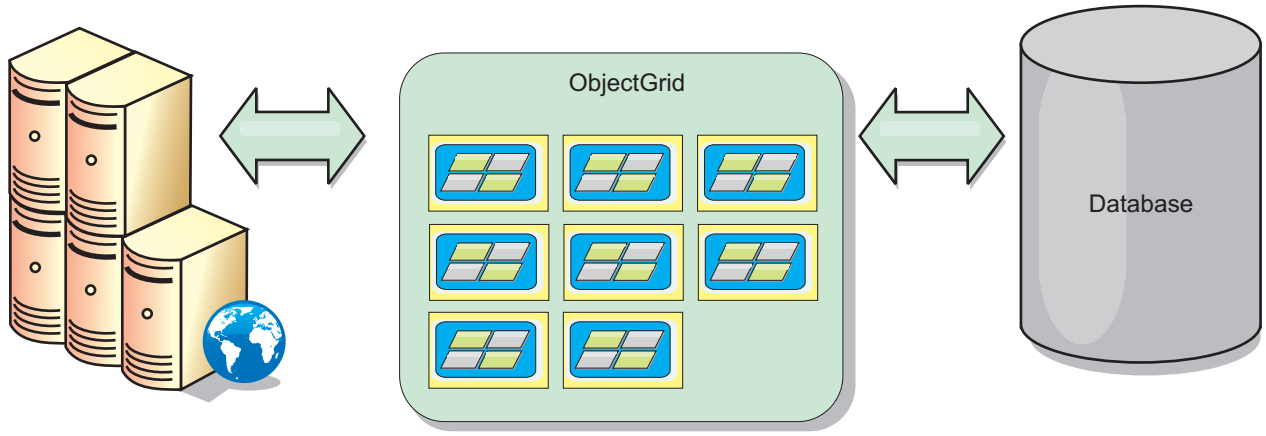


Figura 17. ObjectGrid come un buffer del database

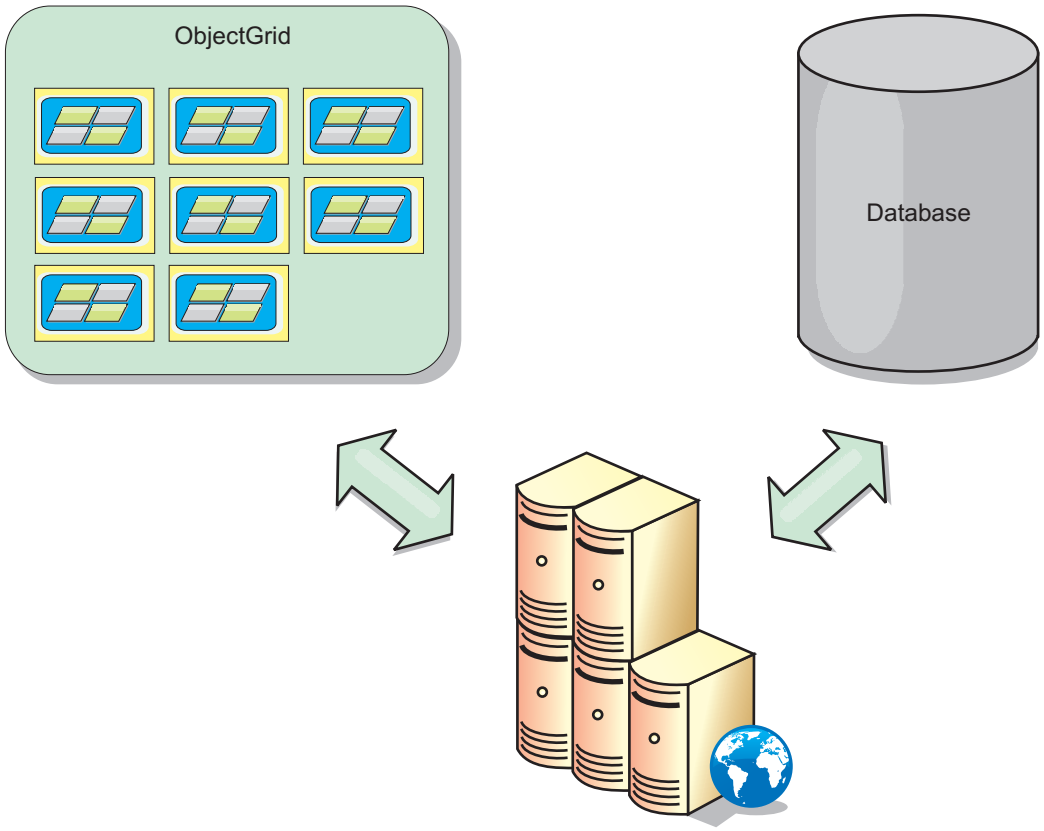


Figura 18. ObjectGrid come una side cache

Cache limitata e cache completa

WebSphere eXtreme Scale può essere utilizzato come cache o cache completa. Una cache limitata conserva solo un sottoinsieme dei dati totali, mentre una cache completa conserva tutti i dati e può essere popolata in modalità lazy su richiesta. Normalmente per accedere alle cache limitate si utilizzano le chiavi (invece di indici o query) in quanto i dati sono disponibili solo parzialmente.

Quando non è presente una chiave (una mancata corrispondenza nella cache), viene richiamato il livello successivo e i dati vengono recuperati e inseriti nel livello della rispettiva cache. Se si utilizza una query o un indice, vengono acceduti solo i valori caricati attualmente e le richieste non vengono inoltrate agli altri livelli. Una cache completa contiene tutti i dati richiesti e può essere acceduta utilizzando attributi non di chiavi con indici o query.

Una cache completa viene precaricata con i dati prima dell'utilizzo da parte delle applicazioni, e può funzionare in modo efficace come sostituzione del database. Dopo il caricamento dei dati, può essere trattata in modo simile a un database. Poiché tutti i dati sono disponibili, le query e gli indici possono essere utilizzati per trovare ed aggregare dati.

Side cache e in-line cache

WebSphere eXtreme Scale viene utilizzato utilizzato per fornire la memorizzazione nella in-line cache per un back-end del database oppure come side cache per un database. La memorizzazione nella in-line cache utilizza eXtreme Scale come

mezzo principale per interagire con i dati. Quando eXtreme Scale viene utilizzato come side cache, il back-end viene utilizzato insieme a eXtreme Scale.

Side cache

eXtreme Scale può essere utilizzato come side-cache per un DAL (Data Access Layer) di un'applicazione. In questo scenario eXtreme Scale viene utilizzato per memorizzare temporaneamente gli oggetti che sarebbero normalmente recuperati da un database di back-end. Le applicazioni controllano se eXtreme Scale contiene i dati desiderati. Se i dati sono presenti, vengono restituiti al chiamante. Se i dati non sono presenti, vengono recuperati dal back-end e inseriti in eXtreme Scale in modo che la richiesta successiva possa utilizzare la copia memorizzata nella cache. Il diagramma di seguito riportato illustra il modo in cui eXtreme Scale può essere utilizzato come side-cache con un Data Access Layer arbitrario come ad esempio OpenJPA o Hibernate.

Plug-in side cache per Hibernate e OpenJPA

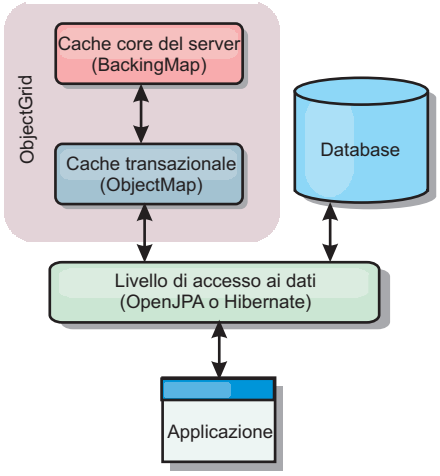


Figura 19. Side cache

I plug-in della cache per OpenJPA e Hibernate sono inclusi in eXtreme Scale, il che consente di utilizzare eXtreme Scale come una side-cache automatica. L'utilizzo di eXtreme Scale come provider della cache migliora le prestazioni nella lettura e nell'esecuzione di query sui dati e riduce il caricamento sul database. eXtreme Scale ha dei vantaggi sulle implementazioni di cache incorporate, poiché la cache viene replicata automaticamente tra tutti i processi. Quando un client memorizza un valore nella cache, tutti gli altri client potranno utilizzare il valore memorizzato nella cache.

In-line cache

Quando utilizzato come in-line cache, eXtreme Scale interagisce con il back-end utilizzando un plug-in Loader. Questo scenario può semplificare l'accesso ai dati consentendo alle applicazioni di accedere direttamente alle API eXtreme Scale. In eXtreme Scale sono supportati diversi scenari di memorizzazione nella cache per assicurare che i dati nella cache e i dati nel back-end siano sincronizzati. Il diagramma di seguito riportato illustra il modo in cui una in-line cache interagisce con l'applicazione e il back-end.

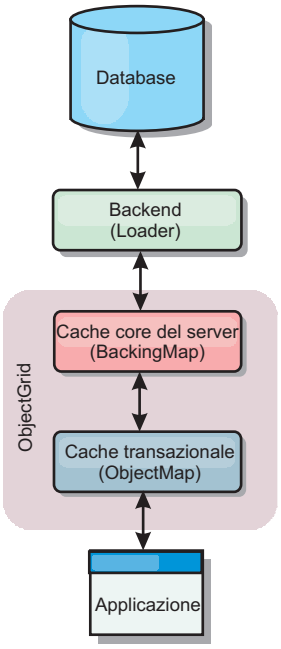


Figura 20. In-line cache

Cache in linea

La memorizzazione nella cache in linea utilizza eXtreme Scale come mezzo principale per interagire con i dati. Quando eXtreme Scale viene utilizzato come cache in linea, l'applicazione interagisce con il back-end utilizzando un plug-in Loader.

L'opzione di memorizzazione nella cache in linea semplifica l'accesso ai dati in quanto consente alle applicazioni di accedere direttamente alle API di eXtreme Scale. WebSphere eXtreme Scale supporta diversi scenari di memorizzazione nella cache in linea, riportati di seguito.

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

Scenario di memorizzazione nella cache read-through

Una cache read-through è una cache limitata che carica le immissioni di dati in modalità lazy in base alla chiave quando vengono richieste. Questa operazione viene effettuata senza che il chiamante debba sapere come vengono popolate le voci. Se non è possibile trovare i dati nella cache eXtreme Scale, eXtreme Scale richiamerà i dati mancanti dal plug-in Loader, che carica i dati dal database back-end e li inserisce nella cache. Le successive richieste della stessa chiave di dati verranno trovate nella cache fino a quando non verranno rimossi, invalidati o eliminati.

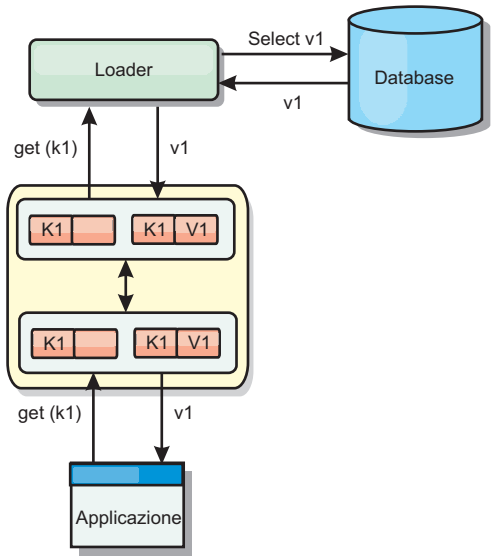


Figura 21. Memorizzazione nella cache read-through

Scenario di memorizzazione nella cache write-through

In una cache write-through, ogni scrittura nella cache scrive in modo sincrono nel database utilizzando Loader. Questo metodo fornisce la congruenza con il back-end, ma diminuisce le prestazioni della scrittura in quanto l'operazione del database è sincrona. Poiché la cache e il database vengono entrambi aggiornati, le successive letture degli stessi dati verranno individuate nella cache, evitando la chiamata al database. Una cache write-through viene spesso utilizzata insieme ad una cache read-through.

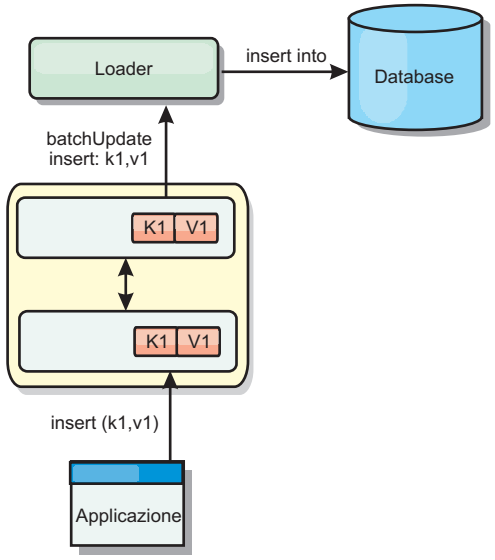


Figura 22. Memorizzazione nella cache write-through

Scenario di memorizzazione nella cache write-behind

La sincronizzazione del database può essere migliorata scrivendo le modifiche in modo asincrono. Ciò è noto come cache write-behind o write-back. Le modifiche che normalmente verrebbero scritte in modo sincrono nel programma di

caricamento, vengono invece inserite come buffer in eXtreme Scale e scritte nel database utilizzando un thread di sfondo. Le prestazioni della scrittura vengono migliorate in modo significativo in quanto l'operazione del database viene rimossa dalla transazione del client e le scritture del database possono essere compresse. Per ulteriori informazioni consultare “Cache write-behind”.

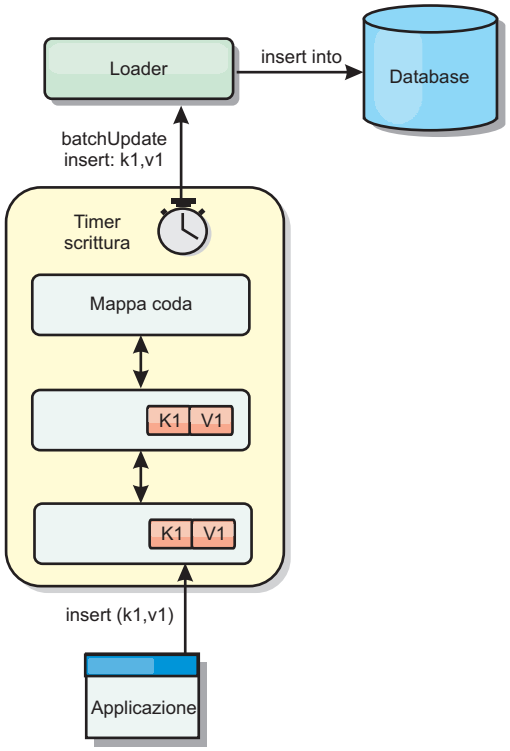


Figura 23. Memorizzazione nella cache write-behind

Per ulteriori informazioni consultare “Cache write-behind”.

Cache write-behind

È possibile utilizzare la cache write-behind per ridurre il sovraccarico che si verifica durante l'aggiornamento di un database che si sta utilizzando come back-end.

Introduzione

La cache write-behind accoda in modo asincrono gli aggiornamenti nel plug-in Loader. È possibile migliorare le prestazioni scollegando gli aggiornamenti, gli inserimenti e le rimozioni per una mappa, il sovraccarico dell'aggiornamento del database di back-end. L'aggiornamento asincrono viene eseguito dopo un ritardo basato sull'orario (ad esempio, cinque minuti) o un ritardo basato sulle voci (1000 voci).

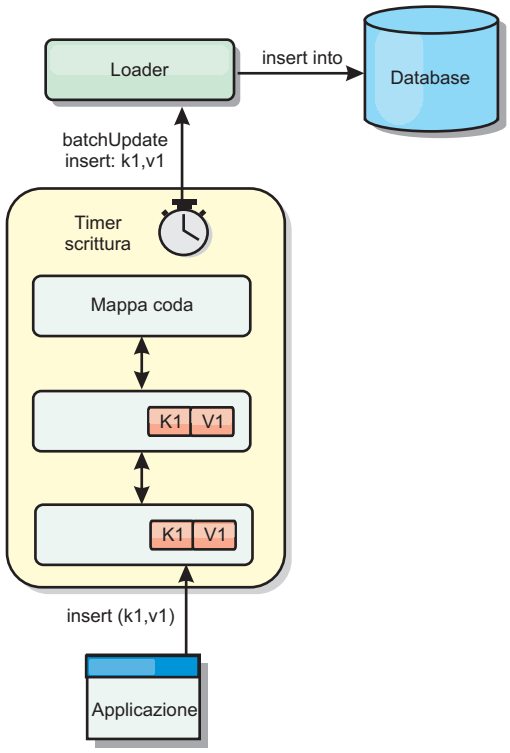


Figura 24. Memorizzazione nella cache write-behind

La configurazione write-behind su BackingMap crea un thread tra il programma di caricamento e la mappa. Il programma di caricamento delega quindi le richieste di dati tramite il thread in base alle impostazioni di configurazione nel metodo BackingMap.setWriteBehind. Quando una transazione eXtreme Scale inserisce, aggiorna o rimuove una voce da una mappa, viene creato un oggetto LogElement per ciascuno di questi record. Questi elementi vengono inviati al programma di caricamento write-behind e accodati in un ObjectMap speciale, chiamato mappa di coda. Ogni mappa di backup con l'impostazione write-behind abilitata dispone di proprie mappe di coda. Un thread write-behind rimuove periodicamente i dati accodati dalle mappe di coda e li inserisce nel programma di caricamento di back-end effettivo.

Il programma di caricamento write-behind invia solo tipi di inserimento, aggiornamento e eliminazione degli oggetti LogElement al programma di caricamento effettivo. Tutti gli altri tipi di oggetto LogElement, ad esempio, tipo EVICT, vengono ignorati.

Vantaggi

L'abilitazione del supporto write-behind presenta i seguenti vantaggi:

- **Isolamento degli errori di back-end:** la cache write-behind fornisce un livello di isolamento dagli errori di back-end. Quando il database di back-end non funziona correttamente, gli aggiornamenti vengono accodati nella mappa di coda. L'applicazione può proseguire inviando le transazioni a eXtreme Scale. Quando viene ripristinato il back-end, i dati nella mappa di coda vengono inviati al back-end.

- **Caricamento di back-end ridotto:** il programma di caricamento write-behind unifica gli aggiornamenti in base a una chiave così esiste un solo aggiornamento unificato per chiave nella mappa di coda. Questa unione riduce il numero di aggiornamenti al database di back-end.
- **Prestazioni di transazione migliorata:** vengono ridotte le ore delle singole transazioni eXtreme Scale poiché per la transazione non è necessario attendere la sincronizzazione dei dati con il back-end.

Considerazioni sulla progettazione delle applicazioni

L'abilitazione del supporto write-behind è semplice, ma per progettare l'utilizzo di un'applicazione con il supporto write-behind sono necessarie alcune considerazioni. Senza il supporto write-behind, la transazione ObjectGrid racchiude la transazione back-end. La transazione ObjectGrid viene avviata prima della transazione di back-end e termina dopo.

Con il supporto write-behind abilitato, la transazione ObjectGrid termina prima dell'avvio della transazione back-end. La transazione ObjectGrid e quella back-end vengono divise.

Vincoli di integrità referenziale

Ogni mappa di backup, che viene configurata con il supporto write-behind, dispone del proprio thread write-behind per inserire i dati nel back-end. Pertanto i dati, aggiornati in mappe differenti in una transazione ObjectGrid, vengono aggiornati nel back-end in differenti transazioni back-end. Ad esempio, la transazione T1 aggiorna la chiave key1 nella mappa Map1 e la chiave key2 nella mappa Map2. L'aggiornamento di key1 nella mappa Map1 viene aggiornato nel back-end in un'altra transazione back-end e la chiave key2 aggiornata nella mappa Map2 viene aggiornata nel back-end in un'altra transazione back-end da thread write-behind differenti. Se i dati memorizzati in Map1 e Map2 possiedono delle relazioni, ad esempio vincoli di chiavi esterne nel back-end, gli aggiornamenti potrebbero non riuscire correttamente.

Quando si progettano vincoli di integrità referenziale nel database di back-end, assicurarsi che siano consentiti gli aggiornamenti non funzionanti.

Comportamento di blocco della mappa di coda

Un'altra differenza principale del comportamento della transazione è il comportamento di blocco. ObjectGrid supporta tre strategie differenti di blocco: PESSIMISTIC, OPTIMISITIC, e NONE. Le mappe di coda write-behind utilizzando la strategia di blocco pessimistico indipendentemente da quale strategia di blocco viene configurata per la relativa mappa di backup. Vi sono due differenti tipi di operazioni che acquisiscono un blocco sulla mappa di coda:

- Quando si esegue il commit di una transazione ObjectGrid o si verifica un flush (flush di mappa o di sessione), la transazione legge la chiave nella mappa di coda e pone un blocco S sulla chiave.
- Quando viene eseguito il commit di una transazione ObjectGrid, la transazione cerca di aggiornare il blocco S nel blocco X sulla chiave.

A causa di questo comportamento della coda di mappa particolare, è possibile notare alcune differenze nel comportamento di blocco.

- Se la mappa utente è configurata come strategia di blocco PESSIMISTIC, non c'è tanta differenza nel comportamento di blocco. Ogni volta che viene richiamato

un flush o un commit, viene posto un blocco S sulla stessa chiave nella mappa di coda. Durante il periodo di commit, non viene acquisito solo un blocco X per la chiave nella mappa utente, ma viene anche acquisito per la chiave nella mappa di coda.

- Se la mappa utente è configurata come strategia di blocco OPTIMISTIC o NONE, la transazione utente seguirà il pattern della strategia di blocco PESSIMISTIC. Ogni volta che viene richiamato un flush o un commit, viene acquisito un blocco S per la stessa chiave nella mappa di coda. Durante il periodo di commit, viene acquisito un blocco X per la chiave nella mappa di coda utilizzando la stessa transazione.

Nuovi tentativi di transazione del programma di caricamento

ObjectGrid non supporta transazioni a due fasi o XA. Il thread write-behind rimuove i record dalla mappa di coda e aggiorna i record nel back-end. Se il server non riesce correttamente nel corso della transazione, è possibile che si perdano alcuni aggiornamenti back-end.

Il programma di caricamento write-behind riproverà automaticamente a scrivere le transazioni non riuscite e invierà un LogSequence dubbio al back-end per evitare la perdita di dati. Questa azione richiede che il programma di caricamento sia idempotente, cioè, quando Loader.batchUpdate(TxId, LogSequence) viene richiamato due volte con lo stesso valore, fornisce lo stesso risultato come se fosse stato applicato una volta sola. Le implementazioni del programma di caricamento devono implementare l'interfaccia RetryableLoader per abilitare questa funzione. Per ulteriori dettagli consultare la documentazione API.

Errori di programma di caricamento

Il plug-in Loader può non riuscire quando non è in grado di comunicare con il back-end del database. Ciò può accadere se il server del database o la connessione di rete non è attiva. Il programma di caricamento write-behind accoderà gli aggiornamenti e cercherà di inviare le modifiche dei dati al programma di caricamento periodicamente. Il programma di caricamento deve notificare al runtime di ObjectGrid che vi è un problema di connettività del database generando un'eccezione LoaderNotAvailableException.

Pertanto, l'implementazione Loader deve essere in grado di distinguere un errore di dati da un errore fisico del programma di caricamento. L'errore di dati deve essere generato o generato nuovamente come LoaderException o OptimisticCollisionException, ma un errore fisico del programma di caricamento deve essere generato o generato nuovamente come un LoaderNotAvailableException. ObjectGrid gestisce queste due eccezioni in modo differente:

- Se un LoaderException viene rilevato dal programma di caricamento write-behind, il programma di caricamento write-behind riterrà che non funziona correttamente a causa di alcuni dati errati, ad esempio, un errore di chiave duplicata. Il programma di caricamento write-behind annullerà il batch dell'aggiornamento e cercherà di aggiornare un record alla volta per isolare l'errore di dati. Se A {{LoaderException}} viene rilevato nuovamente durante l'aggiornamento di un record, viene creato un record di aggiornamento non riuscito e registrato nella mappa di aggiornamento non riuscito.
- Se viene rilevata un'eccezione LoaderNotAvailableException dal programma di caricamento write-behind, il programma di caricamento write-behind riterrà che non funziona correttamente poiché non riesce a connettersi con il lato database,

ad esempio, il back-end del database non è attivo, una connessione di database non è disponibile o la rete non è attiva. Il programma di caricamento write-behind attenderà per 15 secondi e poi ritenterà l'aggiornamento batch al database.

Un errore comune è generare LoaderException mentre deve essere generato LoaderNotAvailableException. Tutti i record accodati nel programma di caricamento write-behind diventeranno record di aggiornamento non riusciti, cosa che compromette il proposito di isolare gli errori di back-end.

Considerazioni sulle prestazioni

Il supporto della cache write-behind aumenta il tempo di risposta rimuovendo l'aggiornamento del programma di caricamento dalla transazione. Inoltre, aumenta il livello di prestazione del database poiché gli aggiornamenti del database sono combinati. È importante comprendere il sovraccarico presentato dal thread write-behind, che estrae i dati della mappa di coda e li inserisce nel programma di caricamento.

Il numero massimo di aggiornamenti o il tempo massimo di aggiornamento deve essere regolato in base ai pattern e all'ambiente di utilizzo previsti. Se il valore del numero massimo di aggiornamenti o il tempo massimo di aggiornamenti è troppo piccolo, è possibile che il sovraccarico del thread write-behind superi i vantaggi. Impostando un valore ampio per questi due parametri è possibile anche aumentare l'utilizzo della memoria per l'accodamento dei dati e aumentare il ritardo di aggiornamento dei record di database.

Per prestazioni ottimali, mettere a punto i parametri write-behind in base ai seguenti fattori:

- Rapporto di transazioni in lettura e scrittura
- Stessa frequenza di aggiornamento record
- Latenza di aggiornamento del database.

Programmi di caricamento

Con un plug-in Loader eXtreme Scale una mappa eXtreme Scale può comportarsi come una cache di memoria per i dati che generalmente sono conservati nell'archivio persistente sia sullo stesso sistema che su un altro sistema. Generalmente, un database o un file system viene utilizzato come archivio persistente. Una JVM (Java virtual machine) può anche essere utilizzata come origine dei dati consentendo alle cache basate sugli hub di essere costruite utilizzando eXtreme Scale. Un programma di caricamento dispone della logica per leggere e scrivere dati in e da un archivio persistente.

Panoramica

I programmi di caricamento sono plug-in di mappe di backup che vengono richiamati quando sono effettuate modifiche alla mappa di backup o quando la mappa di backup non è in grado di soddisfare la richiesta dati (una mancata corrispondenza nella cache). Il programma di caricamento è richiamato quando la cache non è in grado di soddisfare una richiesta per una chiave fornendo capability attraverso la lettura e il popolamento lazy della cache. Un programma di caricamento consente anche aggiornamenti al database quando cambiano i valori della cache. Tutte le modificano all'interno di una transazione vengono raggruppate insieme per consentire di minimizzare il numero di interazioni di database. Un plug-in TransactionCallback viene utilizzato insieme al programma di

caricamento per attivare la demarcazione della transazione di backend. L'utilizzo di questo plug-in è importante quando più mappe sono incluse in una singola transazione o quando i dati della transazione vengono distribuite alla cache senza eseguire il commit.

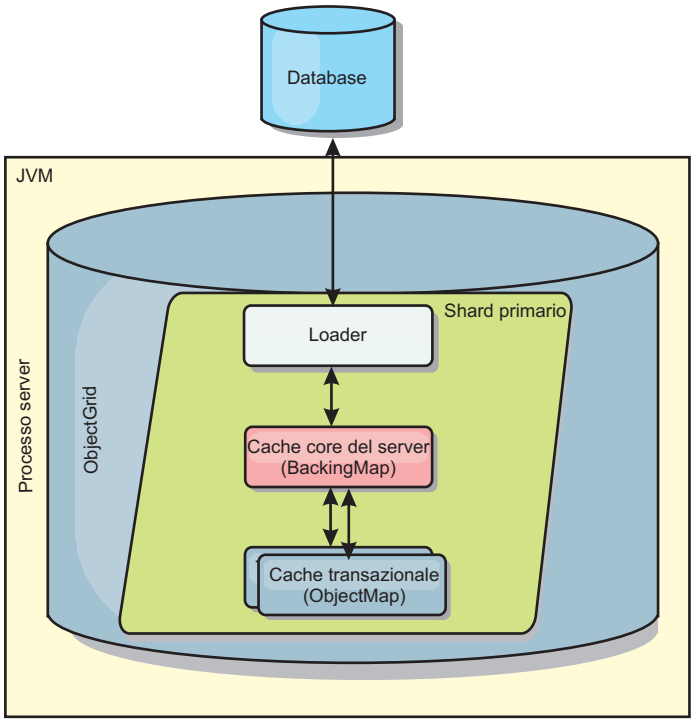


Figura 25. Programma di caricamento

Il programma di caricamento può utilizzare anche aggiornamenti sovraqualificati per evitare di mantenere il database bloccato. Memorizzando un attributo versione nel valore della cache, il programma di caricamento può vedere l'immagine prima e dopo del valore quando questo valore è aggiornato nella cache. Questo valore successivamente può essere utilizzato durante l'aggiornamento del database o del back end per verificare che i dati non sono stati aggiornati. Un programma di caricamento può anche essere configurato per precaricare una griglia quando è avviato. Quando ripartito, un'istanza del programma di caricamento viene associata a ciascuna partizione. Se la mappa "Company" ha dieci partizioni, esistono dieci istanze del programma di caricamento una per partizione dell'elemento primario. Quando è attivato il frammento dell'elemento primario per la mappa, il metodo preloadMap per il programma di caricamento viene richiamato in modalità sincrona o asincrona il che consente che il caricamento della partizione della mappa con i dati dal back-end si verifichi automaticamente. Quando viene richiamato in modalità sincrona, tutte le transazioni dei client vengono bloccate evitando accesso non coerente all'intera griglia. In alternativa, un programma di precaricamento del client può essere utilizzato per caricare l'intera griglia.

Due programmi di caricamento incorporati possono enormemente semplificare l'integrazione con i back-end del database. I programmi di caricamento JPA utilizzano le funzioni di ORM (Object-Relational Mapping) di entrambe le implementazioni OpenJPA e Hibernate della specifica JPA (Java Persistence API).

Consultare per ulteriori informazioni relative ai programmi di caricamento JPA in *Panoramica sul prodotto*.

Utilizzo di un programma di caricamento

Per aggiungere un programma di caricamento nella configurazione BackingMap, è possibile utilizzare una configurazione programmatica o una configurazione XML. Un programma di caricamento ha la seguente relazione con una mappa di backup.

- Una mappa di backup può avere solo un programma di caricamento.
- Una mappa di backup del client (cache locale) non può avere un programma di caricamento.
- Una definizione di programma di caricamento può essere valida per più mappe di backup, ma ciascuna mappa di backup ha una sua propria istanza di programma di caricamento.

Per ulteriori informazioni, consultare la sezione relativa alla scrittura di un programma di caricamento in *Panoramica sul prodotto*.

L'approccio configurazione XML per un plug in un programma di caricamento

Un programma di caricamento fornito da un'applicazione può essere collegato utilizzando un file XML. Il seguente esempio mostra come collegare il programma di caricamento "MyLoader" nella mappa di backup "map1" . È possibile specificare il className per il proprio programma di caricamento, il nome del database e i dettagli di connessione e le proprietà del livello di isolamento. È possibile utilizzare la stessa struttura XML solo se si sta utilizzando un programma di precaricamento specificando il classname del programma di precaricamento invece di un classname del programma di caricamento completo.

Configurazione del programma di caricamento utilizzando XML

```
<?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="grid">
        <
backingMap name="map1" pluginCollectionRef="map1" lockStrategy="OPTIMISTIC" />
    <
/objectGrid>
<
/objectGrids>
<backingMapPluginCollections>
    <backingMapPluginCollection id="map1">
        <bean id="Loader" className="com.myapplication.MyLoader">
            <property name="dataBaseName"
                      type="java.lang.String"
                      value="testdb"
                      description="database name" />
            <property name="isolationLevel"
                      type="java.lang.String"
                      value="read committed"
                      description="iso level" />
        </bean>
    </backingMapPluginCollection>
</backingMapPluginCollections>
</objectGridConfig>
```

Plug-in di un programma di caricamento in modo programmatico

È possibile solo utilizzare una configurazione programmatica con griglie locali in-memoria. Il seguente frammento di codice mostra come collegare un programma di caricamento fornito da un'applicazione in una mappa di backup map1 utilizzando l'API ObjectGrid:

Configurazione programmatica di un programma di caricamento

```
import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
import com.ibm.websphere.objectgrid.ObjectGridManager;
import com.ibm.websphere.objectgrid.ObjectGrid;
import com.ibm.websphere.objectgrid.BackingMap;
ObjectGridManager ogManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid og = ogManager.createObjectGrid( "grid" );
BackingMap bm = og.defineMap( "map1" );
MyLoader loader = new MyLoader();
loader.setDataBaseName("testdb");
loader.setIsolationLevel("read committed");
bm.setLoader( loader );
```

Questo frammento assume che la classe MyLoader sia la classe fornita dall'applicazione che implementa l'interfaccia com.ibm.websphere.objectgrid.plugins.Loader. Poiché l'associazione di un programma di caricamento con una mappa di backup non può essere modificata dopo che ObjectGrid sia stato inizializzato, il codice deve essere eseguito prima di richiamare il metodo initialize dell'interfaccia ObjectGrid che si sta richiamando. Si verifica un'eccezione IllegalStateException nel richiamo del metodo setLoader se esso viene richiamato dopo che sia stata realizzata l'inizializzazione.

Il programma di caricamento fornito dall'applicazione può avere delle proprietà impostate. Nell'esempio il programma di caricamento MyLoader viene utilizzato per leggere e scrivere dati da una tabella in un database relazionale. Il programma di caricamento deve specificare il nome del database e il livello di isolamento SQL. Il programma di caricamento MyLoader ha i metodi setDataBaseName e setIsolationLevel che consentono all'applicazione di impostare queste due proprietà del programma di caricamento.

- L'utente 'bob' è autenticato come un utente eXtreme Scale. L'applicazione ha accesso ad una griglia denominata 'mygrid' utilizzando il nome dell'unità di persistenza di 'DB2Hibernate'. Il server contenitore è 'XS_Server1'. L'informazione risultante dovrebbe essere la seguente:
  - **Utente**=bob
  - **Nome stazione di lavoro** =XS_Server1,192.168.1.101
  - **Nome applicazione**=mygrid,DB2Hibernate
  - **Informazioni di account**=1, DEFAULT,FE7954BD-0126-4000-E000-2298094151DB,com.ibm.db2.jcc.t4.b@71787178
- L'utente 'bob' viene autenticato utilizzando un token WAS. L'applicazione ha accesso ad una griglia denominata 'mygrid" utilizzando il nome dell'unità di persistenza 'DB2OpenJPA'. Il server contenitore è 'XS_Server2'. L'informazione risultante dovrebbe essere come segue:
  - **utenter**

    =acme.principal.UserPrincipal[Bob],acme.principal.GroupPrincipal[admin]
  - **Nome stazione di lavoro**=XS_Server2,192.168.1.102
  - **Nome applicazione**=mygrid,DB2OpenJPA
  - **Informazioni di account**=188,DEFAULT,FE72BC63-0126-4000-E000-851C092A4E33,com.ibm.ws.rsadapter.jdbc.WSJccSQLJConnection@2b432b43

Consultare DB2 Performance Expert informazioni sul monitoraggio dell'accesso al database.

Warm-up e precaricamento dei dati

In svariati scenari che prevedono l'utilizzo di un programma di caricamento, è possibile preparare la propria griglia attraverso il precaricamento di dati.

Quando si utilizza la griglia come una cache completa, essa deve contenere tutti i dati e deve essere caricata prima che i client ci si possano collegare. Quando si utilizza una cache limitata, è necessario effettuarne il warm-up con i dati, in modo da consentire ai client un accesso immediato ad essi al momento del collegamento.

Esistono due approcci per il precaricamento dei dati nella griglia: utilizzando il plug-in Loader oppure un programma di caricamento del client, come descritto nelle seguenti sezioni.

Plug-in Loader

Il plug-in Loader viene associato a ciascuna mappa ed è responsabile della sincronizzazione di un singolo frammento della partizione primaria con il database. Il metodo preloadMap del plug-in Loader viene richiamato automaticamente quando si attiva un frammento. Quindi, se esistono 100 partizioni, esisteranno 100 istanze del programma di caricamento, ciascuna che carica i dati per la propria partizione. Quando sono in esecuzione contemporaneamente, tutti i client saranno bloccati fino al termine del precaricamento.

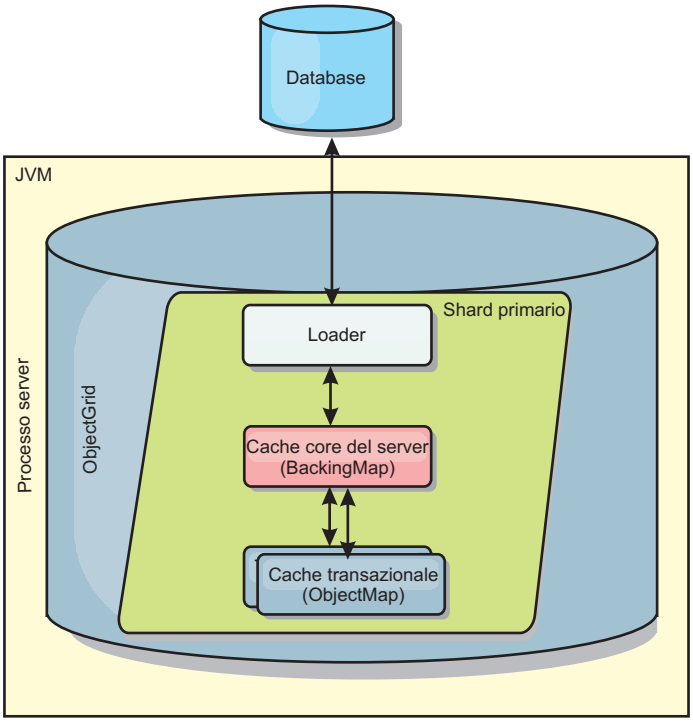


Figura 26. Plug-in Loader

Consultare i dettagli sull'utilizzo di un programma di caricamento in *Guida alla programmazione* per ulteriori informazioni.

Programma di caricamento del client

Un programma di caricamento del client è un pattern per l'utilizzo di uno o più client per il caricamento della griglia con i dati. L'utilizzo di più client, per il caricamento dei dati della griglia, può essere efficace quando lo schema della partizione non è memorizzato nel database. I programmi di caricamento del client possono essere richiamati in modo manuale o automatico all'avvio della griglia. I programmi di caricamento del client possono facoltativamente utilizzare lo StateManager per impostare lo stato della griglia sulla modalità di precaricamento, in modo che i client non possano accedere la griglia durante il precaricamento dei dati. WebSphere eXtreme Scale include un programma di caricamento basato su JPA (Java Persistence API) che può essere utilizzato per caricare la griglia automaticamente con provider OpenJPA o Hibernate JPA.

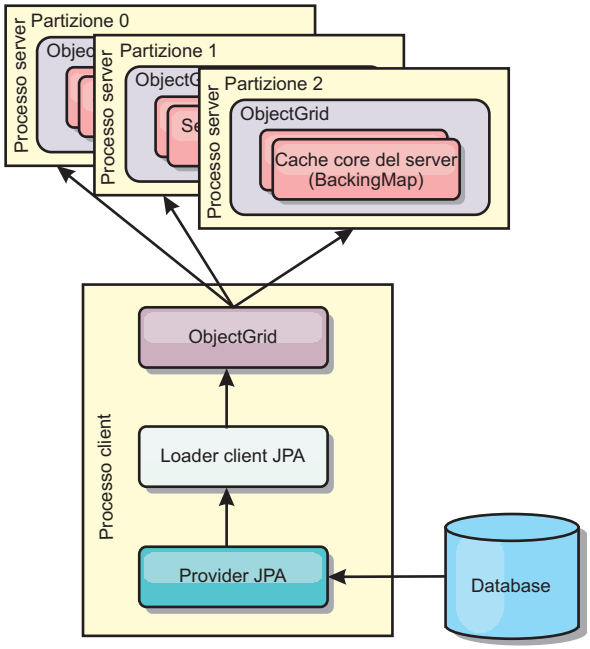


Figura 27. Programma di caricamento del client

Precaricamento della mappa

Le mappe possono essere associate ai programmi di caricamento. Un programma di caricamento viene utilizzato per eseguire il fetch di oggetti quando questi non vengono trovati nella mappa (una mancata corrispondenza nella cache) ed anche per scrivere le modifiche su un backend quando una transazione esegue il commit. I programmi di caricamento possono anche essere utilizzati per precaricare i dati in una mappa. Il metodo preloadMap dell'interfaccia Loader viene chiamato su ciascuna mappa quando la sua partizione corrispondente nella MapSet diventa primaria. Il metodo preloadMap non viene chiamato sulle repliche. Esso prova a caricare dal backend tutti i dati di riferimento previsti nella mappa utilizzando la sessione fornita. La mappa in questione viene identificata dall'argomento BackingMap passato al metodo preloadMap.

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

Precaricamento nella MapSet partizionata

Le mappe possono essere suddivise in N partizioni. Le mappe possono essere distribuite su più server, con ciascuna voce identificata da una chiave memorizzata

solo su uno di questi server. È possibile gestire mappe molto grandi in eXtreme Scale poiché l'applicazione non è più limitata dalla dimensione heap di una singola JVM per contenere tutte le voci di una mappa. Le applicazioni che eseguono il precaricamento con il metodo preloadMap dell'interfaccia Loader devono identificare il sottoinsieme dei dati di cui esso effettua il precaricamento. Esiste sempre un numero fisso di partizioni. È possibile determinare questo numero utilizzando il seguente codice di esempio:

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

Questo codice di esempio mostra come un'applicazione può identificare il sottoinsieme dei dati da precaricare dal database. Le applicazioni devono sempre utilizzare questo metodo, anche quando la mappa non è inizialmente partizionata. Questo metodo consente flessibilità: se la mappa venisse in seguito partizionata dall'amministratore, il programma di caricamento continuerebbe a lavorare in modo corretto.

L'applicazione dovrà inviare delle query per poter recuperare la serie secondaria *myPartition* dal backend. Se si utilizza un database, potrebbe risultare più semplice avere una colonna con l'identificatore della partizione per un determinato record, a meno che non vi sia una query naturale che consenta ai dati nella tabella di essere partizionati in modo semplice.

Consultare i dettagli sulla scrittura di un programma di caricamento con un controller del precaricamento in *Guida alla programmazione* per un esempio su come implementare un programma di caricamento su un eXtreme Scale replicato.

Prestazioni

L'implementazione del precaricamento copia i dati dal backend nella mappa, ordinando più oggetti nella mappa in una singola transazione. Il numero ottimale di record da memorizzare per transazione dipende da diversi fattori, comprese la complessità e la dimensione. Ad esempio, dopo che la transazione ha incluso blocchi di più di 100 voci, il beneficio sulle prestazioni diminuisce all'aumentare del numero di voci. Per determinare il numero ottimale, cominciare con 100 voci ed aumentare il numero finché non si annulla il beneficio sulle prestazioni. Transazioni più grandi risultano in prestazioni di replica migliori. Ricordarsi che solo i frammenti primari eseguono il codice precaricato. I dati precaricati sono replicati dal frammento primario su qualsiasi replica in linea.

Precaricamento delle MapSet

Se l'applicazione utilizza una MapSet con più mappe ognuna di esse avrà il proprio programma di caricamento. Ciascun programma di caricamento ha un metodo di precaricamento. Ciascuna mappa viene caricata in modo seriale da eXtreme Scale. Potrebbe risultare più efficiente precaricare tutte le mappe designando una sola mappa come mappa di precaricamento. Questo processo rappresenta una convenzione di applicazione. Ad esempio, due mappe, department ed employee, potrebbero utilizzare il programma di caricamento di department per precaricare le mappe relative ai reparti ed agli impiegati. Questa procedura assicura che, dal punto di vista della transazione, se un'applicazione richiede un reparto di conseguenza anche tutti gli impiegati di tale reparto saranno nella cache. Quando il programma di caricamento di department precarica un reparto dal backend, esso esegue anche il fetch degli impiegati di tale reparto. L'oggetto department ed i suoi oggetti employee associati saranno quindi aggiunti alla mappa con un'unica transazione.

Precaricamento recuperabile

Alcuni clienti hanno insiemi di dati molto grandi che devono essere memorizzati nella cache. Il precaricamento di tali dati potrebbe richiedere molto tempo. In alcuni casi, prima che un'applicazione possa andare in linea si deve attendere il precaricamento dei dati. È possibile però beneficiare del precaricamento recuperabile. Si supponga di dover precaricare un milione di record. L'elemento primario inizia il precaricamento, ma va in errore dopo 800.000 record. Solitamente la replica scelta per diventare l'elemento primario, ripulisce i record dallo stato 'replicato' e riparte dall'inizio. eXtreme Scale può utilizzare un'interfaccia ReplicaPreloadController. Anche il programma di caricamento dell'applicazione dovrà implementare l'interfaccia. Questo esempio aggiunge un singolo metodo al programma di caricamento: `Status checkPreloadStatus(Session session, BackingMap bmap);`. Questo metodo viene chiamato dal runtime di eXtreme Scale prima che venga normalmente chiamato il metodo di precaricamento dell'interfaccia del programma di caricamento. Il programma eXtreme Scale verifica il risultato di questo metodo (Stato) per determinare il suo comportamento nel caso in cui una replica venisse promossa a primario.

Tabella 4. Valore di stato e risposta

Valori di stato restituiti	Risposta di eXtreme Scale
Status.PRELOADED_ALREADY	eXtreme Scale non chiama il metodo di precaricamento poiché questo valore di stato indica che la mappa è stata completamente caricata.
Status.FULL_PRELOAD_NEEDED	eXtreme Scale ripulisce la mappa e chiama il metodo di precaricamento normalmente.
Status.PARTIAL_PRELOAD_NEEDED	eXtreme Scale lascia la mappa com'è e chiama il precaricamento. Questa strategia consente al programma di caricamento dell'applicazione di proseguire il precaricamento da quel punto in avanti.

Chiaramente, mentre un elemento primario esegue il precaricamento di una mappa, dovrà lasciare uno stato in una mappa della MapSet che è in fase di replica, in modo da consentire alla replica di determinare quale stato restituire. È possibile utilizzare un'ulteriore mappa denominata ad esempio, RecoveryMap. Questa RecoveryMap dovrà far parte della stessa MapSet che è in fase di precaricamento, per poter essere sicuri che la mappa venga replicata in modo congruente con i dati che si stanno precaricando. Di seguito viene riportato un suggerimento di implementazione.

Appena il precaricamento effettua il commit di ciascun blocco di record, il processo aggiorna anche il contatore o il valore nella RecoveryMap come parte della transazione. I dati precaricati ed i dati della RecoveryMap vengono replicati automaticamente nelle repliche. Quando la replica viene promossa a primario potrà verificare RecoveryMap per vedere cosa è successo.

La RecoveryMap può contenere una singola voce con la chiave di stato. Se non esiste alcun oggetto per questa chiave sarà necessario un precaricamento completo (checkPreloadStatus returns FULL_PRELOAD_NEEDED). Se esiste un oggetto per questa chiave di stato ed il suo valore è COMPLETE, il precaricamento si completa, ed il metodo checkPreloadStatus restituisce PRELOADED_ALREADY. Altrimenti l'oggetto valore indica da dove deve essere riavviato il precaricamento ed il metodo checkPreloadStatus restituisce PARTIAL_PRELOAD_NEEDED. Il programma di caricamento può memorizzare il punto di recupero in una variabile di istanza del programma di caricamento in modo che, quando verrà chiamato il precaricamento, il programma di caricamento conoscerà il punto di inizio. La RecoveryMap potrà inoltre contenere una voce per ciascuna mappa se queste vengono caricate separatamente.

Gestione del recupero in modalità di replica sincrona con un programma di caricamento

Il runtime di eXtreme Scale è progettato per non perdere i dati su cui è stato eseguito il commit quando l'elemento primario va in errore. La seguente sessione mostra gli algoritmi utilizzati. Questi algoritmi si applicano solo quando un gruppo di replica utilizza la replica sincrona. L'utilizzo di un programma di caricamento è facoltativo.

Il runtime di eXtreme Scale può essere configurato per replicare tutte le modifiche da un elemento primario ad una replica in modo sincrono. Quando si posiziona una replica sincrona, essa riceve una copia dei dati esistenti sul frammento primario. In questo frattempo, il frammento primario continua a ricevere transazioni e le copia sulla replica in modo asincrono. La replica a questo punto non è considerata in linea.

Dopo che la replica ha raggiunto il frammento primario entra in modalità peer ed inizia la replica sincrona. Ogni transazione di cui si effettua il commit sul frammento primario viene inviata alle repliche sincrone ed il frammento primario attende una risposta da ciascuna replica. Una sequenza di commit sincrona con il programma di caricamento sul frammento primario, assomiglia alla seguente serie di fasi:

Tabella 5. Sequenza commit sul frammento primario

Fase con il programma di caricamento	**Fase senza programma di caricamento**
Ricezione dei blocchi per le voci	uguale
Esecuzione del flush delle modifiche sul programma di caricamento	no-op
Salvataggio delle modifiche nella cache	uguale
Invio delle modifiche alle repliche ed attesa della conferma di ricezione	uguale
Esecuzione del commit sul programma di caricamento con il plug-in TransactionCallback	chiamata al commit del plug-in, ma nulla viene fatto
Rilascio dei blocchi sulle voci	uguale

Si noti come le modifiche vengono inviate alla replica prima che su di esse si esegua il commit sul programma di caricamento. Per determinare quando viene effettuato il commit delle modifiche sulla replica, consultare questa sequenza: al momento dell'inizializzazione, inizializzare gli elenchi tx sul frammento primario come riportato di seguito.

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

Durante l'elaborazione del commit singolo, utilizzare la seguente sequenza:

Tabella 6. Elaborazione del commit sincrono

Fase con il programma di caricamento	**Fase senza programma di caricamento**
Ricezione dei blocchi per le voci	uguale
Esecuzione del flush delle modifiche sul programma di caricamento	no-op
Salvataggio delle modifiche nella cache	uguale

Tabella 6. Elaborazione del commit sincrono (Continua)

Fase con il programma di caricamento	**Fase senza programma di caricamento**
Invio delle modifiche con una transazione su cui è stato effettuato il commit. Rollback della transazione sulla replica ed attesa della conferma di ricezione	uguale
Cancellazione dell'elenco di transazioni di cui si è effettuato il commit. Rollback delle transazioni	uguale
Esecuzione del commit del programma di caricamento con il plug-in TransactionCallBack	chiamata del commit del plug-in TransactionCallBack, ma solitamente nulla viene effettuato
Se il commit riesce, aggiunta della transazione alle transazioni di cui si è effettuato il commit, altrimenti aggiunta alle transazioni di cui si è effettuato il rollback	no-op
Rilascio dei blocchi sulle voci	uguale

Per l'elaborazione della replica, utilizzare la seguente sequenza:

1. Ricezione delle modifiche
2. Esecuzione del commit di tutte le transazioni ricevute nell'elenco delle transazioni di cui si è effettuato il commit
3. Esecuzione del rollback di tutte le transazioni ricevute nell'elenco delle transazioni di cui si è effettuato il rollback
4. Avvio di una transazione o di una sessione
5. Applicazione delle modifiche alla transazione o alla sessione
6. Salvataggio della transazione o della sessione nell'elenco 'in sospeso'
7. Invio della risposta

Si noti che, sulla replica, non si verificano interazioni del programma di caricamento mentre essa è in modalità di replica. Il frammento primario deve trasmettere tutte le modifiche attraverso il programma di caricamento. La replica non effettuerà alcuna modifica. Un effetto collaterale di questo algoritmo è che le repliche avranno le transazioni, ma su di esse non verrà eseguito alcun commit finché la successiva transazione del frammento primario non invierà lo stato del commit di tali transazioni. Sulla replica verrà quindi eseguito il commit ed il rollback delle transazioni. Fino ad allora sulle transazioni non verrà eseguito il commit. È possibile aggiungere un timer sul frammento primario che invii il risultato della transazione dopo un breve periodo di tempo (pochi secondi). Questo timer limita, ma non elimina, la pesantezza di questa finestra temporale. Questa pesantezza rappresenta un problema solo quando si utilizza la replica in modalità di lettura. Altrimenti, essa non ha alcun impatto sull'applicazione.

Quando il frammento primario va in errore, è possibile che su di esso si sia eseguito il commit o il rollback di alcune transazioni, ma che il messaggio non abbia raggiunto la replica con i seguenti risultati. Quando una replica viene promossa a frammento primario, una delle prime azioni è quella di gestire questa condizione. Ciascuna transazione pendente viene nuovamente elaborata sulla serie di mappe del frammento primario. Se c'è un programma di caricamento, ciascuna transazione viene consegnata ad esso. Queste transazioni vengono applicate in rigido ordine FIFO (first in first out). Se una transazione va in errore, viene ignorata. Se tre transazioni sono in sospeso, A, B e C, sulla A si esegue il commit,

sulla B il rollback e sulla C di nuovo il commit. Nessuna transazione ha alcun impatto sulle altre. Sono da considerare indipendenti.

Un programma di caricamento potrebbe utilizzare una logica leggermente differente quando si trova nella modalità di recupero da failover rispetto alla modalità normale. Il programma di caricamento può venire facilmente a conoscenza di essere in modalità di recupero da failover implementando l'interfaccia ReplicaPreloadController. Il metodo checkPreloadStatus viene chiamato unicamente quando il recupero da failover viene completato. Quindi, se il metodo apply dell'interfaccia del programma di caricamento viene chiamato prima del metodo checkPreloadStatus, significa che si tratta di una transazione di recupero. Il recupero da failover viene completato dopo che è stato chiamato il metodo checkPreloadStatus.

Tecniche di sincronizzazione del database

Quando WebSphere eXtreme Scale viene utilizzato come una cache, le applicazioni devono essere scritte in modo da tollerare dati obsoleti se il database può essere aggiornato in modo indipendente da una transazione eXtreme Scale. Per servire da spazio di elaborazione di database in memoria sincronizzato, eXtreme Scale fornisce alcuni modi per mantenere aggiornata la cache.

Tecniche di sincronizzazione del database

Aggiornamento periodico

La cache può essere invalidata automaticamente o aggiornata periodicamente utilizzando il programma di aggiornamento database basato sul tempo JPA (Java Persistence API).Il programma di aggiornamento esegue periodicamente nel database, utilizzando un provider JPA, una query degli aggiornamenti o inserimenti verificatisi dal precedente aggiornamento. Tutte le modifiche identificate vengono automaticamente invalidate o aggiornate se viene utilizzato con una cache limitata. Se viene utilizzato con una cache completa, le voci possono essere rilevate ed inserite nella cache. Le voci non vengono mai rimosse dalla cache.

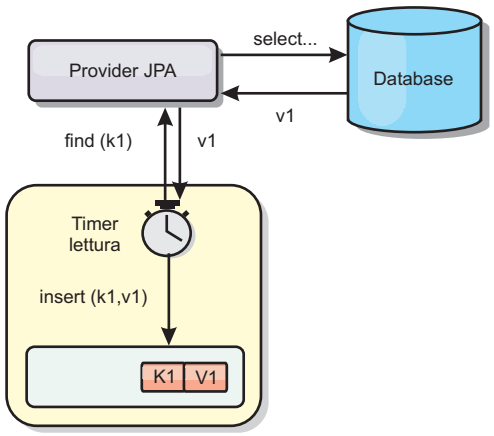


Figura 28. Aggiornamento periodico

Eliminazione

Le cache limitate possono utilizzare le politiche di eliminazione per rimuovere automaticamente i dati dalla cache senza interessare il database. Sono incluse tre

politiche integrate in eXtreme Scale: time-to-live, least-recently-used e least-frequently-used. Tutte e tre le politiche possono facoltativamente eliminare i dati in modo più aggressivo quando la memoria diviene vincolata, abilitando l'opzione di eliminazione basata sulla memoria.

Invalidazione basata sugli eventi

È possibile invalidare o aggiornare cache complete e limitate utilizzando un generatore eventi come JMS (Java Message Service). L'invalidazione mediante JMS può essere collegata manualmente a qualunque processo che aggiorna il back-end mediante un trigger del database. In eXtreme Scale viene fornito un plug-in ObjectGridEventListener JMS che può inviare una notifica ai client quando la cache del server viene modificata. Questo può ridurre la quantità di tempo in cui il client può visualizzare dati obsoleti.

Invalidazione programmatica

Le API di eXtreme Scale consentono l'interazione manuale della cache locale e quella del server mediante i metodi API Session.beginNoWriteThrough(), ObjectMap.invalidate() e EntityManager.invalidate(). Se un processo server o client non necessita più di una parte dei dati, è possibile utilizzare i metodi di invalidazione per rimuovere i dati dalla cache locale o da quella del server. Il metodo beginNoWriteThrough viene applicato a qualunque operazione ObjectMap o EntityManager alla cache locale senza chiamare il programma di caricamento. Se viene richiamato da un client, l'operazione viene applicata solo alla cache locale (il programma di caricamento remoto non viene richiamato). Se viene richiamato sul server, l'operazione viene applicata solo alla cache principale del server senza richiamare il programma di caricamento.

Invalidazione dei dati della cache obsoleti

Per ridurre il tempo in cui i client possono visualizzare dati obsoleti, è possibile utilizzare un meccanismo di invalidazione programmatica oppure basata sugli eventi.

Invalidazione basata sugli eventi

È possibile invalidare cache complete e limitate utilizzando un generatore eventi come JMS (Java Message Service). L'invalidazione mediante JMS può essere collegata manualmente a qualunque processo che aggiorna il back-end mediante un trigger del database. In eXtreme Scale viene fornito un plug-in JMS ObjectGridEventListener che può inviare una notifica ai client quando la cache del server viene modificata. Questo tipo di notifica riduce il tempo in cui il client può visualizzare dati obsoleti.

L'invalidazione basata sugli eventi è costituita, normalmente, dai seguenti tre componenti.

- **Coda eventi:** una coda eventi memorizza gli eventi di modifica dei dati. Essa può essere una coda JMS, un database, una coda FIFO in memoria o qualunque tipo di manifest fino a quando può gestire gli eventi di modifica dei dati.
- **Publisher eventi:** un publisher eventi pubblica gli eventi di modifica dei dati nella coda eventi. Un publisher eventi, solitamente, è un'applicazione creata dall'utente oppure un'implementazione di plug-in di eXtreme Scale. Il publisher eventi è a conoscenza del momento in cui i dati vengono modificati o esso stesso

li modifica. Quando viene eseguito il commit di una transazione, vengono generati degli eventi per i dati modificati e il publisher eventi li pubblica nella coda eventi.

- **Consumer eventi:** un consumer eventi utilizza gli eventi di modifica dei dati. Il consumer eventi è, solitamente, un'applicazione che assicura che i dati della griglia di destinazione vengano aggiornati con le modifiche più recenti delle altre griglie. Questo consumer eventi interagisce con la coda eventi per ottenere le più recenti modifiche dei dati e applica tali modifiche alla griglia di destinazione. I consumer eventi possono utilizzare le API di eXtreme Scale per invalidare dati obsoleti o aggiornare la griglia con i dati più recenti.

Ad esempio, JMSObjectGridEventListener dispone di un'opzione per un modello client-server, in cui la coda eventi è una destinazione JMS designata. Tutti i processi server sono publisher eventi. Quando viene eseguito il commit di una transazione, il server ottiene le modifiche apportate ai dati e le pubblica nella destinazione JMS designata. Tutti i processi client sono consumer eventi. Essi ricevono le modifiche ai dati dalla destinazione JMS designata e le applicano alla cache locale del client.

Per ulteriori informazioni, consultare l'argomento relativo al meccanismo di invalidazione client contenuto in *Guida alla gestione*.

Invalidazione programmatica

Le API di WebSphere eXtreme Scale consentono l'interazione manuale della cache locale e quella del server mediante i metodi API Session.beginNoWriteThrough(), ObjectMap.invalidate() e EntityManager.invalidate(). Se un processo server o client non necessita più di una parte dei dati, è possibile utilizzare i metodi di invalidazione per rimuovere tali dati dalla cache locale o da quella del server. Il metodo beginNoWriteThrough applica qualunque operazione ObjectMap o EntityManager alla cache locale senza chiamare il programma di caricamento. Se viene richiamato da un client, l'operazione viene applicata solo alla cache locale (il programma di caricamento remoto non viene richiamato). Se viene richiamato sul server, l'operazione viene applicata solo alla cache principale del server senza richiamare il programma di caricamento.

È possibile utilizzare l'invalidazione programmatica con altre tecniche per determinare quando invalidare i dati. Ad esempio, questo metodo di invalidazione utilizza i meccanismi di invalidazione basati sugli eventi per ricevere gli eventi di modifica dei dati e poi utilizza le API per invalidare i dati obsoleti.

Indicizzazione

Utilizzare MapIndexPlugin per effettuare il build di un indice o di più indici su una BackingMap per supportare l'accesso dati non chiave.

Tipi di indice e configurazione

La funzione di indicizzazione è rappresentata da MapIndexPlugin o, in breve, da Index. Index è un plug-in della BackingMap. Un BackingMap può avere più plug-in Index configurati, finché ciascuno di essi segue le regole di configurazione di Index.

È possibile utilizzare la funzione di indicizzazione per effettuare il build di un indice o di più indici su una BackingMap. Un indice creato da un attributo o da un elenco di attributi di un oggetto nella BackingMap. Questa funzione consente alle

applicazioni di trovare determinati oggetti più velocemente. Con la funzione di indicizzazione, le applicazioni possono trovare oggetti con un valore specifico o contenuti in un intervallo di valori degli attributi indicizzati.

Sono possibili due tipi di indicizzazione: statica e dinamica. Con l'indicizzazione statica, è necessario configurare il plug-in Index sulla BackingMap prima di inizializzare l'istanza ObjectGrid. È possibile fare ciò con la configurazione programmatica o XML di una BackingMap. L'indicizzazione statica avvia il build durante l'inizializzazione di ObjectGrid. L'indice è sempre sincronizzato con la BackingMap e pronto per essere utilizzato. Dopo l'avvio del processo di indicizzazione statico, la gestione dell'indice diventa parte del processo di gestione della transazione eXtreme Scale. Quando la transazione effettua il commit delle modifiche, esse aggiornano anche l'indice statico e se si effettua il rollback della transazione viene eseguito anche quello dell'indice.

Con l'indicizzazione dinamica, è possibile creare un indice della BackingMap prima o dopo l'inizializzazione dell'istanza ObjectGrid che la contiene. Le applicazioni controllano il ciclo di vita del processo di indicizzazione dinamico in modo che sia possibile rimuovere un'indice quando non è più necessario. Quando l'applicazione crea un indice dinamico, esso potrebbe non essere pronto per l'utilizzo immediato a causa del tempo richiesto per il completamento del suo processo di build. Poiché il tempo richiesto dipende dalla quantità di dati da indicizzare, esiste l'interfaccia DynamicIndexCallback per le applicazioni che desiderano ricevere notifiche del verificarsi di determinati eventi di indicizzazione. Tra questi sono compresi ready, error e destroy. Le applicazioni possono implementare quest'interfaccia di callback e registrarsi con il processo di indicizzazione dinamico.

Se una BackingMap ha configurato un plug-in indice, si potrà ottenere l'oggetto proxy dell'indice dell'applicazione dalla ObjectMap corrispondente. La chiamata del metodo getIndex sulla ObjectMap ed il passaggio del nome del plug-in Index, restituisce l'oggetto proxy dell'indice. È necessario effettuare il cast dell'oggetto proxy dell'indice in un'interfaccia dell'indice dell'applicazione appropriata, quale MapIndex, MapRangeIndex, o un'interfaccia dell'indice personalizzata. Dopo aver ottenuto l'oggetto proxy dell'indice, si potranno utilizzare i metodi definiti nell'interfaccia dell'indice dell'applicazione per trovare gli oggetti memorizzati nella cache.

Le fasi per l'utilizzo dell'indicizzazione sono riepilogate nel seguente elenco:

- Aggiungere un plug-in indice statico o dinamico nella BackingMap.
- Ottenere un oggetto proxy dell'indice dall'applicazione con il metodo getIndex della ObjectMap.
- Eseguire il cast dell'oggetto proxy dell'indice in un'interfaccia dell'indice dell'applicazione appropriata, ad esempio MapIndex, MapRangeIndex o un'interfaccia dell'indice personalizzata.
- Utilizzare metodi che siano definiti nell'interfaccia dell'indice dell'applicazione per trovare gli oggetti memorizzati nella cache.

La classe HashIndex è l'implementazione del plug-in dell'indice incorporato in grado di supportare entrambe le interfacce dell'indice dell'applicazione incorporate: MapIndex e MapRangeIndex. È anche possibile creare i propri indici. É possibile aggiungere HashIndex nella BackingMap sia come indice statico che dinamico, ottenere l'oggetto proxy dell'indice MapIndex o MapRangeIndex per trovare gli oggetti memorizzati nella cache.

Per informazioni sulla configurazione di HashIndex, fare riferimento a Configurazione di HashIndex.

Per ulteriori informazioni sulla scrittura dei propri plug-in di indice, consultare le informazioni relative alla scrittura di un plug-in di indice in *Guida alla programmazione*..

Per informazioni su come utilizzare l'indicizzazione, consultare le informazioni relative all'utilizzo dell'indicizzazione per l'accesso ai dati non chiave in *Guida alla programmazione* e HashIndex composto.

Considerazioni sulla qualità dei dati

I risultati dei metodi di query dell'indice rappresentano solo un'istantanea dei dati in un determinato momento. Le voci dei dati non vengono bloccate dopo che i risultati sono stati restituiti all'applicazione. L'applicazione deve essere consapevole che gli aggiornamenti ai dati potrebbero essere eseguiti su una serie di dati restituita. Ad esempio, l'applicazione ottiene una chiave di un oggetto memorizzato nella cache eseguendo il metodo findAll di MapIndex. Questo oggetto chiave restituito è associato ad una voce dati nella cache. L'applicazione deve essere in grado di eseguire il metodo get su ObjectMap per trovare un oggetto fornendo l'oggetto chiave. Se un'altra transazione rimuove l'oggetto dati dalla cache appena prima che il metodo get venga chiamato, i risultati restituiti saranno nulli.

Considerazioni sulle prestazioni dell'indicizzazione

Uno degli obiettivi primari della funzione di indicizzazione è quello di migliorare le prestazioni generali della BackingMap. Se l'indicizzazione non viene utilizzata in modo corretto, le prestazioni dell'applicazione potrebbero essere compromesse. Considerare i seguenti fattori prima di utilizzare questa funzione.

- **Il numero di transazioni in scrittura contemporanee:** l'elaborazione dell'indice può verificarsi ogni volta che una transazione scrive dati in una BackingMap. Le prestazioni peggiorano se più transazioni scrivono dati nella mappa contemporaneamente mentre un'applicazione tenta delle operazioni di query dell'indice.
- **La dimensione della serie di risultati restituita da un'operazione di query:** le prestazioni delle query peggiorano, al crescere delle dimensioni della serie di risultati. Le prestazioni tendono a degradare quando la dimensione della serie di risultati supera il 15% della BackingMap.
- **Il numero di indici creati sulla stessa BackingMap:** ciascun indice consuma risorse di sistema. Le prestazioni peggiorano al crescere del numero di indici creati su una BackingMap.

La funzione di indicizzazione può migliorare drasticamente le prestazioni della BackingMap. I casi ideali sono rappresentati da BackingMap dove la maggior parte delle operazioni è di lettura, la serie di risultati di query è solo una piccola percentuale rispetto alle voci della BackingMap ed solo pochi indici sono stati creati su di essa.

Concetto di memorizzazione nella cache di oggetti Java

WebSphere eXtreme Scale è principalmente usato come una griglia di dati ed una cache per gli oggetti Java. Si possono utilizzare diverse API per interagire con la griglia eXtreme Scale in modo da accedere a questi oggetti e memorizzarli.

Questi argomenti descrivono alcune delle API più comuni ed alcuni dei concetti di cui bisogna essere a conoscenza quando si sceglie un'API e la topologia di distribuzione. Consultare l'argomento “Architettura di memorizzazione nella cache: mappe, contenitori, client e cataloghi” a pagina 9 per una descrizione dei vari servizi e topologie fornite daeXtreme Scale.

Il componente principale di WebSphere eXtreme Scale è l'ObjectGrid. L'ObjectGrid è lo spazio nome che memorizza i dati correlati e che contiene serie di mappe hash, ognuna contenente una coppia chiave-valore. Queste mappe possono essere raggruppate insieme e partizionate e rese altamente disponibili e scalabili.

Poiché la griglia contiene oggetti Java per natura, ci sono alcune importanti considerazioni da fare quando si progetta un'applicazione in modo che la griglia possa memorizzare ed accedere in modo efficiente ai dati. Fattori che possono incidere su scalabilità, prestazioni e utilizzo della memoria includono quanto segue.

Programma di caricamento classe e considerazioni sul percorso di classe.

Poiché, per impostazione predefinita, eXtreme Scale memorizza oggetti Java nella cache, è necessario definire le classi nel percorso di classe ovunque venga eseguito l'accesso ai dati.

Specificamente, i processi client e contenitore di eXtreme Scale devono includere le classi o i file JAR nel percorso di classe quando vengono avviati. Quando si progetta un'applicazione per l'utilizzo con eXtreme Scale, separare la logica di business dagli oggetti di dati persistenti.

Per ulteriori informazioni consultare la sezione Caricamento classe nel Centro informazioni di WebSphere Application Server Network Deployment.

Per considerazioni nell'ambito delle impostazioni di Spring Framework, consultare la sezione contenuta nell'argomento relativo all'integrazione con Spring framework in *Guida alla programmazione*.

Per le impostazioni relative all'utilizzo dell'agent di strumentazione di WebSphere eXtreme Scale, consultare l'argomento relativo all'agent di strumentazione in *Guida alla programmazione*.

Gestione delle relazioni

I linguaggi orientati agli oggetti quali Java ed i database relazionali supportano le relazioni o associazioni. Le relazioni riducono la quantità di memoria attraverso l'utilizzo di chiavi esterne o di riferimenti ad oggetti.

Quando si utilizzano le relazioni in una griglia, i dati devono essere organizzati in una struttura ad albero vincolata. La struttura ad albero dovrà contenere un unico tipo root e tutti i child devono essere associati ad un unica root. Ad esempio: un Reparto potrà avere molti Impiegati ed un Impiegato potrà avere molti Progetti. Ma un Progetto non potrà avere molti Impiegati appartenenti a Reparti diversi. Una volta definita una root, tutti gli accessi a tale oggetto root ed ai suoi discendenti, sono gestiti attraverso la root. WebSphere eXtreme Scale utilizza il codice hash della chiave dell'oggetto root per scegliere una partizione.

Ad esempio: partition = (hashCode MOD numPartitions).

Quando tutti i dati di una relazione sono legati all'istanza di un singolo oggetto, l'intera struttura ad albero potrà essere collocata in una singola partizione e può essere acceduta in modo molto efficiente utilizzando una transazione. Se i dati sono distribuiti su più relazioni, si dovranno coinvolgere più partizioni il che richiede chiamate remote aggiuntive, condizione che potrebbe rappresentare un collo di bottiglia per le prestazioni.

Dati di riferimento

Alcune relazioni includono ricerche o dati di riferimento quali: CountryName. Ciò rappresenta un caso speciale in cui i dati devono essere presenti in tutte le partizioni. In esse, i dati potranno essere acceduti da qualsiasi chiave della root ottenendo in ritorno lo stesso risultato. I dati di riferimento come questi dovrebbero essere utilizzati solo nei casi in cui questi siano relativamente statici, poiché il loro aggiornamento può essere dispendioso, siccome sarà necessario aggiornarli su tutte le partizioni. L'API DataGrid rappresenta una tecnica comune che consente di tenere i dati di riferimento aggiornati.

Costi e benefici della normalizzazione

La normalizzazione dei dati tramite le relazioni può aiutare a ridurre la quantità di memoria utilizzata dalla griglia poiché viene diminuita la duplicazione dei dati. Tuttavia, in generale, quanti più dati relazionali vengono aggiunti tanto meno la griglia eseguirà la riduzione di scala (scale out). Quando i dati vengono raggruppati, diventa più difficile amministrare le relazioni e mantenere le dimensioni gestibili. Poiché i dati delle partizioni della griglia sono basati sulla chiave della root della struttura ad albero, la dimensione della struttura ad albero non viene presa in considerazione. Quindi, se vi sono molte relazioni per un'istanza della struttura ad albero, la griglia potrebbe essere sbilanciata, di conseguenza una partizione potrebbe contenere più dati delle altre.

Quando i dati vengono denormalizzati o resi bidimensionali, quelli che normalmente sarebbero condivisi tra due oggetti vengono duplicati e ciascuna tabella potrà essere partizionata in modo indipendente, fornendo una griglia molto più bilanciata. Se, da una parte, ciò incrementa la quantità di memoria utilizzata, dall'altra consente la scalabilità da parte dell'applicazione, poiché si potrà accedere ad una singola riga di dati che conterrà tutti i dati necessari. Questa condizione è ideale soprattutto per le griglie che vengono accedute prevalentemente in lettura, poiché la gestione dei dati diventa più dispendiosa.

Per ulteriori informazioni, consultare Classifying XTP systems and scaling.

Gestione delle relazioni utilizzando le API di accesso dati

L'API ObjectMap è la più veloce, più flessibile e più dettagliata tra tutte le API di accesso dati, poiché fornisce all'accesso dei dati nella griglia di mappe un approccio transazionale e basato sulla sessione. L'API ObjectMap consente ai client di utilizzare le comuni operazioni CRUD (create, read, update e delete) per gestire le coppie di chiave-valore nella griglia distribuita.

Quando si utilizza l'API ObjectMap, le relazioni dell'oggetto devono essere espresse incorporando la chiave esterna per tutte le relazioni nell'oggetto parent.

Segue un esempio.

```
public class Department {
 Collection<String> employeeIds;
}
```

L'API EntityManager semplifica la gestione delle relazioni estraendo i dati persistenti dagli oggetti incluse le chiavi esterne. Quando gli oggetti vengono successivamente recuperati dalla griglia, il grafico di relazioni viene creato nuovamente, come nell'esempio che segue.

```
@Entity
public class Department {
 Collection<String> employees;
}
```

L'API EntityManager è molto simile ad altre tecnologie della persistenza dell'oggetto di Java, quali JPA e Hibernate, per il fatto che esso sincronizza un grafico di istanze di un oggetto Java gestito con l'archivio persistente. In questo caso l'archivio persistente è una griglia di eXtreme Scale dove ciascuna entità viene rappresentata come una mappa che contiene i dati dell'entità piuttosto che le istanze dell'oggetto.

Considerazioni sulla chiave della cache

WebSphere eXtreme Scale utilizza mappe di hash per memorizzare dati nella griglia, in cui un oggetto Java viene utilizzato per la chiave.

Linee guida

Quando si sceglie una chiave, considerare i seguenti requisiti:

- Le chiavi non possono essere mai modificate. Se una parte della chiave deve essere modificata, la voce della cache deve essere rimossa e inserita nuovamente.
- Le chiavi devono essere di piccole dimensioni. Poiché le chiavi vengono utilizzate in ogni operazione di accesso ai dati, è preferibile mantenerne piccole le dimensioni, così che la chiave possa essere serializzata in modo efficiente ed utilizzare meno memoria.
- Implementare un valido algoritmo hash e equals. I metodi hashCode e equals(Object o) devono essere sempre sostituiti per ciascun oggetto chiave.
- Memorizzare nella cache l'hashCode della chiave. Se possibile, memorizzare l'hashCode nella cache dell'istanza dell'oggetto chiave per velocizzare i calcoli del metodo hashCode(). Poiché la chiave non è mutabile, hashCode deve essere memorizzabile nella cache.
- Evitare duplicazioni della chiave nel valore. Quando di utilizza l'API ObjectMap, è preferibile memorizzare la chiave all'interno dell'oggetto valore. Al termine di queste operazioni, i dati della chiave vengono duplicati nella memoria.

Prestazione della serializzazione

WebSphere eXtreme Scale utilizza più processi Java per contenere i dati. Questi processi serializzano i dati: convertono, cioè, i dati (che si trovano nel modulo delle istanze dell'oggetto Java) in byte e poi di nuovo in oggetti secondo quanto necessario per spostare i dati tra i processi del client e del server. L'esecuzione del marshalling dei dati è l'operazione più dispendiosa e deve essere indirizzata dallo sviluppatore dell'applicazione durante la progettazione dello schema, configurando la griglia e interagendo con le API di accesso ai dati.

Le routine predefinite di copia e serializzazione Java sono relativamente lente e possono consumare il 60 - 70 per cento del processore in una configurazione tipica. Le seguenti sezioni rappresentano delle scelte per il miglioramento della prestazione della serializzazione.

Scrittura di un ObjectTransformer per ogni BackingMap

Un ObjectTransformer può essere associato ad una BackingMap. La propria applicazione può avere una classe che implementa l'interfaccia di ObjectTransformer e che fornisce implementazioni per le operazioni di seguito riportate:

- Copia di valori
- Chiavi di serializzazione e deserializzazione verso e dai flussi
- Valori di serializzazione e deserializzazione verso e dai flussi.

L'applicazione non ha bisogno di copiare le chiavi poiché le chiavi sono considerate non mutabili.

Per ulteriori informazioni, consultare Plug-in per la serializzazione e la copia di oggetti memorizzati nella cache e Migliori prassi per l'interfaccia ObjectTransformer.

Nota: ObjectTransformer è l'unico richiamato quando ObjectGrid è informato sui dati in fase di trasformazione. Ad esempio, quando vengono utilizzati gli agent dell'API DataGrid, gli stessi agent così come i dati dell'istanza dell'agent o i dati restituiti dall'agent, devono essere ottimizzati utilizzando le tecniche personalizzate di serializzazione. ObjectTransformer non viene richiamato per gli agent delle API DataGrid.

Utilizzo delle entità

Quando si utilizzano le API EntityManager con entità, l'ObjectGrid non memorizza gli oggetti dell'entità direttamente nelle BackingMaps. L'API EntityManager converte l'oggetto dell'entità in oggetti Tupla. Per ulteriori informazioni, consultare Per ulteriori informazioni, consultare l'argomento sull'utilizzo di un programma di caricamento con mappe di entità e tuple in *Guida alla programmazione*. Le mappe dell'entità sono associate automaticamente ad un ObjectTransformer altamente ottimizzato. Ogni volta che viene utilizzata l'API ObjectMap o l'API EntityManager per interagire con le mappe dell'entità, viene richiamata l'entità ObjectTransformer.

Personalizzazione della serializzazione

Esistono alcuni casi in cui gli oggetti devono essere modificati per utilizzare la serializzazione personalizzata, come l'implementazione dell'interfaccia java.io.Externalizable o l'implementazione dei metodi writeObject e readObject per classi che implementano l'interfaccia java.io.Serializable. Le tecniche di serializzazione personalizzata devono essere impiegate quando vengono serializzati gli oggetti utilizzando meccanismi diversi dai metodi delle API ObjectGrid o EntityManager.

Ad esempio, quando oggetti o entità vengono memorizzati come dati dell'istanza in un agent dell'API DataGrid oppure quando l'agent restituisce oggetti o entità, quegli oggetti non vengono trasformati utilizzando un ObjectTransformer. L'agent, tuttavia, utilizzerà automaticamente l'ObjectTransformer quando utilizza

l'interfaccia `EntityMixin`. Per ulteriori dettagli, consultare Agent DataGrid e Mappe basate su entità.

Array di byte

Quando si utilizzano le API ObjectMap o DataGrid, gli oggetti chiave e valore vengono serializzati ogni volta che il client interagisce con la griglia e quando viene replicato l'oggetto. Per evitare il sovraccarico della serializzazione, utilizzare gli array di byte invece degli oggetti Java. Gli array di byte sono molto più semplici da inserire in memoria poiché JDK ha un numero inferiore di oggetti da cercare per la raccolta dati obsoleti e possono essere deserializzati solo all'occorrenza. Gli array di byte devono essere utilizzati solo se non è necessario accedere agli oggetti usando query o indici. Poiché i dati vengono memorizzati come byte, è possibile accedere ai dati solo attraverso chiavi.

WebSphere eXtreme Scale può memorizzare dati automaticamente come array di byte utilizzando l'opzione di configurazione della mappa CopyMode.COPY_TO_BYTES oppure può essre gestito manualmente dal client. Questa opzione memorizzerà in modo efficiente i dati in memoria e può anche deserializzare gli oggetti all'interno dell'array di byte per l'uso mediante query e indici on demand.

Per ulteriori informazioni, consultare le Migliori pratiche per il metodo CopyMode in *Guida alla programmazione*.

Inserimento di dati per fusi orari diversi

Quando si inseriscono dati con gli attributi calendar, java.util.Date e timestamp in un ObjectGrid, è necessario assicurarsi che questi attributi di data/ora vengano creati in base allo stesso fuso orario, specialmente quando vengono distribuiti in più server in fusi orari diversi. L'utilizzo di oggetti data/ora basati sullo stesso fuso orario può assicurare che l'applicazione sia affidabile da un punto di vista di fuso orario e che sia possibile eseguire query dei dati tramite i predicati calendar, java.util.Date e timestamp.

Senza la specifica esplicita di un fuso orario durante la creazione di un oggetto data/ora, Java utilizzerà il fuso orario locale e questo potrebbe causare un'incongruenza nei valori di data e ora nei client e nei server.

Prendere in considerazione un esempio in una distribuzione distribuita in cui client1 è nel fuso orario [GMT-0] e client2 è in [GMT-6] ed entrambi desiderano creare un oggetto java.util.Date con valore '1999-12-31 06:00:00'. Client1 creerà l'oggetto java.util.Date con valore '1999-12-31 06:00:00 [GMT-0]' e client2 lo crerà con valore '1999-12-31 06:00:00 [GMT-6]'. I due oggetti java.util.Date non sono uguali perché il fuso orario è diverso. Un problema simile si verifica quando vengono precaricati i dati in partizioni che risiedono su server di fusi orari diversi e viene utilizzato il fuso orario locale per creare gli oggetti data/ora.

Per evitare il problema descritto, l'applicazione può scegliere un fuso orario, ad esempio [GMT-0], come fuso orario base per la creazione di oggetti calendar, java.util.Date e timestamp.

Per ulteriori informazioni, consultare l'argomento relativo alla query dei dati in più fusi orari nella *Guida alla programmazione*.

Capitolo 3. Panoramica relativa all'integrazione della cache: JPA, sessioni e memorizzazione nella cache dinamica

L'elemento cruciale che fornisce la capability a WebSphere eXtreme Scale di essere eseguito con tale versatilità ed affidabilità, è l'applicazione dei concetti di memorizzazione in cache per ottimizzare la persistenza e conservare in memoria i dati nell'ambiente di distribuzione virtualmente.

Programmi di caricamento JPA

JPA (Java Persistence API) è una specifica che consente l'associazione di oggetti Java a database relazionali. JPA contiene una specifica ORM (Object-Relational Mapping) completa che utilizza descrittori XML, annotazioni metadati Java o entrambi per definire l'associazione tra gli oggetti Java e un database relazionale. Sono disponibili varie implementazioni open-source e commerciali.

È possibile utilizzare un'implementazione del plug-in del programma di caricamento JPA (Java Persistence API) con eXtreme Scale per interagire con qualunque database supportato dal programma di caricamento scelto. Per utilizzare JPA, è necessario disporre di un provider JPA supportato, come ad esempio OpenJPA o Hibernate, di file JAR e di un file `META-INF/persistence.xml` nel percorso di classe.

I plug-in JPALoader com.ibm.websphere.objectgrid.jpa.JPALoader e JPAEntityLoader com.ibm.websphere.objectgrid.jpa.JPAEntityLoader sono due plug-in incorporati del programma di caricamento JPA che vengono utilizzati per sincronizzare le mappe ObjectGrid con un database. Per utilizzare questa funzione, bisogna avere un'implementazione JPA come Hibernate o OpenJPA. Il database può essere qualsiasi backend supportato dal provider JPA scelto.

È possibile utilizzare il plug-in JPALoader durante la memorizzazione di dati utilizzando l'API ObjectMap. Utilizzare il plug-in JPAEntityLoader quando si stanno memorizzando dati utilizzando l'API EntityManager.

Architettura del programma di caricamento JPA

Il programma di caricamento JPA viene utilizzato per le mappe eXtreme Scale che memorizzano POJO Java (plain old Java objects).

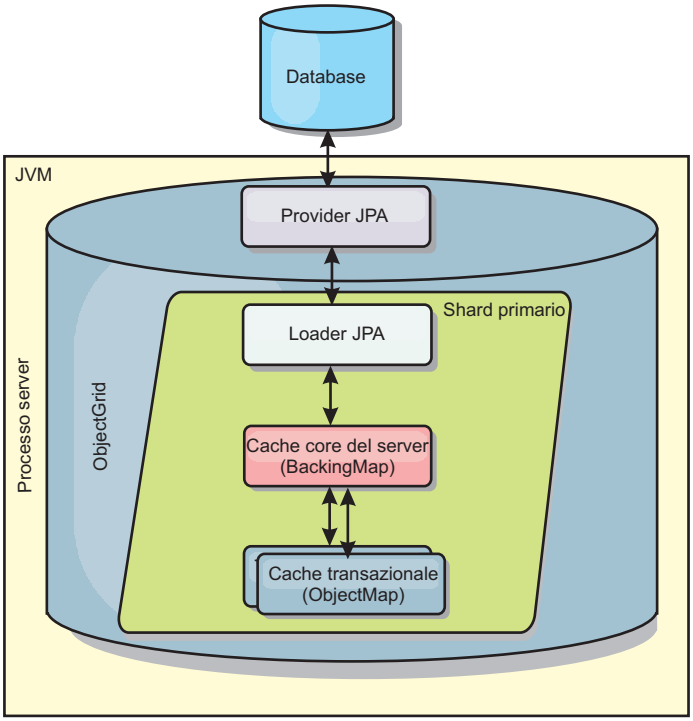


Figura 29. Architettura del programma di caricamento JPA

Quando viene chiamato un metodo ObjectMap.get(Object key), il runtime eXtreme Scale controlla prima se la voce è contenuta nel livello ObjectMap. Se non lo è, il runtime delega la richiesta al programma di caricamento JPA. Su richiesta di caricamento della chiave, JPALoader chiama il metodo EntityManager.find(Object key) JPA per cercare i dati dal livello JPA. Se i dati sono contenuti nel gestore entità JPA, vengono restituiti; altrimenti il provider JPA interagisce con il database per ottenere il valore.

Quando si verifica un aggiornamento di ObjectMap, ad esempio utilizzando il metodo ObjectMap.update(Object key, Object value), il runtime eXtreme Scale crea un LogElement per questo aggiornamento e lo invia a JPALoader. JPALoader chiama il metodo EntityManager.merge(Object value) JPA per aggiornare il valore nel database.

Per JPAEntityLoader, vengono coinvolti tutti e quattro i livelli. Tuttavia, poiché il plug-in JPAEntityLoader viene utilizzato per mappe che memorizzano le entità eXtreme Scalele relazioni tra entità potrebbero complicare gli scenari di utilizzo. Un'entità eXtreme Scale è distinta dall'entità JPA. Per ulteriori informazioni, consultare le informazioni sul plug-in JPAEntityLoader plug-in in *Guida alla programmazione*.

Metodi

I programmi di caricamento forniscono tre metodi principali:

1. get: restituisce un elenco di valori che corrisponde all'elenco di chiavi che vengono passate mediante il richiamo dei dati che utilizzano JPA. Il metodo utilizza JPA per cercare le entità nel database. Per il plug-in JPALoader, l'elenco restituito contiene un elenco di entità JPA direttamente dall'operazione di ricerca. Per il plug-in JPAEntityLoader, l'elenco restituito contiene tuple di valori dell'entità eXtreme Scale convertite dall'entità JPA.

2. batchUpdate: scrive i dati dalle mappe ObjectGrid nel database. A seconda dei diversi tipi di operazione (insert, update o delete), il programma di caricamento utilizza le operazioni JPA persist, merge e remove per aggiornare i dati nel database. Per JPALoader, gli oggetti nella mappa vengono utilizzati direttamente come entità JPA. Per JPAEntityLoader, le tuple delle entità nella mappa vengono convertite in oggetti che vengono utilizzati come entità JPA.
3. preloadMap: precarica la mappa utilizzando il metodo di caricamento del client ClientLoader.load. Per le mappe partizionate, il metodo preloadMap viene chiamato in una sola partizione. Viene specificata la partizione, la proprietà preloadPartition della classe JPALoader o JPAEntityLoader. Se il valore preloadPartition viene impostato su un valore minore di zero o maggiore di (*numero_totale_di_partizioni* - 1), viene disabilitato il precaricamento.

Entrambi i plug-in JPALoader e JPAEntityLoader funzionano con la classe JPATxCallback per coordinare le transazioni eXtreme Scale e le transazioni JPA. Per poter utilizzare questi due programmi di caricamento, JPATxCallback deve essere configurato nell'istanza ObjectGrid.

Configurazione e programmazione

Per ulteriori informazioni sulla configurazione dei programmi di caricamento JPA, consultare le informazioni sui programmi di caricamento JPA in *Guida alla gestione*. Per ulteriori informazioni sulla programmazione dei programmi di caricamento JPA, consultare *Guida alla programmazione*.

Plug-in della cache JPA

WebSphere eXtreme Scale include i plug-in della cache di livello 2 (L2) per entrambi i provider JPA (Java Persistence API) OpenJPA e Hibernate.

Utilizzando eXtreme Scale come un provider della cache L2 si aumentano le prestazioni durante la lettura dei dati e le query sui dati e si riduce il carico sul database. WebSphere eXtreme Scale ha un vantaggio sulle implementazioni della cache incorporata poiché la cache viene automaticamente replicata tra tutti i processi. Quando un client memorizza nella cache un valore, tutti gli altri client sono in grado di utilizzare il valore della cache che si trova localmente in memoria.

Con i plug-in della cache OpenJPA e Hibernate ObjectGrid, è possibile creare tre tipi di topologia: incorporata, suddivisa in partizioni incorporate e remota.

Topologia incorporata

Una topologia incorporata crea un server eXtreme Scale all'interno dello spazio di elaborazione di ogni applicazione. OpenJPA e Hibernate leggono direttamente la copia in memoria della cache e scrivono in tutte le altre copie. È possibile migliorare le prestazioni di scrittura utilizzando la replica asincrona. Questa topologia predefinita ha le migliori prestazioni quando la quantità di dati nella cache è abbastanza piccola per rientrare in un solo processo.

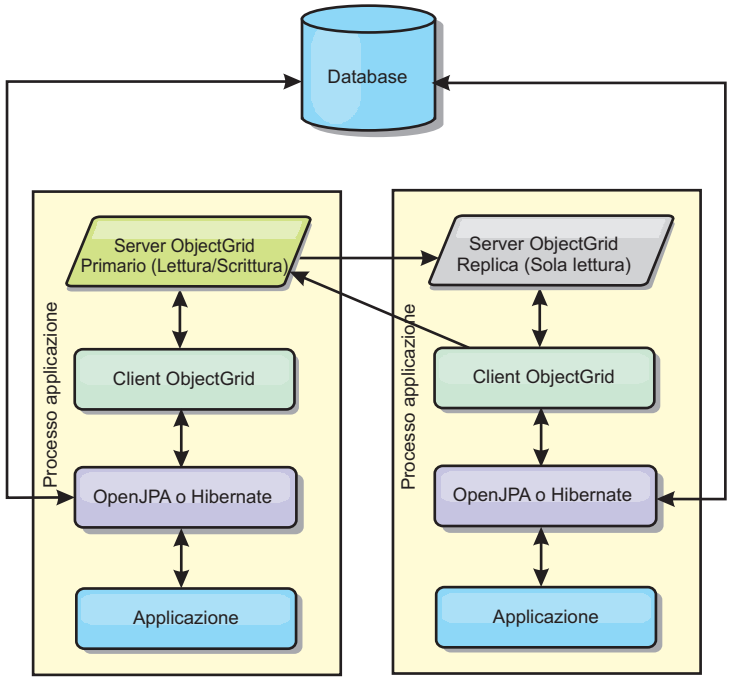


Figura 30. Topologia incorporata JPA

Vantaggi:

- Tutte le letture della cache sono accessi locali molto veloci.
- Semplice da configurare.

Limitazioni:

- La quantità di dati è limitata alla dimensione del processo.
- Tutti gli aggiornamenti della cache vengono inviati ad un processo.

Topologia incorporata suddivisa in partizioni

Quando i dati nella cache sono troppo grandi per rientrare in un singolo processo, la topologia incorporata suddivisa in partizioni utilizza le partizioni ObjectGrid per suddividere i dati tra più processi. Le prestazioni non sono tanto elevate quanto quelle della topologia incorporata poiché la maggior parte delle letture della cache sono remote. Tuttavia, è sempre possibile utilizzare questa opzione quando la latenza del database è elevata.

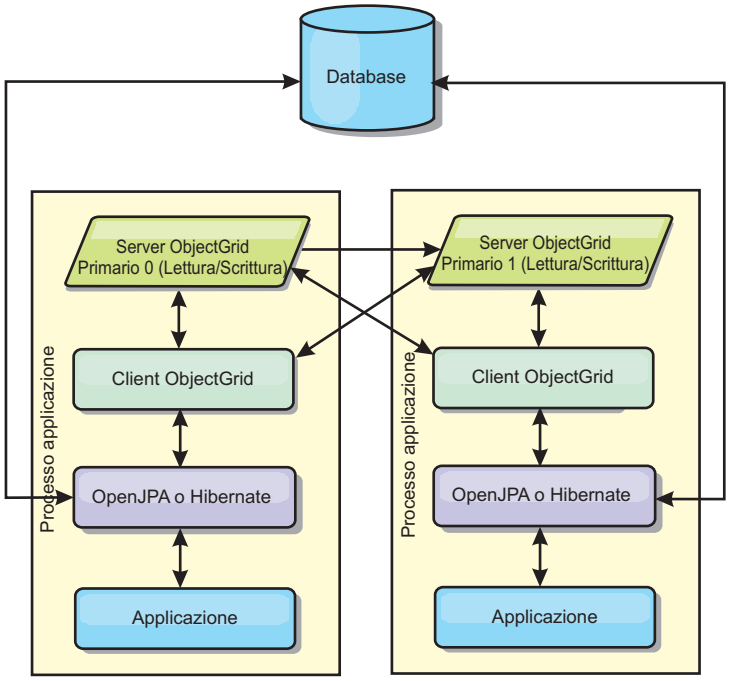


Figura 31. Topologia incorporata JPA suddivisa in partizioni

Vantaggi:

- Memorizza grandi quantità di dati.
- Semplice da configurare
- Gli aggiornamenti della cache sono distribuiti su più processi.

Limite:

- La maggior parte delle letture e degli aggiornamenti della cache sono remoti.

Ad esempio, per memorizzare nella cache 10 GB di dati con un massimo di 1 GB per JVM, sono richiesti dieci Java virtual machine. Il numero di partizioni deve pertanto essere impostato su 10 o più. Idealmente, il numero di partizioni deve essere impostato su un numero primo in cui ogni frammento memorizza una quantità ragionevole di memoria. In genere, l'impostazione numberOfPartitions è uguale al numero di Java virtual machine. Con questa impostazione, ogni JVM memorizza una partizione. Se si abilita la replica, si deve aumentare il numero di Java virtual machine nel sistema. Altrimenti, ogni JVM memorizza anche una partizione della replica, che consuma tanta memoria quanto una partizione primaria.

Per aumentare al massimo le prestazioni della configurazione scelta, consultare la sezione relativa al dimensionamento della memoria e al calcolo del numero di partizioni in *Guida alla gestione*

Ad esempio, in un sistema con 4 Java virtual machine e il valore dell'impostazione numberOfPartitions pari a 4, ogni JVM ospita una partizione primaria. Una operazione di lettura ha il 25 percento di possibilità di eseguire il fetch dei dati da una partizione disponibile in locale, che è molto più veloce se confrontata con il richiamo dei dati da una JVM remota. Se un'operazione di lettura, come l'esecuzione di una query, deve eseguire il fetch di una raccolta di dati che coinvolge 4 partizioni in egual misura, il 75 percento delle chiamate sono remote e il 25 percento delle chiamate sono locali. Se l'impostazione ReplicaMode è

impostata su SYNC o ASYNC e l'impostazione ReplicaReadEnabled è impostata su true, quattro partizioni della replica vengono create e distribuite su quattro Java virtual machine. Ogni JVM ospita una partizione primaria e una partizione della replica. La possibilità che l'operazione di lettura venga eseguita localmente aumenta del 50 percento. L'operazione di lettura che esegue il fetch di una raccolta di dati che coinvolge quattro partizioni in egual misura possiede il 50 percento di chiamate remote e il 50 di chiamate locali. Le chiamate locali sono molto più veloci di quelle remote. Quando si verificano chiamate remote, le prestazioni calano.

Topologia remota

Una topologia remota memorizza tutti i dati nella cache in uno o più processi separati, riducendo l'utilizzo della memoria dei processi dell'applicazione. È possibile avvantaggiarsi della distribuzione dei dati in processi separati distribuendo una griglia eXtreme Scale replicata e suddivisa in partizioni. Rispetto alle configurazioni incorporate e a quelle incorporate suddivise in partizioni descritte nelle sezioni precedenti, se si desidera gestire la griglia remota, è necessario eseguire queste operazioni indipendentemente dall'applicazione e dal provider JPA. Per ulteriori informazioni sulla gestione della distribuzione di una griglia eXtreme Scale, consultare la sezione relativa al monitoraggio dell'ambiente di distribuzione.

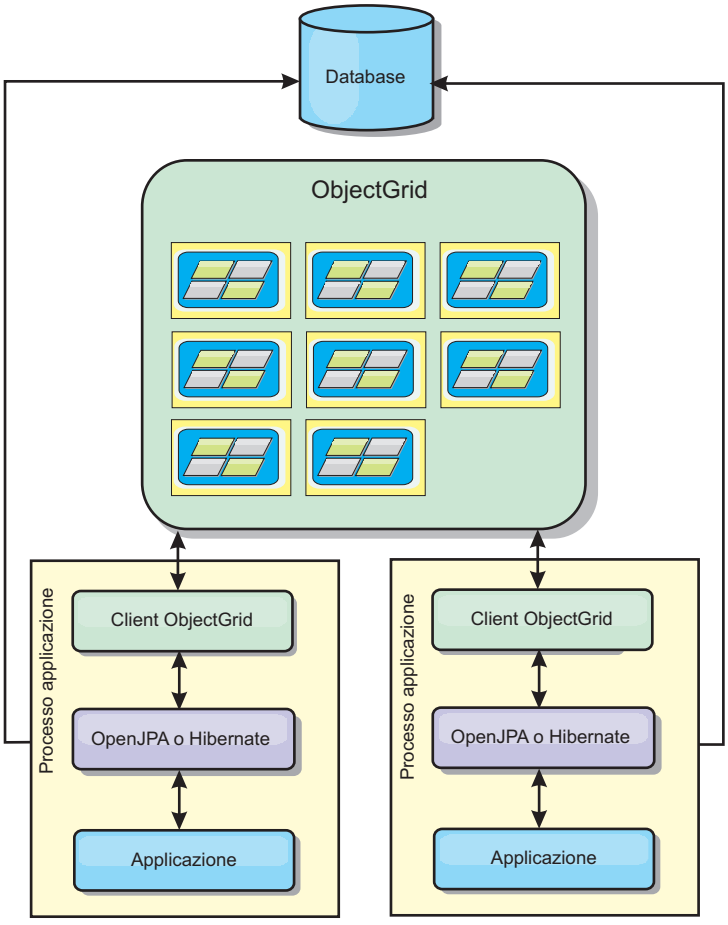


Figura 32. Topologia JPA remota

Vantaggi:

- Memorizza grandi quantità di dati.

- Il processo dell'applicazione non contiene dati nella cache.
- Gli aggiornamenti della cache sono distribuiti su più processi.
- Opzioni di configurazione molto flessibili.

Limite:

- Tutte le letture e gli aggiornamenti della cache sono remoti.

Configurazione

Per ulteriori informazioni sulla configurazione dei plug-in della cache JPA, consultare la sezione sui plug-in in *Guida alla programmazione*.

Gestione della sessione HTTP

Il gestore replica di sessioni contenuto in WebSphere eXtreme Scale può essere utilizzato con il gestore sessioni predefinito sul server delle applicazioni per replicare i dati della sessione da un processo ad un altro processo così da supportare l'elevata disponibilità dei dati della sessione utente.

Funzioni

Il gestore sessioni è stato progettato in modo da poter essere eseguito in qualsiasi contenitore Java Platform, Enterprise Edition Versione 1.4. Poiché il gestore sessioni non ha alcuna dipendenza dalle API WebSphere, può supportare diverse versioni di WebSphere Application Server così come ambienti di server delle applicazioni dei fornitori.

Il gestore sessioni HTTP fornisce capability di replica della sessione per un'applicazione associata. Il gestore replica di sessioni lavora con il gestore sessioni del contenitore Web per creare sessioni HTTP e gestire cicli di vita di sessioni HTTP associate all'applicazione. Queste attività di gestione dei cicli di vita includono: l'invalidazione delle sessioni basate su un timeout o un servlet esplicito o una chiamata JSP (JavaServer Pages) e il richiamo dei listener di sessione associati alla sessione o all'applicazione web. Il gestore sessioni esegue la persistenza delle sessioni in un'istanza ObjectGrid. Questa istanza può essere locale, in memoria o un'istanza replicata completamente, suddivisa in cluster o partizioni. L'utilizzo dell'ultima topologia consente al gestore sessioni di fornire supporto per il failover delle sessioni HTTP quando i server delle applicazioni vengono chiusi o terminano in modo imprevisto. Il gestore sessioni può anche essere utilizzato in ambienti che non supportano affinità, quando l'affinità non viene applicata da un livello del sistema di bilanciamento del carico che diffonde le richieste al livello server delle applicazioni.

Scenari di utilizzo

Il gestore sessioni può essere utilizzato nei seguenti scenari:

- Negli ambienti che utilizzano i server delle applicazioni di versioni differenti di WebSphere Application Server, come in un classico scenario di migrazione.
- Nelle distribuzioni che utilizzano server delle applicazioni di fornitori differenti. Ad esempio, un'applicazione che viene sviluppata su server delle applicazioni open source e che viene ospitata su WebSphere Application Server. Un altro esempio è un'applicazione che viene promossa dallo staging alla produzione. La migrazione seamless di queste versioni dei server delle applicazioni è possibile finché tutte le sessioni HTTP sono attive e servite.

- Negli ambienti in cui è necessario che l'utente esegua la persistenza delle sessioni con livelli QoS (Quality of Service) più elevati e migliori garanzie di disponibilità delle sessioni durante il failover del server dei livelli QoS WebSphere Application Server predefiniti.
- In un ambiente in cui l'affinità di sessione non può essere garantita o in ambienti in cui l'affinità è gestita da un sistema di bilanciamento del carico del fornitore ed è quindi necessario che il meccanismo di affinità venga personalizzato su tale sistema di bilanciamento del carico.
- In un ambiente per scaricare il sovraccarico della gestione sessioni e memorizzare in un processo Java esterno.
- In più celle per abilitare il failover di sessioni tra celle.
- In più centri di dati o in più zone.

Modalità di funzionamento del gestore sessioni

Il gestore delle repliche delle sessioni utilizza listener di sessioni standard per ascoltare le modifiche dei dati delle sessioni, ed esegue la persistenza dei dati della sessione in un'istanza ObjectGrid in locale o in remoto. I dati della sessione vengono ricaricati nel percorso richiesto mediante standard standard derivanti dall'istanza ObjectGrid in locale o in remoto. È possibile aggiungere il listener della sessione ed il filtro del servlet ad ogni modulo Web della propria applicazione con gli strumenti forniti con WebSphere eXtreme Scale. Questi listener e filtri possono anche essere aggiunti manualmente al descrittore di distribuzione Web della propria applicazione.

Questo gestore replica di sessioni è utilizzabile con ciascun gestore sessioni del contenitore Web del fornitore per la replica di dati della sessione tra Java virtual machine. Quando il server originale smette di funzionare, gli utenti possono richiamare i dati della sessione da altri server.

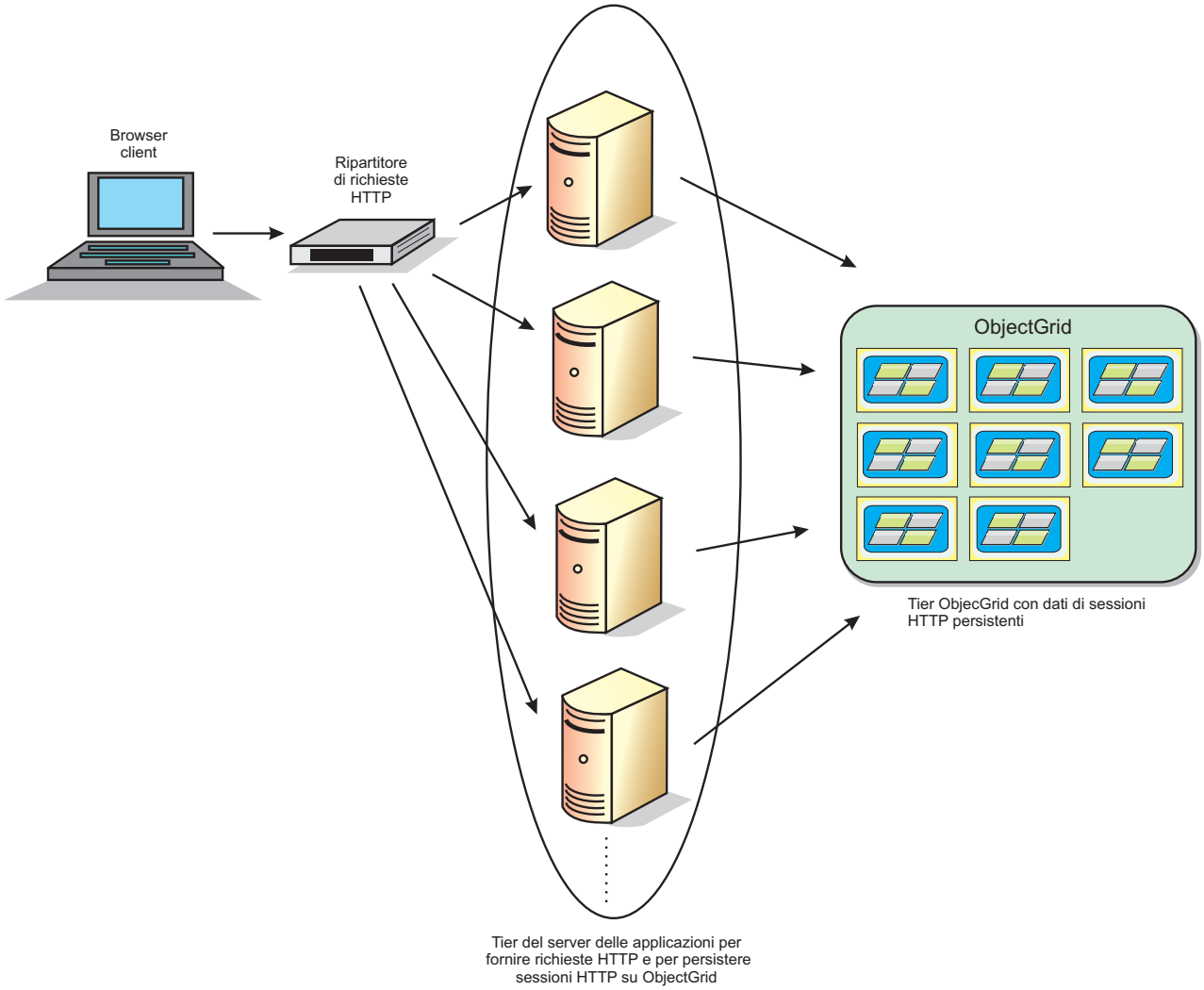


Figura 33. Topologia della gestione sessioni HTTP con una configurazione remota del contenitore

Topologie di distribuzione

Il gestore sessioni può essere configurato utilizzando due diversi scenari di distribuzione dinamica:

- **Contenitori eXtreme Scale incorporati, collegati alla rete**

  In questo scenario, i server eXtreme Scale sono collocati negli stessi processi dei servlet. Il gestore sessioni può comunicare direttamente con l'istanza ObjectGrid locale, evitando costosi ritardi di rete. Questo scenario è da preferire quando l'esecuzione con affinità e prestazioni è fondamentale.
- **Contenitori eXtreme Scale remoti, collegati alla rete**

  In questo scenario, i server eXtreme Scale vengono eseguiti in processi esterni in cui vengono eseguiti i servlet. Il gestore sessioni comunica con una griglia del server remoto eXtreme Scale. Questo scenario è da preferire quando il livello del contenitore Web non ha la memoria necessaria per memorizzare i dati della sessione. I dati della sessione saranno scaricati in un livello separato, il che porterà ad un minore utilizzo della memoria nel livello contenitore, ma ad una maggiore latenza a causa dell'ubicazione remota dei dati.

Avvio generico di un contenitore incorporato

eXtreme Scale avvia automaticamente un contenitore ObjectGrid incorporato all'interno di qualsiasi processo del server delle applicazioni quando il contenitore Web inizializza il listener della sessione o il filtro servlet, se la proprietà objectGridType è impostata su EMBEDDED. Consultare Parametri di inizializzazione del contesto servlet per i dettagli.

Con eXtreme Scale Versione 7.1 non è più necessario impacchettare un file ObjectGrid.xml ed uno objectGridDeployment.xml nel file WAR o EAR dell'applicazione Web. I file ObjectGrid.xml e objectGridDeployment.xml predefiniti sono impacchettati nel prodotto JAR. Per impostazione predefinita le mappe dinamiche vengono create per contesti di applicazioni Web differenti. Le mappe di eXtreme Scale statiche sono ancora supportate.

Questo approccio per l'avvio di contenitori ObjectGrid incorporati è applicabile a ciascun tipo di server delle applicazioni. Gli approcci che prevedono un componente WebSphere Application Server o WebSphere Application Server Community Edition GBean sono obsoleti.

Gestore replica di sessione basata su Listener

Il gestore replica di sessione eXtreme Scale in commercio con WebSphere eXtreme Scale può lavorare sul server delle applicazioni con il gestore predefinito di sessione per replicare dati da un processo ad un altro processo così da supportare un'elevata disponibilità di dati della sessione utente.

Il gestore sessioni è stato progettato in modo che possa funzionare su qualunque contenitore di piattaforma Java™, Enterprise Edition Versione 1.4. Il gestore sessioni non ha dipendenze sulle API WebSphere, per cui è in grado di supportare varie versioni di WebSphere Application Server così come ambienti server delle applicazioni di fornitori.

Il gestore sessioni HTTP fornisce capacità di replica della sessione per un'applicazione associata. Il gestore replica di sessione lavora con il gestore sessioni del contenitore per creare sessioni HTTP e pe rgestire cicli di vita di sessioni HTTP associate all'applicazione. Tali attività di gestione del ciclo di vita includono: l'invalidazione di sessioni basate su un timeout o su un servlet esplicito o su chiamata JSP (JavaServer Pages) ed il richiamo di listener di sessione associati alla sessione o all'applicazione Web. Il gestore sessioni continua le sue sessioni in un'istanza ObjectGrid. Questa istanza può essere locale, in memoria o completamente replicata, suddivisa in cluster e ad istanza partizionata. L'utilizzo dell'ultima topologia consente al gestore sessioni di fornire supporto al failover della sessione HTTP quando i server delle applicazioni sono spenti o quando terminano in modo inaspettato. Il gestore sessioni può inoltre funzionare in ambienti che non supportano affinità, quando l'affinità non viene rafforzata da un livello del sistema di bilanciamento del carico che diffonde le richieste al livello server delle applicazioni.

Scenari di utilizzo

Il gestore sessioni può essere utilizzato nei seguenti scenari:

- Negli ambienti che utilizzano server delle applicazioni di versioni differenti di WebSphere Application Server, come un classico scenario di migrazione.

- Nelle distribuzioni che utilizzano server delle applicazioni di fornitori differenti. Ad esempio, un'applicazione che viene sviluppata su server delle applicazioni open source ospitati su WebSphere Application Server. Un altro esempio è un'applicazione che viene promossa dallo staging alla produzione. La migrazione seamless di queste versioni dei server delle applicazioni è possibile finché tutte le sessioni HTTP sono attive e servite.
- Negli ambienti in cui è necessario che l'utente esegua la persistenza delle sessioni con livelli QoS (Quality of Service) più elevati e migliori garanzie di disponibilità delle sessioni durante il failover del server rispetto ai livelli QoS predefiniti di WebSphere Application Server.
- In un ambiente in cui l'affinità di sessione non può essere garantita o in ambienti in cui l'affinità è gestita da un sistema di bilanciamento del carico del fornitore ed è necessario che il meccanismo di affinità venga personalizzato su quel sistema di bilanciamento del carico.
- In un ambiente per scaricare il sovraccarico di gestione sessioni e memorizzare in un processo Java esterno.
- In più celle per abilitare il failover di sessioni tra celle.
- In più centri di dati o in più zone.

Dettagli del gestore sessioni

Il gestore delle repliche delle sessioni utilizza listener di sessioni standard per ascoltare le modifiche dei dati delle sessioni, ed esegue la persistenza dei dati della sessione in un'istanza ObjectGrid in locale o in remoto. I dati della sessione vengono ricaricati nel percorso richiesto mediante standard derivanti dall'istanza ObjectGrid in locale o in remoto. È possibile aggiungere il listener della sessione ed il filtro del servlet ad ogni modulo Web della propria applicazione con strumenti forniti con WebSphere eXtreme Scale. Questi listener e filtri possono anche essere aggiunti al descrittore di distribuzione Web della propria applicazione.

Il gestore di replica delle sessioni funziona con ogni gestore sessioni di base del fornitore per replicare i dati della sessione dell'applicazione. Notare le seguenti considerazioni.

- Scegliere i contenitori ObjectGrid incorporati o i contenitori ObjectGrid in remoto a seconda dei requisiti delle prestazioni e delle dimensioni dei propri dati. Lo scenario incorporato fornisce la configurazione più semplice e la migliore prestazione.
- Scegliere il numero dei contenitori ObjectGrid in remoto per contenitore Web in base alle dimensioni dei dati dell'utente.
- Scegliere di memorizzare i dati dell'intera sessione insieme oppure di memorizzare separatamente ciascun attributo a seconda del numero e della dimensione degli attributi dei dati utente e della frequenza con cui cambiano.
- Scegliere l'intervallo di replica. Meno aggressivo è l'intervallo di replica e migliori saranno le prestazioni.
- Scegliere la dimensione della tabella della sessione per bilanciare la dimensione della memoria locale e le prestazioni. Il prodotto scarica dati utente di sessione quando viene raggiunta la dimensione massima della cache della sessione locale.
- Il prodotto supporta le sessioni HTTP all'interno del contesto dell'applicazione secondo le specifiche del Servlet, per evitare problemi legati alla sicurezza e al conflitto per l'assegnazione dei nomi dell'attributo. È possibile condividere sessioni per tutto il contesto dell'applicazione mediante la propria classe singleton.

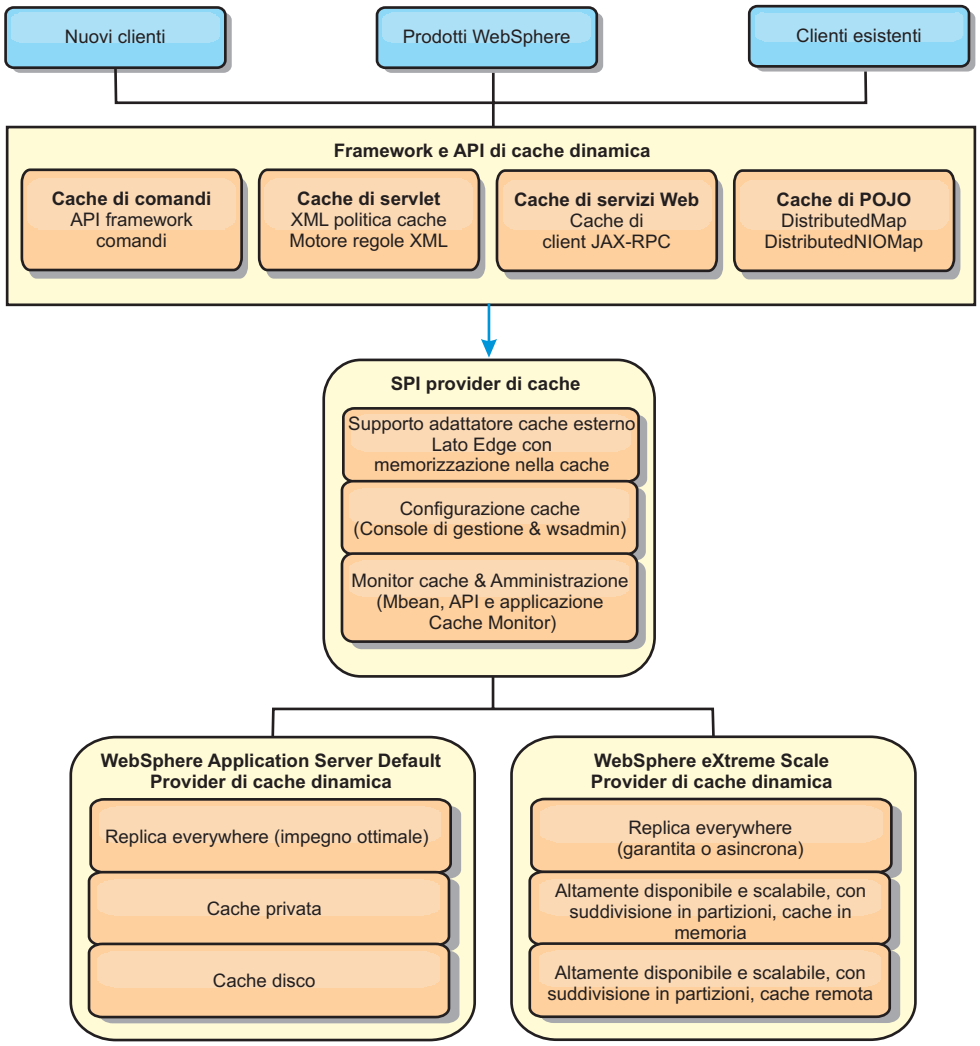
- Il gestore di replica delle sessioni esegue la replica dei dati della sessione per l'HA (high availability) mediante il listening delle modifiche dei dati della sessione ed il ricaricamento dei dati memorizzati della sessione su richiesta. Questo viene realizzato mediante il riutilizzo per ogni fornitore del gestore sessioni di base del contenitore Web. La sessione viene collegata mediante vari ID nativi di sessione. Il failover dei dati della sessione viene eseguito da un contenitore web ad un altro ricaricando i dati della sessione da un'istanza ObjectGrid. L'ID sessione e l'orario di creazione possono differire prima e dopo il failover.

Provider di cache dinamica

L'API Dynamic Cache è disponibile per le applicazioni Java EE distribuite in WebSphere Application Server. Il provider della cache dinamica può essere sfruttato per la cache dei dati di business, per gli HTML generati o per sincronizzare i dati collocati nella cache nella cella utilizzando DRS (data replication service).

Panoramica

Precedentemente, l'unico provider del servizio per l'API Dynamic Cache era il motore della cache dinamica integrato in WebSphere Application Server. I clienti possono utilizzare l'interfaccia del provider del servizio di cache dinamica in WebSphere Application Server per collegare eXtreme Scale alla cache dinamica. Attivando questa funzionalità, è possibile abilitare le applicazioni scritte con l'API Dynamic Cache o le applicazioni che utilizzano la memorizzazione nella cache a livello contenitore (come ad esempio i servlet) per sfruttare le funzioni e le capability della prestazione di WebSphere eXtreme Scale.



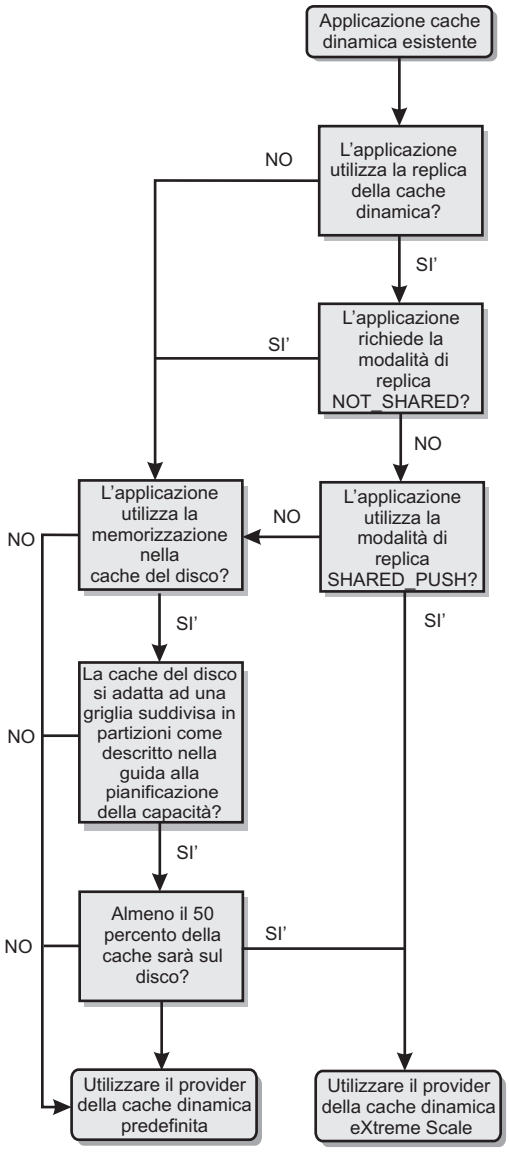
È possibile installare e configurare il provider della cache dinamica come descritto in Configurazione provider cache dinamica per WebSphere eXtreme Scale.

Decidere come sfruttare WebSphere eXtreme Scale

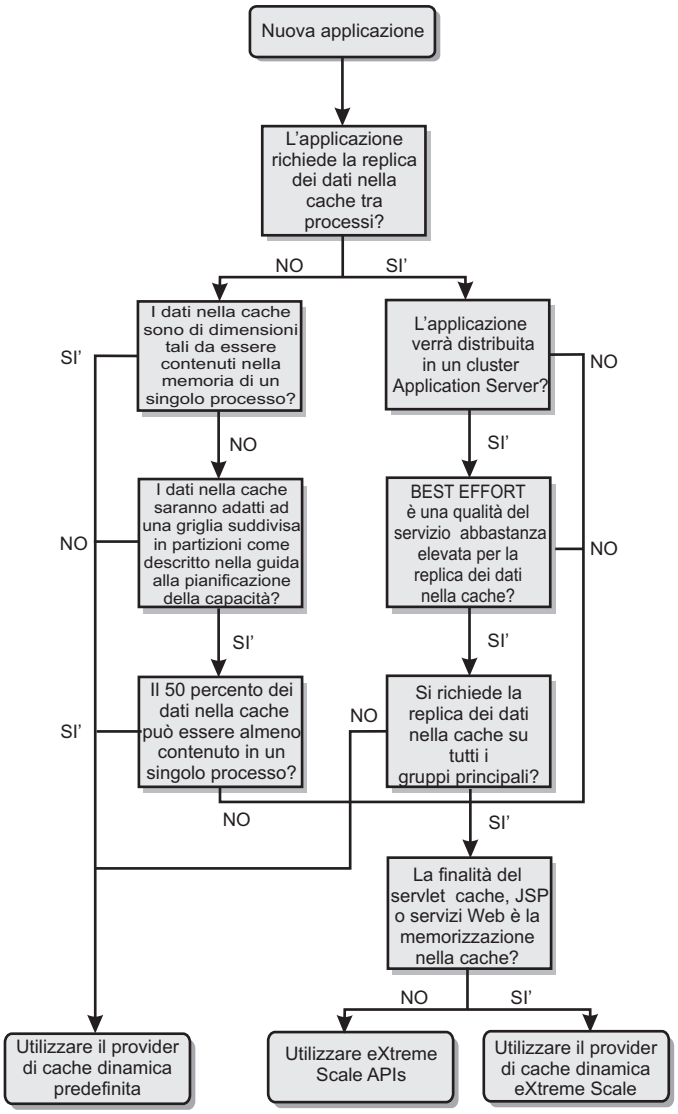
Le funzioni disponibili in WebSphere eXtreme Scale aumentano in modo significativo le capability distribuite dell'API Dynamic Cache al di là di ciò che viene offerto dal motore della cache dinamica predefinito e dal servizio di replica dei dati. Con eXtreme Scale, è possibile creare le cache che vengono veramente distribuite tra più server, anziché solo replicati e sincronizzati tra server. Inoltre, le cache di eXtreme Scale sono transazionali e di elevata disponibilità, garantendo che ogni server veda gli stessi contenuti per il servizio di cache dinamica. WebSphere eXtreme Scale offre una maggiore qualità del servizio per la replica della cache rispetto a DRS.

Tuttavia, questi vantaggi non implicano che il provider della cache dinamica di eXtreme Scale sia la scelta giusta per qualsiasi applicazione. Utilizzare le strutture ad albero per le decisioni e la matrice di confronto delle funzioni riportate di seguito per determinare qual è la tecnologia più adatta per la propria applicazione.

La struttura ad albero per le decisioni per la migrazione delle applicazioni di cache dinamica esistenti



Struttura ad albero per le decisioni per la scelta di un provider della cache per le nuove applicazioni



Confronto delle funzioni

Tabella 7. Confronto delle funzioni

Funzioni cache	Provider predefinito	eXtreme Scale provider	eXtreme Scale API
Locale, cache in memoria	x	x	x
Memorizzazione nella cache distribuita	Incorporata	Incorporata, incorporata-suddivisa in partizioni e suddivisa in partizioni-remota	Molteplice
Scalabile in modo lineare		x	x

Tabella 7. Confronto delle funzioni (Continua)

Funzioni cache	Provider predefinito	eXtreme Scale provider	eXtreme Scale API
Replica affidabile (sincrona)		ORB	ORB
Overflow del disco	x		
Eliminazione	LRU/TTL/basato sulla heap	LRU/TTL (per partizione)	Molteplice
Invalidazione	x	x	x
Relazioni	ID dipendenze, modelli	ID dipendenze, modelli	x
Ricerche non per chiavi			Query e indice
Integrazione back-end			Programmi di caricamento
Transazionale		Implicito	x
Memoria basata su chiavi	x	x	x
Eventi e listener	x	x	x
Integrazione WebSphere Application Server	Solo cella singola	Più celle	Cella indipendente
Supporto Java edizione standard		x	x
Monitoraggio e statistiche	x	x	x
Sicurezza	x	x	x

Tabella 8. Integrazione tecnologia seamless

Funzioni cache	Provider predefinito	eXtreme Scale provider	eXtreme Scale API
Memorizzazione nella cache dei risultati del servlet/JSP di WebSphere Application Server servlet/JSP	V5.1+	V6.1.0.25+	
Memorizzazione nella cache dei risultati di WebSphere Application Server Web Services (JAX-RPC)	V5.1+	V6.1.0.25+	
Memorizzazione nella cache della sessione HTTP			x
Provider della cache per OpenJPA e Hibernate			x

Tabella 8. Integrazione tecnologia seamless (Continua)

Sincronizzazione database tramite OpenJPA e Hibernate			x

Tabella 9. Interfacce di programmazione

Funzioni cache	Provider predefinito	eXtreme Scale provider	eXtreme Scale API
API basata su comandi	API framework comandi	API framework comandi	API DataGrid
API basata su mappa	API DistributedMap	API DistributedMap	API ObjectMap
API EntityManager			x

Per una descrizione più dettagliata su come funzionano le cache distribuite di eXtreme Scale, consultare le informazioni sulla configurazione di distribuzione in *Guida alla gestione*.

Nota: Una cache distribuita di eXtreme Scale può soltanto memorizzare le voci in cui la chiave e il valore implementano entrambi l'interfaccia java.io.Serializable.

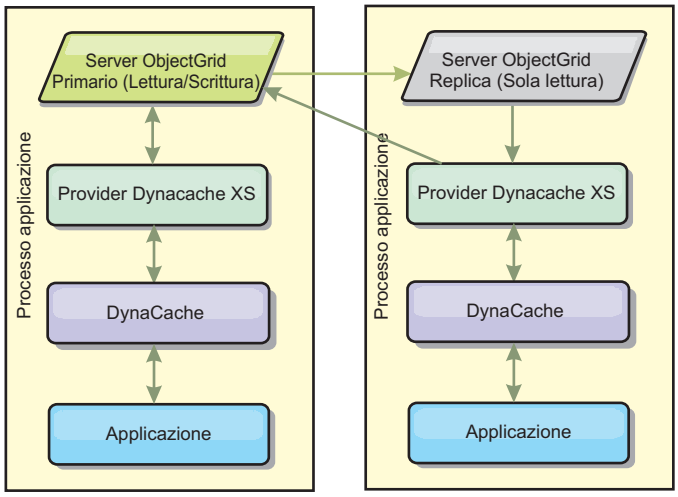
Tipi di topologia

Un servizio di cache dinamica creato con il provider della eXtreme Scale può essere distribuito in tutte e tre le topologie disponibili, consentendo di personalizzare la cache specificamente per esigenze amministrative, di risorse e prestazione. Queste topologie sono incorporate, incorporate suddivise in partizioni e remote.

Topologia incorporata

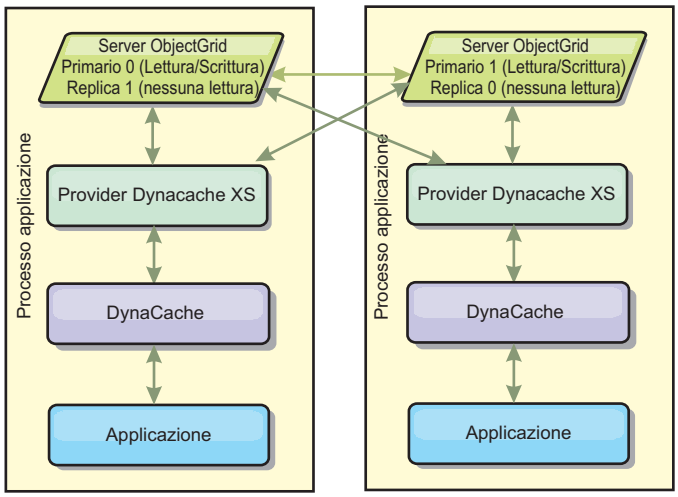
La topologia incorporata è simile alla cache dinamica e al provider DRS. Le istanze di cache distribuita create con la topologia incorporata conservano una copia integrale della cache all'interno di ogni processo eXtreme Scale che accede al servizio di cache dinamica, consentendo a tutte le operazioni in lettura di essere eseguite localmente. Tutte le operazioni di scrittura passano attraverso un processo di singolo server, in cui i blocchi transazionali vengono gestiti prima di essere replicati al resto dei server. Di conseguenza, questa topologia è più adatta per i carichi di lavoro in cui le operazioni di lettura cache superano in gran numero le operazioni di scrittura cache.

Con la topologia incorporata, le voci cache nuove o aggiornate non sono immediatamente visibili su ogni singolo processo del server. Una voce cache non sarà visibile, persino al server che lo ha generato, finché non viene propagata tramite i servizi di replica asincrona di WebSphere eXtreme Scale. La velocità del funzionamento di questi servizi dipenderà dall'hardware, ma c'è ancora un piccolo ritardo. La topologia incorporata viene mostrata nella seguente immagine:



Topologia suddivisa in partizioni incorporata

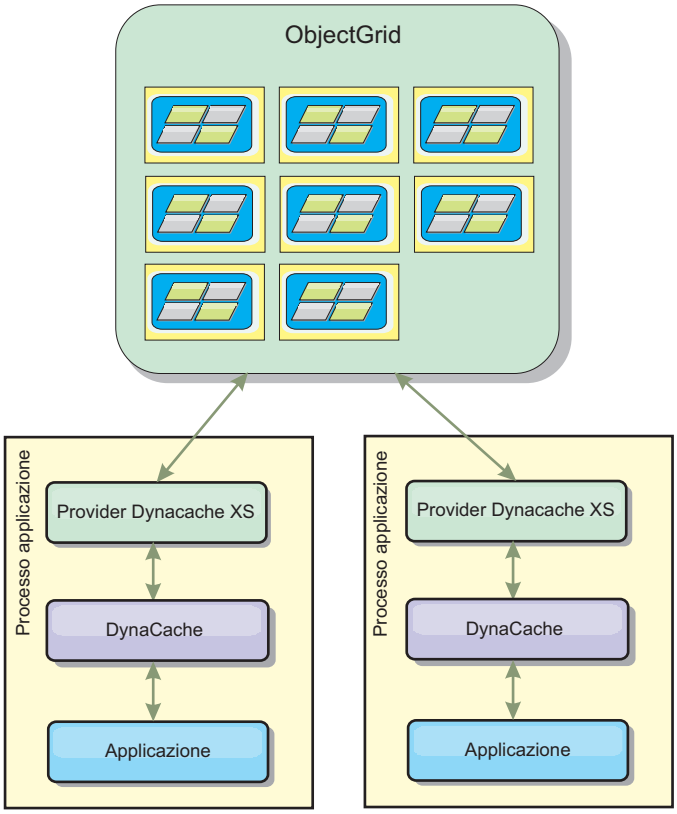
Per i carichi di lavoro in cui le scritture di cache si verificano tanto spesso quanto o più frequentemente delle letture, le topologie remote o suddivise in partizioni incorporate sono consigliate. La topologia suddivisa in partizioni conserva tutti i dati cache all'interno dei processi di WebSphere Application Server che accedono alla cache. Tuttavia, ogni processo memorizza solo una parte dei dati cache. Tutte le letture e scritture relative ai dati ubicati in questa “partizione” passano attraverso il processo, significando che la maggior parte delle richieste alla cache verranno soddisfatte con una chiamata della procedura remota. Ne consegue una maggiore latenza per le operazioni di lettura rispetto alla topologia incorporata, ma la capability della cache distribuita per gestire le operazioni di lettura e scrittura scalerà in modo lineare con il numero dei processi di WebSphere Application Server che accedono alla cache. Inoltre, con questa topologia, la dimensione massima della cache non è vincolata dalla dimensione di un singolo processo di WebSphere. Poiché ogni processo carica solo una parte delle cache, la dimensione massima della cache diventa la dimensione aggregata di tutti i processi, meno i sovraccarichi del processo. La topologia suddivisa in partizioni incorporata viene mostrata nella seguente immagine:



Ad esempio, supponendo di disporre di una griglia di processi server con 256 megabyte di heap libera, ciascuna per ospitare un servizio di cache dinamica. Il provider della cache dinamica predefinito e il provider eXtreme Scale che utilizzano la topologia incorporata sarebbero entrambi limitati a una dimensione della cache in memoria di 256 megabyte meno i sovraccarichi. Vedere la sezione Pianificazione della capacità e alta disponibilità più avanti in questo documento. Il provider eXtreme Scale che utilizza la topologia suddivisa in partizioni incorporata verrebbe limitata ad una dimensione della cache di un gigabyte meno i sovraccarichi. In tal modo, il provider WebSphere eXtreme Scale rende possibile la disponibilità di servizi di cache dinamica in memoria che sono più grandi della dimensione di un singolo processo del server. Il provider della cache dinamica predefinita cache dipende sull'utilizzo della cache del disco per consentire alle istanze della cache di crescere al di là della dimensione di un singolo processo. In molte situazioni, il provider WebSphere eXtreme Scale può eliminare l'esigenza della cache del disco e dei costosi sistemi di memoria su disco necessari per farli funzionare.

Topologia remota

La topologia remota può essere utilizzata inoltre per eliminare la necessità della cache del disco. L'unica differenza tra le topologie suddivise in partizioni incorporata e quelle remote è che tutti i dati cache vengono memorizzati all'esterno dei processi di WebSphere Application Server quando si utilizza la topologia remota. WebSphere eXtreme Scale supporta i processi contenitori autonomi per i dati cache. Questi processi contenitori hanno minori sovraccarichi di un processo di WebSphere Application Server e inoltre non sono vincolati ad utilizzare una particolare macchina virtuale JVM Java. Ad esempio, i dati per un servizio di cache dinamica cui accede un processo di WebSphere Application Server a 32-bit potrebbe essere posizionato in un processo contenitore di eXtreme Scale in esecuzione su una JVM a 64-bit. Ciò consente agli utenti di sfruttare la maggiore capacità di memoria dei processi a 64 bit per la memorizzazione nella cache, senza incorrere in sovraccarichi aggiuntivi di 64 bit per i processi del server delle applicazioni. La topologia remota viene mostrata nella seguente immagine:



Compressione dati

Un'altra funzione della prestazione offerta dal provider della cache dinamica di WebSphere eXtreme Scale che può supportare gli utenti a gestire i sovraccarichi per la cache è la compressione. Il provider della cache dinamica predefinito non consente la compressione dei dati cache in memoria. Con il provider eXtreme Scale, ciò diventa possibile. La compressione della cache che utilizza l'algoritmo Deflate può essere abilitata su una qualsiasi delle tre topologie distribuite. Abilitando la compressione, aumenteranno il sovraccarico relativo alle operazioni di lettura e scrittura ma si ridurrà drasticamente la densità della cache per applicazioni come la memorizzazione nella cache JSP e servlet.

Cache in memoria locale

Il provider della cache dinamica di WebSphere eXtreme Scale può essere inoltre utilizzato per supportare le istanze di cache dinamica che hanno la **replica disabilitata**. Come il provider della cache dinamica predefinita, queste cache possono memorizzare dati non serializzabili. Possono inoltre offrire una migliore prestazione del provider della cache dinamica predefinita su server enterprise a più processori in quanto il percorso del codice di eXtreme Scale è progettato per ottimizzare la contemporaneità della cache in memoria.

Il motore della cache dinamica e le differenze funzionali di eXtreme Scale

Nel caso di cache in memoria locali in cui la replica è disabilitata, non dovrebbero esserci differenze funzionali significative tra le cache supportate dal provider della cache dinamica predefinita e WebSphere eXtreme Scale. Gli utenti non dovrebbero avvertire una differenza funzionale tra le due cache tranne che le cache di

WebSphere eXtreme Scale non supportano l'offload del disco o statistiche e operazioni collegate alla dimensione della cache in memoria.

Nel caso di cache in cui la replica è abilitata non ci saranno differenze significative nei risultati restituiti dalla maggior parte delle chiamate di API Dynamic Cache, indipendentemente se il cliente sta utilizzando il provider della cache dinamica predefinito o il provider della cache dinamica di eXtreme Scale. Per alcune operazioni non è possibile emulare il comportamento del motore della cache dinamica utilizzando eXtreme Scale.

Statistiche della cache dinamica

Le statistiche della cache dinamica vengono riportate tramite l'applicazione CacheMonitor o l'MBean della cache dinamica. Quando si utilizza il provider della di cache dinamica di eXtreme Scale, le statistiche verranno riportate ancora tramite queste interfacce, ma il contesto dei valori statistici sarà diverso.

Se un'istanza di cache dinamica è condivisa tra tre server denominati A, B e C, l'oggetto di statistiche della cache dinamica restituisce solo le statistiche per la copia della cache sul server dove è stata effettuata la chiamata. Se le statistiche vengono richiamate sul server A, riflettono solo l'attività sul server A.

Con eXtreme Scale, solo un'unica cache distribuita è condivisa tra tutti i server, pertanto non è possibile tracciare la maggior parte delle statistiche server per server, come fa invece il provider della di cache dinamica. Segue un elenco delle statistiche riportato dalle API di statistiche cache e ciò che rappresentano quando si utilizza il provider della cache dinamica WebSphere eXtreme Scale. Come il provider predefinito, queste statistiche non sono sincronizzate e pertanto possono variare fino al 10% per carichi di lavoro contemporanei.

- **Corrispondenze nella cache** : le corrispondenze nella cache vengono tracciate per server. Se il traffico sul Server A genera 10 corrispondenze nella cache e il traffico sul Server B genera 20 corrispondenze nella cache, le statistiche della cache riporteranno 10 corrispondenze nella cache sul Server A e 20 corrispondenze nella cache sul Server B.
- **Mancata corrispondenza nella cache**: le mancate corrispondenze nella cache vengono tracciate per server proprio come le corrispondenze nella cache.
- **Voci della cache della memoria**: questa statistica riporta il numero di voci della cache nella cache distribuita. Ogni server che accede alla cache riporterà lo stesso valore per questa statistica e quel valore sarà il numero totale di voci della cache nella memoria su tutti i server.
- **Dimensione cache della memoria in MB**: questa metrica è supportata solo per cache che utilizzano le topologie remote, embedded o embedded_partitioned. Riporta il numero di megabyte di spazio heap Java utilizzato dalla cache nell'intera griglia. Questa statistica reporta l'utilizzo heap solo per partizioni primarie; è necessario prendere in considerazione le repliche. Poiché l'impostazione predefinita per le topologie remote e embedded_partitioned è di una replica asincrona, raddoppiare questo numero per ottenere il reale consumo di memoria della cache.
- **Rimozioni cache**: questa statistica riporta il numero totale di voci rimosse dalla cache tramite qualsiasi metodo ed è un valore aggregato per l'intera cache distribuita. Se il traffico sul Server A genera 10 invalidazioni e il traffico sul Server B genera 20 invalidazioni, il valore su entrambi i server sarà 30.

- **Rimozioni LRU (Cache Least Recently Used) dalla cache**: questa statistica viene aggregata, come le rimozioni dalla cache. Traccia il numero di voci rimosse per mantenere la dimensione massima della cache.
- **Invalidazioni di timeout**: anche questa è una statistica aggregata e traccia il numero di voci rimosse perché in timeout.
- **Invalidazioni esplicite** : ancora una statistica aggregata, questa traccia il numero di voci rimosse con invalidazione diretta per chiave, ID dipendenza o modello.
- **Statistiche estese** : il provider della cache dinamica di eXtreme Scale esporta le seguenti stringhe chiave di statistiche estese.
  - **com.ibm.websphere.xs.dynacache.remote_hits**: il numero totale di corrispondenze della cache tracciate nel contenitore di eXtreme Scale. Questa è una statistica aggregata e il relativo valore nella mappa di statistiche estese è `long`.
  - **com.ibm.websphere.xs.dynacache.remote_misses**: il numero totale di mancate corrispondenze nella cache tracciate nel contenitore di eXtreme Scale. Una statistica aggregata, il relativo valore nella mappa di statistiche estese è uno `long`.

Report statistiche di reimpostazione

Il provider della cache dinamica consente di reimpostare le statistiche della cache. Con il provider predefinito l'operazione di reimpostazione elimina solo le statistiche del server interessato. Il provider della cache dinamica eXtreme Scale traccia la maggior parte dei dati statistici nei contenitori di cache remota. Questi dati non vengono eliminati o modificati quando le statistiche vengono reimpostate. Invece il comportamento della cache dinamica predefinita viene simulato sul client evidenziando la differenza tra il valore corrente di una determinata statistica e il valore di quella statistica l'ultima volta che la reimpostazione è stata chiamata su quel server.

Ad esempio, se il traffico sul Server A genera 10 rimozioni di cache, le statistiche sul Server A e sul Server B riporteranno 10 rimozioni. Se le statistiche sul Server B vengono reimpostate e il traffico sul Server A genera 10 rimozioni aggiuntive, le statistiche sul Server A riporteranno 20 rimozioni e le stat sul Server B riporteranno 10 rimozioni.

Eventi di cache dinamica

API Dynamic Cache consente agli utenti di registrare i listener di eventi. Quando si utilizza eXtreme Scale come provider della cache dinamica, i listener di eventi lavorano come previsto per le cache in memoria.

Per le cache distribuite, il comportamento dell'evento dipenderà dalla topologia che si sta utilizzando. Per le cache che utilizzano la topologia incorporata, gli eventi verranno generati sul server che gestisce le operazioni di scrittura, note anche come frammento primario. Ciò significa che solo un server riceverà le notifiche eventi ma avrà tutte le notifiche eventi normalmente previste dal provider della cache dinamica. Poiché WebSphere eXtreme Scale sceglie il frammento primario in fase di runtime, non è possibile assicurare che un particolare processo server riceva sempre questi eventi.

Le cache suddivise in partizioni genereranno eventi su qualsiasi server che ospita una partizione della cache. In questo modo se una cache ha 11 partizioni e ciascun server in una griglia di WebSphere Application Server Network Deployment di 11 server ospita una delle partizioni, ogni server riceve gli eventi di cache dinamica

per le voci di cache che ospita. Nessun singolo processo del server vedrebbe tutti gli eventi a meno che tutte le 11 partizioni non fossero ospitate in quel processo del server. Come per la topologia incorporata, non è possibile assicurare che un particolare processo server riceva una particolare serie di eventi o nessun evento.

Le cache che utilizzano la topologia remota non supportano eventi di cache dinamica.

Chiamate MBean

Il provider della cache dinamica di WebSphere eXtreme Scale non supporta la memorizzazione nella cache del disco. Qualsiasi chiamata MBean relativa alla memorizzazione nella cache del disco non funzionerà.

Associazione della politica di replica cache dinamica

Il provider della cache dinamica integrata di WebSphere Application Server supporta più politiche di replica. Queste politiche possono essere configurate globalmente su ogni voce di cache. Consultare la documentazione sulla cache dinamica per una descrizione di queste politiche di replica.

Il provider della cache dinamica di eXtreme Scale non onora queste politiche direttamente. Queste caratteristiche di replica di una cache vengono determinate dal tipo di topologia distribuita di eXtreme Scale configurata e si applica a tutti i valori collocati in quella cache, indipendentemente dalla politica di replica impostata sulla voce tramite il servizio di cache dinamica. Segue un elenco di tutte le politiche di replica supportate dal servizio di cache dinamica e illustra quale topologia di eXtreme Scale fornisce caratteristiche di replica analoghe.

Notare che il provider della cache dinamica di eXtreme Scale ignora le impostazioni della politica di replica DRS sulla cache o voce di cache. Gli utenti scelgono la topologia appropriata alle proprie esigenze di replica.

- NOT_SHARED – attualmente nessuna delle topologie fornite dal provider della cache dinamica di eXtreme Scale può avvicinarsi a questa politica. Ciò significa che tutti i dati memorizzati nella cache devono avere chiavi e valori che implementino java.io.Serializable.
- SHARED_PUSH – La topologia incorporata si avvicina a questa politica di replica. Quando una voce di cache viene creata, viene replicata a tutti i server. I server cercano le voci di cache localmente. Se una voce non viene trovata localmente, si assume che è inesistente e gli altri server non vengono sottoposti a query.
- SHARED_PULL e SHARED_PUSH_PULL – Le topologie remote e suddivise in partizioni si avvicinano a questa politica di replica. Lo stato distribuito della cache è completamente coerente tra tutti i server.

Queste informazioni vengono fornite principalmente per assicurarsi che la topologia incontri le esigenze dell'utente di coerenza distribuite. Ad esempio, se la topologia incorporata è una scelta migliore per le proprie esigenze di prestazione e distribuzione, ma si richiede il livello di coerenza di cache fornita da SHARED_PUSH_PULL, è opportuno prendere in considerazione l'utilizzo della suddivisione in partizioni, anche se la prestazione può leggermente peggiorare.

Sicurezza

È possibile mettere in sicurezza le istanze di cache dinamica in esecuzione nelle topologie suddivise in partizioni incorporate o incorporate con la funzionalità di sicurezza integrate in WebSphere Application Server. Consultare la documentazione relativa ai server delle applicazioni sulla sicurezza nel WebSphere Application Server Centro informazioni.

Quando una cache è in esecuzione in una topologia remota, è possibile per un client eXtreme Scale autonomo, connettersi alla cache e condizionare il contenuto dell'istanza di cache dinamica. Il provider della cache dinamica di eXtreme Scale ha una funzione di codifica di sovraccarico bassa che può impedire che i dati cache vengano letti o modificati da client non WebSphere Application Server. Per abilitare questa funzione, impostare il parametro facoltativo **com.ibm.websphere.xs.dynacache.encryption_password** sullo stesso valore per ogni istanza di WebSphere Application Server che accede il provider della cache dinamica. Ciò codificherà il valore e i metadati dell'utente per la CacheEntry utilizzando la codifica AES a 128 bit. È molto importante che lo stesso valore venga impostato su tutti i server. I server non riescono a leggere i dati inseriti nella cache dai server con un valore diverso per questo parametro.

Se il provider della eXtreme Scale rileva che vengono impostati valori diversi per questa variabile nella stessa cache, genera un'avvertenza nel log del processo contenitore di eXtreme Scale.

Consultare la documentazione eXtreme Scale sulla sicurezza WebSphere eXtreme Scale se si richiede l'autenticazione SSL o client.

Informazioni aggiuntive

- Cache dinamica Redbook
- Documentazione sulla cache dinamica
  - WebSphere Application Server 7.0
  - WebSphere Application Server 6.1
- Documentazione su DRS
  - WebSphere Application Server 7.0
  - WebSphere Application Server 6.1

Pianificazione della capacità e alta disponibilità (cache dinamica)

L'API Dynamic Cache è disponibile per le applicazioni Java EE distribuite in WebSphere Application Server. La cache dinamica può essere sfruttata per memorizzare dati di business nella cache, generati in HTML, o per sincronizzare i dati memorizzati nella cache nella cella mediante DRS (Data Replication Service).

Panoramica

Tutte le istanze della cache dinamica create con il provider della cache dinamica WebSphere eXtreme Scale hanno un'elevata disponibilità per impostazione predefinita. Il costo dell'alta disponibilità, in termini di memoria e di livello, dipende dalla topologia utilizzata.

Quando si utilizza la topologia incorporata, la dimensione della cache viene limitata alla quantità di memoria libera in un singolo processo del server e ogni processo del server memorizza una copia completa della cache. Finché un singolo

processo del server continua ad essere in esecuzione, la cache resta attiva. I dati cache verranno persi solo se tutti i server che accedono alla cache vengono chiusi.

Per la memorizzazione nella cache che utilizza la topologia incorporata, suddivisa in partizioni, la dimensione della cache è limitata ad un aggregato di spazio libero in tutti i processi del server. Per impostazione predefinita, il provider di cache dinamica eXtreme Scale utilizza 1 replica per ogni frammento primario, così ogni dato memorizzato nella cache viene memorizzato due volte.

Utilizzare la seguente formula A per determinare la capacità di una cache incorporata suddivisa in partizioni.

Formula A

F * C / (1 + R) = M

Dove:

- F = memoria libera per il processo contenitore
- C = numero di contenitori
- R = numero di repliche
- M = dimensione totale della cache

Per una griglia WebSphere Network Deployment con 256 MB di spazio disponibile in ogni processo, con un totale di 4 processi server, un'istanza della cache tra tutti questi server può memorizzare fino a 512 megabyte di dati. In questo modo, la cache riesce a superare una fine anomala di un server senza perdere i dati. Inoltre, è possibile chiudere fino a due server in sequenza senza che vi sia alcuna perdita di dati. Pertanto, per l'esempio precedente, la formula è la seguente:

256mb * 4 contenitori/ (1 primario + 1 replica) = 512mb.

Le cache che utilizzano la topologia remota possiedono caratteristiche di dimensionamento simili alle cache incorporate suddivise in partizioni, ma queste sono limitate dalla quantità di spazio disponibile in tutti i processi del contenitore eXtreme Scale.

Nelle topologie remote, è possibile aumentare il numero di repliche per fornire un più elevato livello di disponibilità a costo di un sovraccarico di memoria aggiuntivo. Nella maggior parte delle applicazioni di cache dinamiche ciò non dovrebbe essere necessario, ma è possibile modificare il file dynacache-remote-deployment.xml per aumentare il numero di repliche.

Utilizzare le seguenti formule, B e C, per determinare l'effetto dell'aggiunta di più repliche nell'HA (High Availability) della cache.

Formula B

N = Minimo(T -1, R)

Dove:

- N = il numero di processi che possono simultaneamente terminare in modo anomalo
- T = il numero totale di contenitori
- R = il numero totale di repliche

Formula C

Ceiling(T/ (1+N)) = m

Dove:
- T = il numero totale di contenitori
- R = il numero totale di repliche
- m = il numero minimo di contenitori necessario per supportare dati cache.

Per l'ottimizzazione delle prestazioni con il provider della cache dinamica, consultare Ottimizzazione del provider della cache dinamica.

Dimensionamento della cache

Prima di poter distribuire un'applicazione che utilizza il provider WebSphere eXtreme Scale Dynamic Cache, i principal generali descritti nella precedente sezione devono essere combinati con i dati ambientali per i sistemi di produzione. Il primo valore numerico da stabilire è il numero totale dei processi del contenitore e la quantità di memoria disponibile in ogni processo per contenere i dati cache. Quando si utilizza la topologia incorporata, i contenitori della cache verranno posti insieme all'interno dei processi di WebSphere Application Server in modo che vi sia un solo contenitore per ogni server che sta condividendo la cache. Determinando il sovraccarico di memoria dell'applicazione senza aver abilitato la memorizzazione nella cache e WebSphere Application Server è il modo migliore per comprendere la quantità di spazio disponibile nel processo. Ciò è possibile analizzando la raccolta dati obsoleti verbosi. Quando si utilizza una topologia remota, è possibile reperire queste informazioni cercando nell'output della raccolta dati obsoleti verbosi di un contenitore autonomo appena avviato che non è stato ancora riempito con i dati cache. Infine è necessario ricordare che, quando si stabilisce la quantità di spazio per processo disponibile per i dati cache, occorre riservare una certa quantità di spazio dell'heap per la raccolta dati obsoleti. Il sovraccarico del contenitore, WebSphere Application Server o autonomo, più la dimensione riservata alla cache deve non superare il 70% dell'heap totale.

Una volta raccolte queste informazioni, è possibile eseguire il plug-in dei valori nella formula A, descritta precedentemente, per descrivere la dimensione massima della cache suddivisa in partizioni. Una volta stabilita la dimensione massima, il passo successivo è determinare il numero totale di voci cache da poter supportare, per cui è necessario determinare la dimensione media per voce cache. Un modo semplice di eseguire tale operazione è aggiungere il 10% alla dimensione dell'oggetto cliente. Per ulteriori informazioni approfondite sulla modifica delle dimensioni delle voci cache durante l'utilizzo di Dynamic Cache, consultare il manuale Tuning guide for dynamic cache and data replication service.

Quando la compressione è abilitata, ciò riguarda la dimensione dell'oggetto cliente, non il sovraccarico del sistema di memorizzazione nella cache. Per determinare la dimensione di un oggetto in cache durante l'utilizzo della compressione, utilizzare la seguente formula:

S = O * C + O * 0.10

Dove:
- S = dimensione media dell'oggetto memorizzato in cache
- O = dimensione media di un oggetto cliente non compresso

- C = rapporto di compressione espresso sotto forma di frazione.

Pertanto, un rapporto di compressione da 2 a 1 è 1/2 = 0,50. È meglio se questo valore è più piccolo. Se l'oggetto, che viene memorizzato, è un POJO normale prevalentemente pieno di tipi primitivi, adottare un rapporto di compressione tra 0,60 e 0,70. Se l'oggetto è un oggetto Servlet, JSP o WebServices, il metodo ottimale per determinare il rapporto di compressione è comprimere un esempio rappresentativo con un programma di utilità di compressione ZIP. Se ciò non è possibile, il rapporto di compressione tra 0,2 e 0,35 è comune per questo tipo di dati.

Successivamente, utilizzare queste informazioni per determinare il numero totale di voci cache che è possibile supportare. Utilizzare la seguente formula D:

Formula D

T = S / A

Dove:

- T= numero totale delle voci cache
- S = dimensione totale disponibile per i dati cache come calcolati mediante la formula A
- A = dimensione media di ogni voce cache

Infine, si deve impostare la dimensione della cache nell'istanza della cache dinamica per far rispettare questo limite. Il provider della cache dinamica WebSphere eXtreme Scale è diverso dal provider della cache dinamica predefinito in questo caso. Per determinare il valore da impostare per la dimensione della cache nell'istanza della cache dinamica, utilizzare la seguente formula: Utilizzare la seguente formula E:

Formula E

Cs = Ts / Np

Dove:

- Ts = dimensione di destinazione totale per la cache
- Cs = impostazione della dimensione della cache da impostare sull'istanza della cache dinamica
- Np = numero di partizioni. Il valore predefinito è 47.

Impostare la dimensione dell'istanza della cache dinamica su un valore calcolato dalla formula E su ciascun server che condivide l'istanza della cache.

Capitolo 4. Panoramica sui concetti di scalabilità

La scalabilità consente ai dati in una distribuzione di WebSphere eXtreme Scale di essere distribuiti tra una serie di server (contenitori), basati sulla scelta della configurazione.

Scalabilità

WebSphere eXtreme Scale è scalabile tramite l'utilizzo di dati partizionati, e può espandersi fino a migliaia di contenitori se necessario, poiché ogni contenitore è indipendente dagli altri contenitori.

WebSphere eXtreme Scale suddivide i dataset in partizioni distinte che possono essere spostate tra i processi o anche tra le macchine al runtime. È possibile, ad esempio, iniziare con una distribuzione di quattro server e poi espandersi ad una distribuzione con dieci server quando le richieste sulla cache aumentano. Esattamente com'è possibile aggiungere altre macchine fisiche ed unità di elaborazione per la scalabilità verticale, è possibile estendere la capability della scalabilità elastica di eXtreme Scale orizzontalmente con le partizioni. Questa è un'altra principale differenza tra gli INDB (in-memory database) e eXtreme Scale (che è una griglia di dati), in quanto gli IMDB possono modificare la scala solo verticalmente.

Con WebSphere eXtreme Scale, è possibile anche utilizzare una serie di API per ottenere l'accesso transazionale a questi dati partizionati e facoltativamente distribuiti. In termini di prestazione, se scelte effettuate per l'interazione con la cache sono significative quanto le funzioni per gestire la disponibilità della cache.

Nota: La scalabilità non è disponibile quando i contenitori comunicano tra loro. Il protocollo di gestione della disponibilità, o di raggruppamento principale, è un heartbeat $O(N^2)$ ed un algoritmo di manutenzione vista, ma viene mitigato mantenendo il numero di membri del gruppo principale su 20. Esiste solo la replica peer-to-peer tra frammenti.

Client distribuiti

Il protocollo client WebSphere eXtreme Scale supporta numeri molto elevati di client. La strategia di partizionamento offre assistenza presupponendo che non tutti i client sono sempre interessati a tutte le partizioni in quanto le connessioni possono essere distribuite tra più contenitori. I client sono connessi direttamente alle partizioni pertanto la latenza è limitata a un'unica connessione trasferita.

Griglie, partizioni e frammenti

Una griglia eXtreme Scale distribuita è divisa in partizioni. Una partizione contiene una serie secondaria esclusiva di dati. Una partizione è costituita da uno o più frammenti: un frammento primario e frammenti di replica. Non è necessario avere frammenti di replica in una partizione ma i frammenti di replica forniscono alta disponibilità. Sia che la propria distribuzione sia una griglia di dati indipendenti in memoria sia che si tratti di spazio di elaborazione in memoria del database, l'accesso ai dati in eXtreme Scale confida fortemente sul concetto di frammentazione.

I dati per una partizione vengono memorizzati al runtime in una serie di frammenti. Questa serie di frammenti include una frammentazione primaria e, possibilmente, uno o più frammenti di replica. Un frammento rappresenta l'unità più piccola che eXtreme Scale può aggiungere o rimuovere da un Java virtual machine.

Esistono due strategie di posizionamento: FIXED_PARTITIONS (impostazione predefinita) e PER_CONTAINER. L'argomento che segue focalizza l'attenzione sull'utilizzo della strategia FIXED_PARTITIONS.

Numero di frammenti

Se il proprio ambiente includesse dieci partizioni che contenessero un milioni di oggetti senza repliche, allora esisterebbero dieci frammenti ognuno dei quali memorizzerebbe 100.000 oggetti. Se a questo scenario si aggiungesse una replica, esisterebbe un frammento extra in ciascuna partizione. In questo caso, esistono 20 frammenti - dieci frammenti primari e dieci frammenti di replica. Di nuovo, ognuno di questi frammenti memorizza 100.000 oggetti. Ogni partizione consiste di un frammento primario e di uno o più frammenti di replica. È di fondamentale importanza determinare il conteggio ottimale del frammento. Se si effettua la configurazione di un piccolo numero di frammenti, i dati non vengono distribuiti equamente tra i frammenti, provocando errori per mancanza di memoria e problemi di sovraccarico del processore. Quando si esegue lo scale, bisogna avere almeno dieci frammenti per ogni JVM. Quando inizialmente si distribuisce la griglia, si vorrebbe utilizzare un gran numero di partizioni.

Numero di frammenti per JVM

Scenario: piccolo numero di frammenti per ogni JVM

I dati vengono aggiunti e rimossi da un JVM utilizzando unità di frammenti. I frammenti non vengono mai suddivisi in parti. Se esistessero 10 GB di dati ed esistono 20 frammenti per contenere questi dati, ogni frammento contiene in media 500 MB di dati. Se nove Java virtual machine ospitano la griglia, in media ogni JVM ha due frammenti. Poiché 20 non è divisibile equamente per 9, alcuni Java virtual machine avranno tre frammenti, nella seguente distribuzione:

- 7 Java virtual machine con 2 frammenti
- 2 Java virtual machine con 3 frammenti

Poiché ogni frammento contiene 500 MB di dati, la distribuzione dei dati è disuguale. I sette Java virtual machine con due frammenti, ognuno ospita 1 GB di dati. I due Java virtual machine con tre frammenti hanno il 50% in più dei dati, o 1.5 GB, che è un carico di memoria molto più grande. Poiché questi due Java virtual machine stanno ospitando tre frammenti, ricevono anche il 50% in più delle richieste per i loro dati. Come risultato, un numero ridotto di frammenti per ogni JVM comporta uno sbilancio. Per incrementare le prestazioni, aumentare il numero di frammenti per ogni JVM.

Scenario: un incrementato numero di frammenti per JVM

In questo scenario, considerare un numero di frammenti più grande. In questo scenario, ci sono 101 frammenti con 9 Java virtual machine che ospitano 10 GB di dati. In questo caso, ogni frammento contiene 99 MB di dati. Java virtual machine presenta la seguente distribuzione di frammenti:

- 7 Java virtual machine con 11 frammenti

- 2 Java virtual machine con 12 frammenti

I due Java virtual machine con 12 frammenti ora hanno 99 MB in più di dati rispetto agli altri frammenti, che è una differenza del 9%. Questo scenario è molto più equamente distribuito rispetto alla differenza del 50% nello scenario con un numero minore di frammenti. Da una prospettiva di utilizzo del processore, esiste solo il 9% di lavoro in più per i due Java virtual machine con 12 frammenti confrontati con i sette Java virtual machine contrari che hanno 11 frammenti. Aumentando il numero di frammenti in ogni JVM, l'uso dei dati e del processore viene distribuito in modo equo e giusto.

Durante la creazione del proprio sistema, utilizzare 10 frammenti per ogni JVM nello scenario della sua massima dimensione, oppure quando il sistema sta eseguendo il suo numero massimo di Java virtual machine nel proprio orizzonte di pianificazione.

Fattori supplementari di posizionamento

Il numero di partizioni, la strategia di posizionamento ed il numero ed il tipo di repliche vengono impostati nella politica di distribuzione. Il numero di frammenti posizionati dipenderà dalla politica di distribuzione che è stata definita. numInitialContainers, minSyncReplicas, developmentMode, maxSyncReplicas e maxAsyncReplicas influenzeranno il dove e il quando le partizioni possono essere posizionate. Se l'avvio iniziale del proprio server non consente il posizionamento di maxSyncReplicas e di maxAsyncReplicas, se si avviano i server aggiuntivi in un momento successivo si può avere il posizionamento di repliche aggiuntive. Quando si pianifica il numero di frammenti per JVM, il numero massimo di frammenti, comprese le repliche, dipenderà dall'avere sufficienti JVM avviate per supportare il numero massimo di repliche richieste. Una replica non viene mai collocata nello stesso processo del suo elemento primario poiché se il processo viene perso, ciò comporterebbe una perdita sia dell'elemento primario che della replica. Quando developmentMode è impostato su false, l'elemento primario e le repliche non verranno posizionati sulla stessa macchina.

Partizionamento

Utilizzare il partizionamento per memorizzare considerevoli quantitativi di dati nella Java Virtual Machine (JVM). Per partizionare i dati, utilizzare uno schema specifico dell'applicazione per dividere i dati. Utilizzando WebSphere eXtreme Scale, il partizionamento incrementa sia la scalabilità che la disponibilità.

Il partizionamento è il meccanismo che WebSphere eXtreme Scale utilizza per eseguire la scalabilità orizzontale (scale-out) di un'applicazione. Il partizionamento è la separazione dello stato dell'applicazione in parti in cui ciascuna parte contiene una serie di dati dell'istanza completa. Il partizionamento non è come lo striping di RAID (Redundant Array of Independent Disks), che suddivide ogni istanza in tutte le strisce. Ogni partizione ospita i dati completi di singole voci. Il partizionamento è un mezzo molto efficace per la scalabilità, ma non è applicabile a tutte le applicazioni. Le applicazioni che richiedono garanzie di transazione nelle serie di dati di grosse dimensioni non possono modificare la scala e non possono essere suddivise in partizioni in maniera efficace, quindi eXtreme Scale al momento non supporta il commit a due fasi nelle partizioni.

Importante: Selezionare il numero di partizioni con attenzione. Il numero di partizioni definite nella politica di distribuzione influisce direttamente sul numero di contenitori in cui un'applicazione può espandersi tramite la scalabilità. Ogni

partizione è costituita da un frammento primario e dal numero configurato di frammenti di replica. La formula `(Number_Partitions*(1 + Number_Replicas))` è il numero di contenitori che possono essere utilizzati per la scalabilità orizzontale (scale-out) di una singola applicazione.

Utilizzo delle partizioni

Una griglia può avere molte partizioni o A grid can have many partitions, o migliaia di partizioni se richieste. Una griglia può scalare fino al prodotto del numero di partizioni moltiplicato per il numero di frammenti per partizione. Ad esempio, se si hanno 16 partizioni e ciascuna partizione ha un elemento primario ed una replica o due frammenti allora potenzialmente è possibile scalare fino a 32 Java virtual machine. In questo caso, viene definito un frammento per ciascuna JVM. È necessario scegliere un numero ragionevole di partizioni basate sul numero previsto di Java virtual machine che presumibilmente si utilizzeranno. Ciascun frammento incrementa l'utilizzo del processore e della memoria per il sistema. Il sistema è progettato per scalare in aumento (scale out), per gestire questo sovraccarico in linea tra quanti server Java virtual machine sono disponibili.

Le applicazioni non dovrebbero utilizzare migliaia di partizioni se l'applicazione è in esecuzione su una griglia di quattro contenitori.Java virtual machine. L'applicazione dovrebbe essere configurata per avere un numero ragionevole di frammenti per ciascun contenitore JVM. Ad esempio, una configurazione non ragionevole è quella che prevede 2000 partizioni con due frammenti che son in esecuzione su quattro contenitoriJava virtual machine. Questa configurazione risulta in 40000 frammenti che vengono posizionati su quattro contenitori Java virtual machine o 1000 frammenti per contenitore JVM.

Una migliore configurazione dovrebbe essere al di sotto di 10 frammenti per ciascun contenitore JVM previsto. Questa configurazione ancora fornisce la possibilità di consentire una scalabilità adattabile che è 10 volte la configurazione iniziale mentre conserva un ragionevole numero di frammenti per contenitore JVM.

Considerare questo esempio di scalabilità: attualmente si hanno sei computer con due contenitori Java virtual machine per computer. Si prevede di aumentare a 20 il numero dei computer nei prossimi tre anni. Con 20 computer, si hanno 40 contenitori Java virtual machine, e scegliere 60 può essere pessimistico. Si desiderano 4 frammenti per contenitore JVM. Si hanno 60 potenziali contenitori oppure un totale di 240 frammenti. Se si ha un elemento primario ed una replica per partizione, allora si desiderano 120 partizioni. Questo esempio è il risultato di 240 diviso per 12 contenitori Java virtual machine, o 20 frammenti per contenitore JVM per la distribuzione iniziale con la possibilità di scalare oltre i 20 computer successivamente.

ObjectMap e partizionamento

Utilizzando la strategia di posizionamento FIXED_PARTITION predefinita, le mappe vengono suddivise tra le partizioni e le chiavi hash per diverse partizioni. Il client non ha necessità di conoscere a quale partizione appartengono le chiavi. Se una mapSet ha più mappe, dovrebbe essere eseguito il commit delle mappe in transazioni separate.

Entità e partizionamento

Il gestore entità dispone di un'ottimizzazione che guida i client che sono in funzione con le entità su un server. Lo schema di entità sul server per la serie di

mappe può specificare una singola entità di root. Il client deve accedere tutte le entità attraverso l'entità root. Il gestore entità può successivamente trovare le relative entità nella root nella stessa partizione senza richiedere che le relative mappe abbiano una chiave comune. L'entità root stabilisce l'affinità con una singola partizione. Questa partizione viene utilizzata per tutti i fetch all'interno della transazione dopo che l'affinità sia stata stabilita. Questa affinità può risparmiare memoria perché le relative mappe non richiedono una chiave comune. L'entità root deve essere specificata con un'annotazione entità modificata come mostrato nell'esempio riportato di seguito:

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

Utilizzare l'entità per trovare la root dell'oggetto Graph. Il grafico dell'oggetto definisce la relazione tra una o più voci. Ciascuna entità collegata deve risolvere la stessa partizione. Si presume che tutte le entità child siano nella stessa partizione della root. Le entità child nel grafico dell'oggetto sono accessibili solo da un client dalla entità root. Le entità root sono richieste sempre in ambienti partizionati quando si utilizza un client eXtreme Scale per comunicare al server. Solo un tipo di entità root può essere definito per client. Le entità root non sono richieste quando si utilizza ObjectGrid di tipo XTP (Extreme Transaction Processing), poiché tutte le comunicazioni alla partizione sono realizzate attraverso accessi locali diretti e non attraverso il client e meccanismi del server.

Posizionamento e partizioni

Sono disponibili due strategie di posizionamento per WebSphere eXtreme Scale: partizione fissa e per contenitore. La scelta della strategia di posizionamento influenza il modo in cui la configurazione della distribuzione posiziona le partizioni sulla griglia remota.

Posizionamento a partizione fissa

È possibile impostare la strategia di posizionamento nel file XML della politica di distribuzione. La strategia di posizionamento predefinita è quella a partizione fissa, abilitata con l'impostazione `FIXED_PARTITION`. Il numero di frammenti dell'elemento primario che vengono posizionati tra i contenitori disponibili è uguale al numero di partizioni che si sono configurate con l'elemento numberOfPartitions. Se si hanno delle repliche configurate, il numero minimo totale di frammenti posizionati viene definito dalla seguente formula: `((1 frammento dell'elemento primario + il minimo dei frammenti sincroni) moltiplicato le partizioni definite)`. Il numero massimo totale di frammenti posizionati viene definito dalla seguente formula: `((1 frammento dell'elemento primario + il numero massimo di frammenti sincroni + il numero massimo di frammenti asincroni) moltiplicato per le partizioni)`. La distribuzione WebSphere eXtreme Scale suddivide questi frammenti tra i contenitori disponibili. È stato eseguito l'hash delle chiavi di ciascuna mappa nelle partizioni assegnate sulla base delle partizioni totali che sono state definite. Essi eseguono l'hash delle chiavi per la stessa partizione anche se la partizione si sposta a causa di un failover o modifiche al server.

Ad esempio, se il valore numberPartitions è 6 e il valore minSync è 1 per MapSet1, i frammenti totali per quella serie di mappe è 12 perché ciascuna delle 6 partizioni richiede una replica sincrona. Se vengono avviati 3 contenitori WebSphere eXtreme Scale posiziona quattro frammenti per contenitore per MapSet1.

Posizionamento per contenitore

La strategia di posizionamento alternativo è quella per contenitore che viene abilitata con l'impostazione `PER_CONTAINER` per placementStrategy nell'elemento della serie di mappe nel file XML di distribuzione. Utilizzando questa strategia, il numero di frammenti dell'elemento primario posizionati su ciascun nuovo contenitore è uguale al numero di partizioni *P*, che sono state configurate. L'ambiente di distribuzione WebSphere eXtreme Scale posiziona le repliche *P* di ciascuna partizione per ciascun contenitore residuo. L'impostazione numInitialContainers viene ignorata quando si utilizza il posizionamento per contenitore. Le partizioni si allargano al crescere dei contenitori. In questa strategia, le chiavi per le mappe non sono fisse per una determinata partizione. Il client instrada ad una partizione ed utilizza un elemento primario random. Se un client desidera riconnettersi nuovamente alla stessa sessione che ha utilizzato per trovare una chiave, è necessario utilizzare una sessione handle.

Per ulteriori informazioni, consultare l'argomento relativo all'utilizzo di una SessionHandle per l'instradamento nella *Gida alla programmazione*.

Per il failover o l'arresto dei server, l'ambiente WebSphere eXtreme Scale sposta i frammenti dell'elemento primario nella strategia di posizionamento per contenitore se essi contengono ancora dei dati. Se i frammenti sono vuoti, essi vengono cancellati. Nella strategia per contenitore, i frammenti del vecchio elemento primario non vengono conservati perché i frammenti del nuovo elemento primario vengono posizionati per ciascun contenitore.

WebSphere eXtreme Scale consente un posizionamento per-contenitore come alternativa a quella che potrebbe essere definita la strategia di posizionamento "tipica", un approccio a partizione fissa con la chiave di una mappa su cui è eseguito l'hash ad una di queste partizioni. Nel caso per-contenitore (che viene impostato con PER_CONTAINER), la distribuzione posiziona le partizioni sulla serie di server contenitori in linea e automaticamente esegue scale-out o scale-in a seconda che i contenitori vengono aggiunti o rimossi dalla griglia del server. Una griglia con l'approccio a partizione fissa funziona bene per le griglie basate sulle chiavi in cui l'applicazione utilizza un oggetto chiave per individuare i dati nella griglia. Di seguito è trattata l'alternativa.

Esempio di griglia per-contenitore

Le griglie PER_CONTAINER sono diverse. Si specifica l'utilizzo della griglia PER_CONTAINER mediante l'attributo placementPolicy nel file XML di distribuzione. Invece di configurare quante partizioni si desiderano in totale nella griglia, si specifica quante partizioni si desiderano per il contenitore che si avvia.

Ad esempio, se si imposta che il numero di partizioni per contenitore siano 5, successivamente, quando si avvia un contenitore eXtreme Scale crea 5 nuovi elementi primari anonimi della partizione su quel contenitore e crea delle repliche necessarie sugli altri contenitori già distribuiti.

Di seguito è riportata una potenziale sequenza in un ambiente per-contenitore quando aumenta la dimensione della griglia.

1. Avviare il contenitore C0 che ospita 5 elementi primari (P0 - P4).
   - C0 ospita: P0, P1, P2, P3, P4.
2. Avviare il contenitore C1 che ospita altri 5 elementi primari (P5 - P9). Le repliche sono bilanciate tra i contenitori.

- C0 ospita: P0, P1, P2, P3, P4, R5, R6, R7, R8, R9.
- C1 ospita: P5, P6, P7, P8, P9, R0, R1, R2, R3, R4.

3. Avviare il contenitore C2 che ospita altri 5 elementi primari (P10 - P14). Le repliche sono ulteriormente bilanciate.
   - C0 ospita: P0, P1, P2, P3, P4, R7, R8, R9, R10, R11, R12.
   - C1 ospita: P5, P6, P7, P8, P9, R2, R3, R4, R13, R14.
   - C2 ospita: P10, P11, P12, P13, P14, R5, R6, R0, R1.

Il pattern continua quanti più contenitori vengono avviati, crea 5 nuove partizioni dell'elemento primario ogni volta e bilancia nuovamente le repliche tra i contenitori disponibili nella griglia.

Nota: WebSphere eXtreme Scale non sposta gli elementi primari quando si utilizza la strategia PER_CONTAINER, replica solo.

Ricordare che i numeri di partizione sono arbritrari e non hanno nulla a che fare con le chiavi quindi non è possibile utilizzare l'instradamento basato sulla chiave. Se un contenitore si arresta, gli ID di partizione creati per quel contenitore non vengono più utilizzati, quindi esiste un divario negli ID della partizione. Nell'esempio non sarebbero più utilizzate le partizioni P5 - P9 se il contenitore C2 fosse in errore, lasciando solo P0 - P4 e P10 - P14, rendendo impossibile l'hash basato sulla chiave.

Utilizzando il numero 5 o persino più probabilmente 10 per il numero di partizioni, per-contenitore funziona meglio se si considerano le conseguenze di un errore del contenitore. Per distribuire uniformemente il carico dei frammenti ospitati tra la griglia è necessario disporre di più di una sola partizione per ciascun contenitore. Se si dispone di una singola partizione per contenitore, quando un contenitore è in errore, solo un contenitore (l'unico che ospita il frammento di replica corrispondente) deve sopportare il pieno carico dell'elemento primario perso. In questo caso, il carico è immediatamente raddoppiato per il contenitore. Tuttavia, se si dispone di 5 partizioni per contenitore i 5 contenitori si addossano il carico del contenitore perso riducendo l'impatto su ognuno del 80 per cento. Utilizzando più partizioni per contenitore generalmente si riduce sostanzialmente l'impatto potenziale su ciascun contenitore. Più precisamente, considerare un caso in cui un contenitore ha un picco di carico inaspettato, il carico della replica di quel contenitore viene distribuito su 5 contenitori piuttosto che su uno solo.

Utilizzo della politica per-contenitore

Diversi scenari rendono la strategia per-contenitore una configurazione ideale, è il caso della replica della sessione HTTP o lo stato della sessione dell'applicazione. In tal caso, un router HTTP assegna una sessione ad un contenitore servlet. Il contenitore servlet necessita di creare una sessione HTTP e sceglie uno dei 5 elementi primari della partizione locale per la sessione. L'"ID" della partizione scelta viene memorizzata in un cookie. Il contenitore servlet dispone ora dell'accesso locale allo stato della sessione che significa accesso ai dati con latenza zero per questa richiesta finché si conserva l'affinità di sessione. Un eXtreme Scale replica qualsiasi modifica nella partizione.

In pratica, tener presente le ripercussioni di un caso in cui si hanno più partizioni per contenitore (dico di nuovo 5). Naturalmente, con ciascun nuovo contenitore avviato, si hanno altri 5 elementi primari della partizione e altre 5 repliche. Nel corso del tempo, si dovrebbero creare altre partizioni ed esse non dovrebbero essere spostate o distrutte. Ma questo non è il modo in cui i contenitori

effettivamente si comporterebbero. Quando un contenitore si avvia, ospita 5 frammenti di elementi primari che possono essere denominati elementi primari "home", esistenti sui rispettivi contenitori che li crearono. Se il contenitore è in errore, le repliche divengono elementi primari e eXtreme Scale crea altre 5 repliche per conservare l'alta disponibilità (a meno che non si sia disabilita la correzione automatica). I nuovi elementi primari sono in un differente contenitore rispetto a quello che li creò che può essere denominato elementi primari "esterni". L'applicazione non dovrebbe mai posizionare il nuovo stato o le sessioni in un elemento primario esterno. Eventualmente, l'elemento primario esterno non ha voci e eXtreme Scale automaticamente lo elimina insieme alle repliche ad esso associate. Lo scopo degli elementi primari esterni è consentire alle sessioni esistenti di essere ancora disponibili (ma non le nuove sessioni).

Un client può ancora interagire con una griglia che non fa affidamento sulle chiavi. Il client inizia solo una transazione e memorizza i dati nella griglia indipendente da qualsiasi chiave. Esso richiede la sessione per un oggetto SessionHandle, un handle serializzabile che consenta al client di interagire con la stessa partizione quando necessario. Per ulteriori informazioni, consultare l'argomento relativo all'utilizzo di un SessionHandle per instradare nella *Guida alla Programmazione*. WebSphere eXtreme Scale sceglie una partizione per il client dall'elenco degli elementi primari della partizione home. Essa non restituisce una partizione dell'elemento primario esterno. La SessionHandle può essere serializzata in un cookie HTTP, ad esempio, e successivamente riconvertire il cookie nella SessionHandle. Successivamente le API WebSphere eXtreme Scale possono ottenere una sessione associata di nuovo alla stessa partizione utilizzando SessionHandle.

Nota: Non è possibile utilizzare agent per interagire con una griglia PER_CONTAINER.

Vantaggi

La precedente descrizione è diversa da una normale FIXED_PARTITION o griglia hash perché il client per contenitore memorizza i dati in una posizione nella griglia, ottiene un handle per la griglia ed utilizza l'handle per accederla di nuovo. Non esiste una chiave fornita dall'applicazione come invece esiste nel caso della partizione fissa.

La distribuzione non crea una nuova partizione per ciascuna sessione. Perciò in una distribuzione per contenitore, le chiavi utilizzate per memorizzare i dati nella partizione devono essere univoche all'interno di quella partizione. Ad esempio, può accadere che il client generi un SessionID univoco e successivamente lo utilizzi come chiave per trovare le informazioni nelle mappe in quella partizione. Più sessioni client che interagiscono con la stessa partizione quindi l'applicazione necessita di utilizzare chiavi univoche per memorizzare i dati della sessione in ciascuna partizione data.

I precedenti esempi utilizzavano 5 partizioni, ma può essere utilizzato il parametro numberOfPartitions nel file XML di Objectgrid per specificare le partizioni richieste. Invece che per griglia, l'impostazione è per contenitore. (Il numero di repliche viene specificato nello stesso modo in cui vengono specificate utilizzando la politica della partizione fissa)

La politica per contenitore può essere utilizzata anche con più zone. Se possibile, eXtreme Scale restituisce una SessionHandle ad una partizione il cui elemento primario sia ubicato nella stessa zona del client. Il client può specificare la zona

come un parametro per il contenitore o utilizzando un'API. L'ID di zona del client può essere impostato utilizzando `serverproperties` o `clientproperties`.

La strategia PER_CONTAINER per una griglia soddisfa lo stato del tipo conversazionale di archiviazione delle applicazioni piuttosto che i dati orientati al database. La chiave per accedere i dati dovrebbe essere un ID conversazione e non è correlato ad uno specifico record di database. Fornisce prestazioni più elevate (perché gli elementi primari della partizione possono essere collocati con i servlet ad esempio) e fornisce una configurazione più semplice (senza dover calcolare le partizioni e i contenitori).

Transazioni su partizione singola e sulle partizioni della griglia

La distinzione principale tra WebSphere eXtreme Scale e le soluzioni di memorizzazione dei dati tradizionali come i database relazionali o i database in memoria è rappresentata dall'utilizzo delle partizioni, che consentono alla cache di scalare in modo lineare. I tipi importanti di transazione da considerare sono le transazioni su partizione singola e su tutte le partizioni (sulla griglia).

In generale, le interazioni con la cache possono essere suddivise in transazioni su partizione singola o transazioni sulla griglia, come riportato di seguito.

Transazioni su partizione singola

Le transazioni su partizione singola rappresentano il metodo preferito per l'interazione con le cache ospitate da WebSphere eXtreme Scale. Quando una transazione è limitata ad una partizione singola, per impostazione predefinita è limitata ad una singola Java virtual machine e quindi ad un singolo computer server. Un server può completare un numero *M* di tali transazioni al secondo e, se si dispone di *N* computer, è possibile completare `M*N` transazioni al secondo. Se le proprie attività vengono incrementate ed è necessario raddoppiare il numero di transazioni al secondo, è possibile raddoppiare *N* acquistando altri computer. Quindi, è possibile soddisfare le richieste di capacità senza modificare l'applicazione, aggiornare l'hardware o portare l'applicazione fuori linea.

Oltre a consentire alla cache di scalare in modo così significativo, le transazioni su partizione singola incrementano al massimo la disponibilità della cache. Ciascuna transazione dipende solo da un computer. Anche in caso di malfunzionamento di uno degli altri `(N-1)` computer, il successo o il tempo di risposta della transazione non vengono modificati. Quindi, se vengono utilizzati 100 computer e si verifica un malfunzionamento di uno di essi, viene eseguito il rollback solo dell'1 percento delle transazioni in esecuzione al momento del malfunzionamento del server. In seguito al malfunzionamento del server, WebSphere eXtreme Scale assegna nuovamente le partizioni ospitate dal server malfunzionante agli altri 99 computer. Durante tale periodo, prima che l'operazione venga completata, gli altri 99 computer possono ancora completare le transazioni. Vengono bloccate solo le transazioni che coinvolgono le partizioni in fase di riassegnazione. Una volta completato il processo di failover, la cache può continuare a funzionare, completamente operativa, al 99 percento della propria capacità di trasmissione originale. Dopo che il server malfunzionante viene sostituito e restituito alla griglia, la cache ritorna al 100 percento della capacità di trasmissione.

Transazioni sulla griglia

In termini di prestazioni, disponibilità e scalabilità, le transazioni sulla griglia sono l'opposto delle transazioni su partizione singola. Le transazioni sulla griglia

accedono a tutte le partizioni e quindi a tutti i computer nella configurazione. A ciascun computer nella griglia viene richiesto di ricercare alcuni dati e di restituire i risultati. La transazione non può essere completata fino a quando non hanno risposto tutti i computer e quindi il livello di prestazione dell'intera griglia è limitato dal computer più lento. L'aggiunta di altri computer non rende più veloce il computer più lento e quindi non migliora il livello di prestazione della cache.

Le transazioni sulla griglia hanno un effetto simile sulla disponibilità. Facendo riferimento all'esempio precedente, se vengono utilizzati 100 server e si verifica il malfunzionamento di un server, viene eseguito il rollback del 100 percento delle transazioni in esecuzione al momento del malfunzionamento del server. Dopo il malfunzionamento del server, WebSphere eXtreme Scale inizia a riassegnare le partizioni ospitate da tale server agli altri 99 computer. Durante questo periodo di tempo, prima che venga completato il processo di failover, la griglia non è in grado di elaborare alcuna di tali transazioni. Una volta completato il processo di failover, la cache può continuare a funzionare, ma con una capacità ridotta. Se ciascun computer nella griglia gestiva 10 partizioni, 10 dei rimanenti 99 computer ricevono almeno una partizione supplementare in seguito al processo di failover. L'aggiunta di una partizione supplementare incrementa il carico di lavoro di tale computer di almeno il 10 percento. Poiché il livello di prestazione della griglia è limitato alla velocità di trasmissione del computer più lento in una transazione sulla griglia, il livello di prestazione viene ridotto in media del 10 percento.

È preferibile utilizzare le transazioni su partizione singola invece delle transazioni su griglia per eseguire lo scale out con una cache di oggetti distribuita, ad alta disponibilità come WebSphere eXtreme Scale. L'ottimizzazione delle prestazioni di questi tipi di sistemi richiede l'utilizzo di tecniche differenti dalle metodologie relazionali tradizionali, ma è possibile convertire le transazioni su griglia in transazioni su partizione singola scalabili.

Migliori pratiche per la creazione di modelli di dati scalabili

Le migliori pratiche per la creazione di applicazioni scalabili con prodotti come WebSphere eXtreme Scale includono due categorie: principi fondamentali e suggerimenti per l'implementazione. I principi fondamentali sono le idee principali da catturare nella progettazione dei dati. Un'applicazione che non osserva tali principi non sarà in grado di scalare correttamente, anche per le proprie transazioni principali. I suggerimenti per l'implementazione vengono applicati per transazioni problematiche in un'applicazione altrimenti ben progettata che osserva i principi generali per i modelli di dati scalabili.

Principi fondamentali

Alcuni degli importanti mezzi di ottimizzazione della scalabilità sono concetti di base o principi da tenere in considerazione.

Duplicazione invece della normalizzazione

Il concetto principale da ricordare quando si utilizzano prodotti come WebSphere eXtreme Scale è che tali prodotti sono progettati per distribuire i dati su un numero elevato di computer. Se l'obiettivo è il completamento della maggior parte o di tutte le transazioni su una singola partizione, il progetto del modello di dati deve garantire che tutti i dati che potrebbero essere richiesti dalla transazione si trovino nella partizione. Nella maggior parte dei casi, ciò è possibile solo duplicando i dati.

Ad esempio, considerare un'applicazione come una bacheca messaggi. Due transazioni molto importanti per una bacheca messaggi mostrano tutti i messaggi inviati da un determinato utente e tutti i messaggi relativi ad un determinato argomento. Innanzitutto, considerare il modo in cui tali transazioni funzionano con un modello di dati normalizzato che contiene un record utente, un record dell'argomento ed un record del messaggio che contiene il testo reale. Se i messaggi sono partizionati con i record utente, la visualizzazione degli argomenti diventa una transazione su griglia e viceversa. Gli argomenti e gli utenti non possono essere partizionati insieme perché hanno una relazione molti-a-molti.

Il modo migliore per consentire a questa bacheca messaggi di scalare è quello di duplicare i messaggi, memorizzando una copia con il record dell'argomento ed una copia con il record utente. Quindi, la visualizzazione dei messaggi da un utente è una transazione su partizione singola, la visualizzazione dei messaggi su un argomento è una transazione su partizione singola e l'aggiornamento o eliminazione di un messaggio è una transazione su due partizioni. Tutte e tre tali transazioni scaleranno linearmente con l'incremento del numero di computer nella griglia.

Scalabilità piuttosto che risorse

L'ostacolo maggiore da superare quando si considerano modelli di dati denormalizzati è l'impatto di tali modelli sulle risorse. La conservazione di due, tre o più copie di alcuni dati può dare l'impressione di utilizzare un numero troppo elevato di risorse per essere pratica. In questo scenario, ricordare quanto riportato di seguito: ogni anno, le risorse hardware diventano meno costose. Secondo, e più importante, WebSphere eXtreme Scale elimina la maggior parte dei costi nascosti associati alla distribuzione di ulteriori risorse.

Misurare le risorse in termini di costi piuttosto che in termini di hardware, come megabyte e processori. Gli archivi di dati che utilizzano dati relazionali normalizzati generalmente devono essere ubicati sullo stesso computer. Tale collocazione obbligatoria indica che è necessario acquistare un singolo computer enterprise di capacità maggiori piuttosto che diversi computer di capacità minori. Con l'hardware enterprise, non è raro il caso in cui un computer in grado di completare un milione di transazioni al secondo costi molto più della soma dei costi di 10 computer in grado, ciascuno, di eseguire 100000 transazioni al secondo.

Inoltre, esiste un costo aziendale relativo all'aggiunta di risorse. Un business crescente alla fine esaurisce le proprie capacità. Quando si esaurisce la capacità, è necessario passare ad un computer più potente e veloce oppure creare un secondo ambiente di produzione. In entrambi i casi, derivano ulteriori costi a causa di affari non realizzati o della gestione di una capacità quasi doppia rispetto a quella necessaria durante il periodo di transizione.

Con WebSphere eXtreme Scale, non è necessario chiudere l'applicazione per aggiungere capacità. Se i propri progetti per l'anno successivo richiedono una capacità superiore del 10 percento, incrementare del 10 percento il numero di computer nella griglia. È possibile incrementare tale percentuale senza rendere indisponibile l'applicazione e senza acquistare capacità supplementare.

Evitare le trasformazioni dei dati

Quando si utilizza WebSphere eXtreme Scale, i dati devono essere memorizzati in un formato direttamente utilizzabile dalla logica di business. L'utilizzo di un formato più primitivo per i dati rappresenta un costo. È necessario eseguire la trasformazione quando vengono eseguite la scrittura e la lettura dei dati. Con i database relazionali tale trasformazione è necessaria, in quanto i dati vengono resi persistenti su disco abbastanza frequentemente, ma con WebSphere eXtreme Scale, non è necessario eseguire tali trasformazioni. Per la maggior parte, i dati sono memorizzati nella memoria e possono essere memorizzati nel formato esatto richiesto dall'applicazione.

L'osservazione di questa semplice regola consente di denormalizzare i dati in base al primo principio. Il tipo più comune di trasformazione per i dati di business è rappresentato dalle operazioni JOIN, necessarie per convertire dati normalizzati in una serie di risultati che soddisfi le necessità dell'applicazione. La memorizzazione dei dati nel formato corretto evita, in modo implicito, l'esecuzione di tali operazioni JOIN e produce un modello di dati denormalizzato.

Eliminare le query illimitate

Anche se i dati sono strutturati correttamente, le query illimitate non vengono scalate correttamente. Ad esempio, non utilizzare una transazione che richiede un elenco di tutti gli elementi ordinati in base al valore. Questa transazione potrebbe funzionare inizialmente, quando il numero totale di elementi è uguale a 1000; quando, però, il numero totale di elementi raggiunge 10 milioni, la transazione restituisce tutti gli elementi. Se viene eseguita questa transazione, i due risultati più probabili sono la scadenza della transazione o la visualizzazione di un errore di memoria esaurita del client.

L'opzione migliore è quella di modificare la logica di business in modo che possano essere restituiti solo i primi 10 o 20 elementi. Tale modifica della logica rende gestibile la dimensione della transazione, indipendentemente dal numero di elementi presenti nella cache.

Definizione dello schema

Il vantaggio principale della normalizzazione dei dati consiste nel fatto che il sistema del database può occuparsi della congruenza dei dati in modo trasparente. Quando i dati vengono denormalizzati per la scalabilità, tale gestione della congruenza dei dati automatica non esiste più. Per garantire la congruenza dei dati, è necessario implementare un modello di dati che possa essere utilizzato nel livello dell'applicazione oppure come plug-in sulla griglia distribuita.

Considerare l'esempio relativo alla bacheca messaggi. Se una transazione rimuove un messaggio da un argomento, è necessario rimuovere il messaggio duplicato sul record utente. Senza un modello di dati, è possibile che uno sviluppatore scriva il codice dell'applicazione per rimuovere il messaggio dall'argomento, dimenticando di rimuovere il messaggio dal record utente. Tuttavia, se lo sviluppatore utilizzasse un modello di dati invece di interagire direttamente con la cache, il metodo removePost sul modello di dati potrebbe estrarre l'ID utente dal messaggio, ricercare il record utente e rimuovere il messaggio duplicato in modo trasparente.

In alternativa, è possibile implementare un listener che viene eseguito sulla partizione reale che rileva la modifica all'argomento e modifica automaticamente il record utente. Un listener può essere utile perché la

modifica al record utente potrebbe essere eseguita in locale se la partizione dispone del record utente oppure, anche se il record utente si trova su una partizione differente, la transazione viene eseguita tra i server invece che tra client e server. È probabile che la connessione di rete tra i server sia più rapida rispetto alla connessione di rete tra il client ed il server.

Evitare i conflitti

Evitare scenari in cui è presente un contatore globale. La griglia non scala se un singolo record viene utilizzato un numero di volte sproporzionato in confronto agli altri record. Le prestazioni della griglia saranno limitate dalle prestazioni del computer che contiene il record indicato.

In queste situazioni, provare a suddividere il record, in modo che sia gestito per partizione. Ad esempio, considerare una transazione che restituisce il numero totale di voci nella cache distribuita. Invece di impostare il sistema in modo che tutte le operazioni di inserimento e rimozione accedono ad un singolo record, utilizzare, su ciascuna partizione, un listener che tiene traccia delle operazioni di inserimento e rimozione. Grazie alla traccia del listener, le operazioni di inserimento e rimozione possono diventare operazioni su partizione singola.

La lettura del contatore diventa un'operazione sulla griglia, ma per la maggior parte è già inefficiente come operazione sulla griglia perché le proprie prestazioni sono collegate a quelle del computer che contiene il record.

Suggerimenti per l'implementazione

Per ottenere la massima scalabilità, è possibile considerare i suggerimenti riportati di seguito.

Utilizzare gli indici di ricerca inversi

Considerare un modello di dati denormalizzato in modo appropriato in cui i record del cliente sono partizionati in base al numero ID del cliente. Tale metodo di partizionamento rappresenta la scelta logica in quanto quasi tutte le operazioni di business eseguite con il record del cliente utilizzano il numero ID del cliente. Tuttavia, un'importante transazione che non utilizza il numero ID del cliente è la transazione di login. Per il login, è più comune utilizzare nomi utente o indirizzi e-mail invece dei numeri ID del cliente.

L'approccio semplice allo scenario di login è quello di utilizzare una transazione sulla griglia per individuare il record del cliente. Come illustrato in precedenza, tale approccio non esegue la scalabilità.

L'opzione successiva potrebbe essere quella di eseguire il partizionamento sul nome utente o l'e-mail. Questa opzione non è pratica, in quanto tutte le operazioni basate sull'ID cliente diventano transazioni sulla griglia. Inoltre, i clienti del sito potrebbero desiderare di modificare il proprio nome utente o indirizzo e-mail. I prodotti come WebSphere eXtreme Scale richiedono che il valore utilizzato per partizionare i dati resti costante.

La soluzione corretta consiste nell'utilizzo di un indice di ricerca inverso. Con WebSphere eXtreme Scale, è possibile creare una cache nella stessa griglia distribuita della cache che contiene tutti i record utente. Questa cache è altamente disponibile, partizionata e scalabile. Tale cache può essere utilizzata per associare un nome utente o un indirizzo e-mail ad un ID cliente. Questa cache converte l'operazione di login in un'operazione su

due partizioni invece di un'operazione sulla griglia. Tale scenario non è pratico come una transazione su partizione singola, ma il livello di prestazione viene scalato linearmente con l'aumento del numero di computer.

Calcolo al momento della scrittura

I valori calcolati comunemente, come le medie o i totali, possono essere costosi da creare perché tali operazioni generalmente richiedono la lettura di un numero di voci elevato. Poiché nella maggior parte delle applicazioni le letture sono più comuni delle scritture, è preferibile calcolare tali valori al momento della scrittura e memorizzare il risultato nella cache. Questa operazione rende le operazioni di scrittura più rapide e scalabili.

Campi facoltativi

Considerare un record utente che contenga un numero di telefono dell'ufficio, di casa e mobile. Per ciascun utente, potrebbero essere definiti tutti, nessuno o qualsiasi combinazione di tali numeri. Se i dati fossero normalizzati, esisterebbero una tabella utente ed una tabella dei numeri telefonici. I numeri telefonici di un determinato utente potrebbero essere individuati utilizzando un'operazione JOIN tra le due tabelle.

La denormalizzazione di tale record non richiede la duplicazione dei dati, perché la maggior parte degli utenti non condivide i numeri di telefono. Al contrario, deve essere possibile che alcuni alloggiamenti siano vuoti nel record utente. Invece di utilizzare una tabella dei numeri di telefono, aggiungere tre attributi a ciascun record utente, uno per ciascun tipo di numero di telefono. Tale aggiunta di attributi elimina l'operazione JOIN e converte l'operazione di ricerca di un numero di telefono per un utente in un'operazione su partizione singola.

Posizionamento di relazioni molti-a-molti

Considerare un'applicazione che tenga traccia dei prodotti e dei negozi in cui i prodotti vengono venduti. Un singolo prodotto viene venduto in molti negozi ed un singolo negozio vende molti prodotti. Si supponga che tale applicazione tracci 50 grandi venditori. Ciascun prodotto viene venduto in un massimo di 50 negozi, ciascuno dei quali vende migliaia di prodotti.

Conservare un elenco dei negozi all'interno dell'entità del prodotto (disposizione A) invece di conservare un elenco dei prodotti all'interno di ciascuna entità del negozio (disposizione B). Analizzando alcune delle transazioni che questa applicazione deve eseguire, è possibile comprendere per quale motivo la disposizione A è più scalabile.

Per prima cosa, si analizzino gli aggiornamenti. Con la disposizione A, la rimozione di un prodotto dall'inventario di un negozio blocca l'entità del prodotto. Se la griglia contiene 10000 prodotti, per eseguire l'aggiornamento è necessario bloccare solo 1/10000 della griglia. Con la disposizione B, la griglia contiene solo 50 negozi, per cui per completare l'aggiornamento è necessario bloccare solo 1/50 della griglia. Quindi, anche se entrambe queste operazioni possono essere considerate operazioni su partizione singola, la disposizione A esegue lo scale out in modo più efficiente.

Ora, considerando le letture con la disposizione A, la ricerca dei negozi in cui viene venduto un prodotto è una transazione su partizione singola che scala ed è rapida perché la transazione trasmette solo una piccola quantità di dati. Con la disposizione B, la transazione diventa una transazione sulla

griglia, perché è necessario accedere a ciascuna entità del negozio per verificare se il prodotto viene venduto in tale negozio, rivelando un enorme vantaggio per le prestazioni per la disposizione A.

Scalabilità con dati normalizzati

Un utilizzo valido delle transazioni sulla griglia è quello di scalare l'elaborazione dei dati. Se una griglia dispone di 5 computer e viene distribuita una transazione sulla griglia che esamina circa 100000 record su ciascun computer, la transazione esamina 500000 record. Se il computer più lento nella griglia è in grado di eseguire 10 di tali transazioni al secondo, la griglia è in grado di esaminare 5000000 record al secondo. Se i dati nella griglia vengono raddoppiati, ciascun computer deve esaminare 200000 record e ciascuna transazione esamina 1000000 record. Tale incremento di dati riduce il livello di prestazione del computer più lento a 5 transazioni al secondo, riducendo di conseguenza livello di prestazione della griglia a 5 transazioni al secondo. La griglia esamina 5000000 di record al secondo.

In questo scenario, raddoppiando il numero di computer, ogni computer può tornare al proprio precedente carico di lavoro esaminando 100000 record, consentendo al computer più lento di elaborare 10 di tali transazioni al secondo. Il livello di prestazione della griglia resta invariato a 10 richieste al secondo, ma ora ciascuna transazione elabora 1000000 record, per cui la griglia ha raddoppiato la propria capacità in termini di elaborazione dei record, portandola a 10000000 al secondo.

Per le applicazioni come i motori di ricerca, che devono eseguire operazioni di scalabilità in termini di elaborazione dei dati per adattarsi all'incremento di Internet ed in termini di livello di prestazione per adattarsi al numero sempre crescente di utenti, è necessario creare più griglie, con una condivisione delle richieste sulle griglie. Se è necessario aumentare il livello di prestazione, aggiungere altri computer ed aggiungere un'altra griglia alle richieste di servizio. Se è necessario aumentare l'elaborazione dei dati, aggiungere altri computer e tenere costante il numero di griglie.

Scalabilità nelle unità o nei pod

Questo argomento tratta la scalabilità ma non nel modo tradizionale, ad esempio come eseguire la scalabilità orizzontale (scale-out) della griglia con un determinato numero X di JVM. Invece, qui la prospettiva tratta la scalabilità in termini di operazioni, pianificazione e gestione dei rischi. Anche se questi fattori sono in genere più importanti delle considerazioni tradizionali sulla scalabilità del prodotto, sfortunatamente vengono spesso ignorati. La creazione di sistemi altamente disponibili richiede entrambi i tipi di considerazioni sulla scalabilità per distribuire eXtreme Scale in un processo di distribuzione affidabile.

Distribuzione di una singola griglia di grandi dimensioni

I test hanno verificato che eXtreme Scale può disporre di una scalabilità orizzontale (scale out) che arriva fino a 1000 JVM. Questo genere di test incoraggia la creazione di applicazioni per distribuire singole griglie con un numero elevato di unità. Sebbene sia possibile effettuare questa operazione, non è consigliato, per diversi motivi riportati di seguito.

1. **Motivi di budget:** l'ambiente non può realisticamente testare una griglia di 1000 server. Tuttavia, può testare una griglia molto più piccola che considera le problematiche di budget, quindi non è necessario comprare il doppio dell'hardware, specialmente per un numero così elevato di server.

2. **Versioni diverse delle applicazioni:** la necessità di un numero elevato di unità per ogni thread di test non è pratica. Il rischio è che non vengano testati gli stessi fattori come succederebbe invece in un ambiente di test.
3. **Perdita di dati:** l'esecuzione di un database su una singola unità disco rigido non è affidabile. Qualsiasi problema con l'unità disco rigido comporta la perdita di dati. L'esecuzione di un'applicazione in crescita su una singola griglia è simile. È probabile che siano presenti bug nell'ambiente e nelle applicazioni. Pertanto, posizionare tutti i dati su un unico grande sistema, spesso porta alla perdita di grosse quantità di dati.

Suddivisione della griglia

La suddivisione della griglia dell'applicazione in pod (unità) è più affidabile. Un pod è un gruppo di server che eseguono uno stack dell'applicazione omogeneo. I pod possono essere di qualsiasi dimensione ma idealmente devono essere costituiti da circa 20 unità. Piuttosto che avere 500 unità in una singola griglia, è possibile avere 25 pod da 20 unità. Deve essere in esecuzione un'unica versione di uno stack dell'applicazione su un determinato pod, ma pod diversi possono avere versioni proprie di uno stack dell'applicazione.

In generale, uno stack dell'applicazione prende in considerazione i livelli dei seguenti componenti.

- Sistema operativo
- Hardware
- JVM
- Versione di eXtreme Scale
- Applicazione
- Altri componenti necessari

Un pod è un'unità di distribuzione, con dimensioni appropriate, per l'esecuzione di test. Invece di avere centinaia di server per l'esecuzione di test, è più pratico avere 20 server. In questo caso, si testa comunque la stessa configurazione che si avrebbe in produzione. La produzione utilizza le griglie con una dimensione massima di 20 server, che costituiscono un pod. È possibile sottoporre un singolo pod ad un test di stress e determinare la sua capacità, il numero di utenti, la quantità di dati e il livello di prestazione delle transazioni. Questo rende la pianificazione più semplice e segue lo standard della scalabilità prevedibile ad un costo prevedibile.

Configurazione di un ambiente basato su pod

In casi diversi, il pod non deve necessariamente avere 20 server. Lo scopo della dimensione del pod è un test pratico. La dimensione di un pod deve essere sufficientemente piccola in modo tale che se il pod riscontra problemi nella produzione, la frazione delle transazioni interessate sia tollerabile.

Idealmente, qualsiasi bug avrà un impatto solo su un singolo pod. Nell'esempio precedente, un bug avrebbe un impatto solo sul quattro percento delle transazioni dell'applicazione, piuttosto che il 100 percento. Inoltre, gli aggiornamenti sono più facili perché possono essere eseguiti un pod alla volta. Questo semplifica lo scenario in modo che se un aggiornamento ad un pod crea problemi, l'utente può riportare quel pod al livello precedente. Gli aggiornamenti comprendono tutte le modifiche all'applicazione, lo stack dell'applicazione o gli aggiornamenti di sistema. Per quanto possibile, gli aggiornamenti devono modificare un unico elemento dello alla volta per rendere più precisa la diagnosi del problema.

Per implementare un ambiente con i pod, è necessario un livello di instradamento al di sopra dei pod che sia compatibile in avanti e all'indietro se i pod ricevono aggiornamenti del software. Inoltre, è necessario creare una directory che contenga le informazioni relative a quali dati sono contenuti in ciascun pod. (È possibile utilizzare un'altra griglia eXtreme Scale per questo, con un database alle spalle, preferibilmente utilizzando lo scenario write-behind). Questo porta ad una soluzione a due livelli. Il livello 1 è la directory e viene utilizzata per individuare quale pod gestisce una transazione specifica. Il livello 2 è composto dai pod stessi. Quando il livello 1 identifica un pod, la configurazione instrada ogni transazione al server corretto nel pod, che in genere è il server che contiene la partizione dei dati utilizzati dalla transazione. Facoltativamente, è possibile anche utilizzare una cache locale sul livello 1 per ridurre l'impatto associato alla ricerca del pod corretto.

L'utilizzo dei pod è leggermente più complesso che avere una singola griglia, ma i miglioramenti nell'operatività, nell'esecuzione dei test e nell'affidabilità lo rendono una parte cruciale del test di scalabilità.

Capitolo 5. Panoramica sulla disponibilità

Alta disponibilità

Con l'alta disponibilità (HA), WebSphere eXtreme Scale fornisce ridondanza di dati affidabili e rilevamento dei malfunzionamenti.

WebSphere eXtreme Scale organizza da sé griglie di Java virtual machine in una flessibile struttura ad albero federata, con il servizio catalogo nella root ed i gruppi principali con contenitori negli elementi foglia della struttura ad albero. Per ulteriori informazioni, consultare “Architettura di memorizzazione nella cache: mappe, contenitori, client e cataloghi” a pagina 9.

Ogni gruppo principale viene creato automaticamente dal servizio catalogo in gruppi di circa 20 server. I membri del gruppo principale forniscono un controllo dello stato di salute per altri membri del gruppo. Inoltre, ogni gruppo principale elegge un membro quale leader per comunicare le informazioni del gruppo al servizio catalogo. Limitando la dimensione del gruppo principale si consente un buon controllo dello stato di salute ed un ambiente altamente scalabile.

Nota: In un ambiente WebSphere Application Server, in cui la dimensione del gruppo principale può essere alterata, eXtreme Scale non supporta più di 50 membri per gruppo principale.

Errori

Ci sono diversi motivi per cui un processo può non riuscire. Il processo potrebbe non riuscire a causa del raggiungimento del limite di alcune risorse, quali la dimensione massima heap o perché alcune logiche di controllo del processo hanno fatto terminare un processo. Il sistema operativo potrebbe dare un errore causando la perdita di tutti i processi in esecuzione sul sistema. L'hardware potrebbe presentare un errore, sebbene meno frequentemente, come la scheda NIC (network interface card), causando lo scollegamento dalla rete del sistema operativo. Si possono verificare molti altri punti di errore, che comportano la non disponibilità del processo. In questo contesto, tutti questi errori possono essere ricondotti a uno o due tipi di categorie: errore del processo e perdita della connettività.

Errore del processo

WebSphere eXtreme Scale reagisce agli errori del processo molto velocemente. Quando si verifica un errore nel processo, il sistema operativo è responsabile della ripulitura di eventuali risorse che il processo stava utilizzando. Questa ripulitura include l'allocazione della porta e la connettività. Quando si verifica un errore nel processo, sulle connessioni che erano utilizzate da quel processo viene inviato un segnale per chiudere ogni connessione. Con tali segnali, un errore del processo può essere rilevato istantaneamente da qualunque altro processo collegato al processo non riuscito.

Perdita della connettività

La perdita di connettività si verifica quando il sistema operativo si scollega. Come risultato, il sistema operativo non può inviare segnali ad altri processi. La perdita della connettività può verificarsi per vari motivi ma possono essere divisi in due categorie: errore host e isolamento.

Errore host

Se la macchina viene staccata dalla presa elettrica, la connessione si perde istantaneamente.

Isolamento

Questo scenario rappresenta la condizione di errore più complicata da gestire per il software poiché si presume che il processo non sia disponibile mentre invece lo è. Essenzialmente, un server o un altro processo appare al sistema come non funzionante mentre in realtà sta funzionando correttamente.

Errore del contenitore eXtreme Scale

Gli errori del contenitore vengono scoperti generalmente dai contenitori peer attraverso il meccanismo del gruppo principale. Quando un contenitore o una serie di contenitori va in errore, il servizio catalogo esegue la migrazione dei frammenti ospitati su quel contenitore o contenitori. Il servizio catalogo cerca prima una replica sincrona prima di eseguire la migrazione ad una replica asincrona. Dopo la migrazione dei frammenti primari nei nuovi contenitori dell'host, il servizio catalogo cerca nuovi contenitori dell'host per le repliche che ora stanno mancando.

Nota: Isolamento del contenitore - il servizio catalogo esegue la migrazione dei frammenti fuori dai contenitori quando si scopre che il contenitore non è disponibile. Se questi contenitori diventano poi disponibili, il servizio catalogo considera i contenitori collocabili proprio come nel normale flusso di avvio.

Latenza rilevamento failover del contenitore

Gli errori possono essere categorizzati in soft e hard. Gli errori soft sono di solito causati da un malfunzionamento del processo. Questi errori vengono rilevati dal sistema operativo, che può recuperare le risorse utilizzate, come i socket di rete, molto velocemente. Il tipico rilevamento errori per quanto riguarda errori soft è inferiore a un secondo. Il rilevamento dei malfunzionamenti hard può impiegare fino a 200 secondi utilizzando l'ottimizzazione heartbeat. Errori di questo tipo includono: rotture fisiche delle macchine, scollegamento dei cavi di rete o errori del sistema operativo. Così, eXtreme Scale deve fare affidamento sull'heartbeat per poter rilevare errori hard che possono essere configurati. Consultare “Tipi di rilevamento failover” a pagina 115 per i dettagli su come ridurre il tempo per rilevare errori di tipo hard.

Errore del servizio catalogo

Poiché la griglia del servizio catalogo è una griglia eXtreme Scale, questa utilizza anche il meccanismo del raggruppamento principale nello stesso modo del processo di errore del contenitore. La differenza principale è che il dominio del servizio catalogo utilizza un processo di elezione peer per definire il frammento primario a differenza dell'algoritmo del servizio catalogo che viene utilizzato per i contenitori.

Notare che il servizio di ubicazione ed il servizio di raggruppamento principale sono servizi uno-di-N. Un servizio uno-di-N funziona in un membro del gruppo HA (high availability). Il servizio di ubicazione e di amministrazione funziona in tutti i membri del gruppo HA (high availability). Il servizio di ubicazione ed il servizio di raggruppamento principale sono singleton poiché sono responsabili della caduta del sistema. Il servizio di ubicazione e di amministrazione sono servizi di sola lettura e sono presenti ovunque per fornire scalabilità.

Il servizio catalogo utilizza la replica per rendersi tollerante all'errore. Se si verifica un errore nel servizio catalogo, il servizio deve essere riavviato per ripristinare il sistema sul livello di disponibilità desiderato. Se si verifica un errore in tutti i processi che stanno ospitando il servizio catalogo, eXtreme Scale ha una perdita di dati critici. Questo errore comporta un riavvio obbligatorio di tutti i contenitori. Poiché il servizio catalogo può essere eseguito su molti processi, questo errore è un evento improbabile. Tuttavia, se si stanno eseguendo tutti i processi in una macchina singola, all'interno di uno chassis a lamella singola oppure da un singolo switch di rete, è più probabile che si verifichi un errore. Provare a rimuovere le modalità comuni di errore dalle caselle che stanno ospitando il servizio catalogo per ridurre la possibilità che si verifichi un errore.

Errori di più contenitori

Una replica non viene mai collocata nello stesso processo del suo elemento primario poiché se il processo viene perso, ciò comporterebbe una perdita sia dell'elemento primario che della replica. La politica di distribuzione definisce un attributo booleano di modalità di sviluppo che viene utilizzato dal servizio catalogo per determinare se una replica può essere collocata sulla stessa macchina del primario. In un ambiente di sviluppo su una singola macchina, è possibile che si voglia disporre di due contenitori ed eseguire repliche tra di essi. Tuttavia, in produzione, l'utilizzo di una singola macchina non è sufficiente perché la perdita di quell'host comporta la perdita di entrambi i contenitori. Per cambiare tra modalità di sviluppo su una singola macchina e una modalità di produzione con più macchine, disabilitare la modalità di sviluppo nel file di configurazione della politica di distribuzione.

Replica per disponibilità

La replica consente la tolleranza d'errore ed incrementa le prestazioni di una topologia eXtreme Scale distribuita.

La replica è abilitata tramite l'associazione delle BackingMap con una MapSet.

Una MapSet è una raccolta di mappe catalogate in base alla chiave di partizione. Questa chiave di partizione viene derivata dalla chiave della mappa singola prelevando dal suo modulo hash il numero di partizioni. Quindi, se un gruppo di mappe all'interno della MapSet ha una chiave di partizione X, tali mappe saranno memorizzate nella partizione X corrispondente nella griglia. Se un altro gruppo ha una chiave di partizione Y, tutte le mappe verranno memorizzate nella partizione Y e così via. Inoltre, tutti i dati nelle mappe vengono replicati in base alle politiche definite sulla MapSet, il che è valido solo per le topologie di eXtreme Scale distribuite (non necessario per le istanze locali).

Consultare “Partizionamento” a pagina 91 per ulteriori dettagli.

Alle MapSet vengono assegnati il numero di partizioni ed una politica di replica. La configurazione di replica della MapSet identifica semplicemente il numero di

frammenti di replica sincroni ed asincroni che una MapSet deve avere per il frammento primario. Ad esempio, se vi dovrà essere una replica sincrona ed una asincrona, ciascuna BackingMap assegnata alla MapSet avrà un frammento di replica automaticamente distribuito all'interno della serie di contenitori disponibili per eXtreme Scale. La configurazione di replica può anche consentire ai client di leggere i dati da server replicati in modo sincrono. Ciò potrà distribuire il carico delle richieste di lettura su più server in eXtreme Scale. La replica ha unicamente un impatto sul modello di programmazione quando si esegue il precaricamento delle BackingMap.

Per i dettagli relativi alle diverse opzioni di configurazione, consultare quanto segue:

Precaricamento della mappa

Le mappe possono essere associate ai programmi di caricamento. Un programma di caricamento viene utilizzato per eseguire il fetch di oggetti quando questi non vengono trovati nella mappa (una mancata corrispondenza nella cache) ed anche per scrivere le modifiche su un backend quando una transazione esegue il commit. I programmi di caricamento possono anche essere utilizzati per precaricare i dati in una mappa. Il metodo preloadMap dell'interfaccia Loader viene chiamato su ciascuna mappa quando la sua partizione corrispondente nella MapSet diventa primaria. Il metodo preloadMap non viene chiamato sulle repliche. Esso prova a caricare dal backend tutti i dati di riferimento previsti nella mappa utilizzando la sessione fornita. La mappa in questione viene identificata dall'argomento BackingMap passato al metodo preloadMap.

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

Precaricamento nella MapSet partizionata

Le mappe possono essere suddivise in N partizioni. Le mappe possono essere distribuite su più server, con ciascuna voce identificata da una chiave memorizzata solo su uno di questi server. È possibile gestire mappe molto grandi in eXtreme Scale poiché l'applicazione non è più limitata dalla dimensione heap di una singola JVM per contenere tutte le voci di una mappa. Le applicazioni che eseguono il precaricamento con il metodo preloadMap dell'interfaccia Loader devono identificare il sottoinsieme dei dati di cui esso effettua il precaricamento. Esiste sempre un numero fisso di partizioni. È possibile determinare questo numero utilizzando il seguente codice di esempio:

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

Questo codice di esempio mostra come un'applicazione può identificare il sottoinsieme dei dati da precaricare dal database. Le applicazioni devono sempre utilizzare questo metodo, anche quando la mappa non è inizialmente partizionata. Questo metodo consente flessibilità: se la mappa venisse in seguito partizionata dall'amministratore, il programma di caricamento continuerebbe a lavorare in modo corretto.

L'applicazione dovrà inviare delle query per poter recuperare la serie secondaria *myPartition* dal backend. Se si utilizza un database, potrebbe risultare più semplice avere una colonna con l'identificatore della partizione per un determinato record, a meno che non vi sia una query naturale che consenta ai dati nella tabella di essere partizionati in modo semplice.

Consultare i dettagli sulla scrittura di un programma di caricamento con un controller del precaricamento in *Guida alla programmazione* per un esempio su come implementare un programma di caricamento su un eXtreme Scale replicato.

Prestazioni

L'implementazione del precaricamento copia i dati dal backend nella mappa, ordinando più oggetti nella mappa in una singola transazione. Il numero ottimale di record da memorizzare per transazione dipende da diversi fattori, comprese la complessità e la dimensione. Ad esempio, dopo che la transazione ha incluso blocchi di più di 100 voci, il beneficio sulle prestazioni diminuisce all'aumentare del numero di voci. Per determinare il numero ottimale, cominciare con 100 voci ed aumentare il numero finché non si annulla il beneficio sulle prestazioni. Transazioni più grandi risultano in prestazioni di replica migliori. Ricordarsi che solo i frammenti primari eseguono il codice precaricato. I dati precaricati sono replicati dal frammento primario su qualsiasi replica in linea.

Precaricamento delle MapSet

Se l'applicazione utilizza una MapSet con più mappe ognuna di esse avrà il proprio programma di caricamento. Ciascun programma di caricamento ha un metodo di precaricamento. Ciascuna mappa viene caricata in modo seriale da eXtreme Scale. Potrebbe risultare più efficiente precaricare tutte le mappe designando una sola mappa come mappa di precaricamento. Questo processo rappresenta una convenzione di applicazione. Ad esempio, due mappe, department ed employee, potrebbero utilizzare il programma di caricamento di department per precaricare le mappe relative ai reparti ed agli impiegati. Questa procedura assicura che, dal punto di vista della transazione, se un'applicazione richiede un reparto di conseguenza anche tutti gli impiegati di tale reparto saranno nella cache. Quando il programma di caricamento di department precarica un reparto dal backend, esso esegue anche il fetch degli impiegati di tale reparto. L'oggetto department ed i suoi oggetti employee associati saranno quindi aggiunti alla mappa con un'unica transazione.

Precaricamento recuperabile

Alcuni clienti hanno insiemi di dati molto grandi che devono essere memorizzati nella cache. Il precaricamento di tali dati potrebbe richiedere molto tempo. In alcuni casi, prima che un'applicazione possa andare in linea si deve attendere il precaricamento dei dati. È possibile però beneficiare del precaricamento recuperabile. Si supponga di dover precaricare un milione di record. L'elemento primario inizia il precaricamento, ma va in errore dopo 800.000 record. Solitamente la replica scelta per diventare l'elemento primario, ripulisce i record dallo stato 'replicato' e riparte dall'inizio. eXtreme Scale può utilizzare un'interfaccia ReplicaPreloadController. Anche il programma di caricamento dell'applicazione dovrà implementare l'interfaccia. Questo esempio aggiunge un singolo metodo al programma di caricamento: `Status checkPreloadStatus(Session session, BackingMap bmap);`. Questo metodo viene chiamato dal runtime di eXtreme Scale prima che venga normalmente chiamato il metodo di precaricamento dell'interfaccia del programma di caricamento. Il programma eXtreme Scale verifica il risultato di questo metodo (Stato) per determinare il suo comportamento nel caso in cui una replica venisse promossa a primario.

Tabella 10. Valore di stato e risposta

Valori di stato restituiti	Risposta di eXtreme Scale
Status.PRELOADED_ALREADY	eXtreme Scale non chiama il metodo di precaricamento poiché questo valore di stato indica che la mappa è stata completamente caricata.
Status.FULL_PRELOAD_NEEDED	eXtreme Scale ripulisce la mappa e chiama il metodo di precaricamento normalmente.
Status.PARTIAL_PRELOAD_NEEDED	eXtreme Scale lascia la mappa com'è e chiama il precaricamento. Questa strategia consente al programma di caricamento dell'applicazione di proseguire il precaricamento da quel punto in avanti.

Chiaramente, mentre un elemento primario esegue il precaricamento di una mappa, dovrà lasciare uno stato in una mappa della MapSet che è in fase di replica, in modo da consentire alla replica di determinare quale stato restituire. È possibile utilizzare un'ulteriore mappa denominata ad esempio, RecoveryMap. Questa RecoveryMap dovrà far parte della stessa MapSet che è in fase di precaricamento, per poter essere sicuri che la mappa venga replicata in modo congruente con i dati che si stanno precaricando. Di seguito viene riportato un suggerimento di implementazione.

Appena il precaricamento effettua il commit di ciascun blocco di record, il processo aggiorna anche il contatore o il valore nella RecoveryMap come parte della transazione. I dati precaricati ed i dati della RecoveryMap vengono replicati automaticamente nelle repliche. Quando la replica viene promossa a primario potrà verificare RecoveryMap per vedere cosa è successo.

La RecoveryMap può contenere una singola voce con la chiave di stato. Se non esiste alcun oggetto per questa chiave sarà necessario un precaricamento completo (checkPreloadStatus returns FULL_PRELOAD_NEEDED). Se esiste un oggetto per questa chiave di stato ed il suo valore è COMPLETE, il precaricamento si completa, ed il metodo checkPreloadStatus restituisce PRELOADED_ALREADY. Altrimenti l'oggetto valore indica da dove deve essere riavviato il precaricamento ed il metodo checkPreloadStatus restituisce PARTIAL_PRELOAD_NEEDED. Il programma di caricamento può memorizzare il punto di recupero in una variabile di istanza del programma di caricamento in modo che, quando verrà chiamato il precaricamento, il programma di caricamento conoscerà il punto di inizio. La RecoveryMap potrà inoltre contenere una voce per ciascuna mappa se queste vengono caricate separatamente.

Gestione del recupero in modalità di replica sincrona con un programma di caricamento

Il runtime di eXtreme Scale è progettato per non perdere i dati su cui è stato eseguito il commit quando l'elemento primario va in errore. La seguente sessione mostra gli algoritmi utilizzati. Questi algoritmi si applicano solo quando un gruppo di replica utilizza la replica sincrona. L'utilizzo di un programma di caricamento è facoltativo.

Il runtime di eXtreme Scale può essere configurato per replicare tutte le modifiche da un elemento primario ad una replica in modo sincrono. Quando si posiziona una replica sincrona, essa riceve una copia dei dati esistenti sul frammento primario. In questo frattempo, il frammento primario continua a ricevere transazioni e le copia sulla replica in modo asincrono. La replica a questo punto non è considerata in linea.

Dopo che la replica ha raggiunto il frammento primario entra in modalità peer ed inizia la replica sincrona. Ogni transazione di cui si effettua il commit sul frammento primario viene inviata alle repliche sincrone ed il frammento primario attende una risposta da ciascuna replica. Una sequenza di commit sincrona con il programma di caricamento sul frammento primario, assomiglia alla seguente serie di fasi:

Tabella 11. Sequenza commit sul frammento primario

Fase con il programma di caricamento	**Fase senza programma di caricamento**
Ricezione dei blocchi per le voci	uguale
Esecuzione del flush delle modifiche sul programma di caricamento	no-op
Salvataggio delle modifiche nella cache	uguale
Invio delle modifiche alle repliche ed attesa della conferma di ricezione	uguale
Esecuzione del commit sul programma di caricamento con il plug-in TransactionCallback	chiamata al commit del plug-in, ma nulla viene fatto
Rilascio dei blocchi sulle voci	uguale

Si noti come le modifiche vengono inviate alla replica prima che su di esse si esegua il commit sul programma di caricamento. Per determinare quando viene effettuato il commit delle modifiche sulla replica, consultare questa sequenza: al momento dell'inizializzazione, inizializzare gli elenchi tx sul frammento primario come riportato di seguito.

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

Durante l'elaborazione del commit singolo, utilizzare la seguente sequenza:

Tabella 12. Elaborazione del commit sincrono

Fase con il programma di caricamento	**Fase senza programma di caricamento**
Ricezione dei blocchi per le voci	uguale
Esecuzione del flush delle modifiche sul programma di caricamento	no-op
Salvataggio delle modifiche nella cache	uguale
Invio delle modifiche con una transazione su cui è stato effettuato il commit. Rollback della transazione sulla replica ed attesa della conferma di ricezione	uguale
Cancellazione dell'elenco di transazioni di cui si è effettuato il commit. Rollback delle transazioni	uguale
Esecuzione del commit del programma di caricamento con il plug-in TransactionCallBack	chiamata del commit del plug-in TransactionCallBack, ma solitamente nulla viene effettuato
Se il commit riesce, aggiunta della transazione alle transazioni di cui si è effettuato il commit, altrimenti aggiunta alle transazioni di cui si è effettuato il rollback	no-op
Rilascio dei blocchi sulle voci	uguale

Per l'elaborazione della replica, utilizzare la seguente sequenza:

1. Ricezione delle modifiche
2. Esecuzione del commit di tutte le transazioni ricevute nell'elenco delle transazioni di cui si è effettuato il commit
3. Esecuzione del rollback di tutte le transazioni ricevute nell'elenco delle transazioni di cui si è effettuato il rollback
4. Avvio di una transazione o di una sessione
5. Applicazione delle modifiche alla transazione o alla sessione
6. Salvataggio della transazione o della sessione nell'elenco 'in sospeso'
7. Invio della risposta

Si noti che, sulla replica, non si verificano interazioni del programma di caricamento mentre essa è in modalità di replica. Il frammento primario deve trasmettere tutte le modifiche attraverso il programma di caricamento. La replica non effettuerà alcuna modifica. Un effetto collaterale di questo algoritmo è che le repliche avranno le transazioni, ma su di esse non verrà eseguito alcun commit finché la successiva transazione del frammento primario non invierà lo stato del commit di tali transazioni. Sulla replica verrà quindi eseguito il commit ed il rollback delle transazioni. Fino ad allora sulle transazioni non verrà eseguito il commit. È possibile aggiungere un timer sul frammento primario che invii il risultato della transazione dopo un breve periodo di tempo (pochi secondi). Questo timer limita, ma non elimina, la pesantezza di questa finestra temporale. Questa pesantezza rappresenta un problema solo quando si utilizza la replica in modalità di lettura. Altrimenti, essa non ha alcun impatto sull'applicazione.

Quando il frammento primario va in errore, è possibile che su di esso si sia eseguito il commit o il rollback di alcune transazioni, ma che il messaggio non abbia raggiunto la replica con i seguenti risultati. Quando una replica viene promossa a frammento primario, una delle prime azioni è quella di gestire questa condizione. Ciascuna transazione pendente viene nuovamente elaborata sulla serie di mappe del frammento primario. Se c'è un programma di caricamento, ciascuna transazione viene consegnata ad esso. Queste transazioni vengono applicate in rigido ordine FIFO (first in first out). Se una transazione va in errore, viene ignorata. Se tre transazioni sono in sospeso, A, B e C, sulla A si esegue il commit, sulla B il rollback e sulla C di nuovo il commit. Nessuna transazione ha alcun impatto sulle altre. Sono da considerare indipendenti.

Un programma di caricamento potrebbe utilizzare una logica leggermente differente quando si trova nella modalità di recupero da failover rispetto alla modalità normale. Il programma di caricamento può venire facilmente a conoscenza di essere in modalità di recupero da failover implementando l'interfaccia ReplicaPreloadController. Il metodo checkPreloadStatus viene chiamato unicamente quando il recupero da failover viene completato. Quindi, se il metodo apply dell'interfaccia del programma di caricamento viene chiamato prima del metodo checkPreloadStatus, significa che si tratta di una transazione di recupero. Il recupero da failover viene completato dopo che è stato chiamato il metodo checkPreloadStatus.

Bilanciamento del carico tra repliche

eXtreme Scale, se non altrimenti configurato, invia tutte le richieste di lettura e scrittura sul server primario di un dato gruppo di replica. Il primario deve servire tutte le richieste provenienti dai client. Potrebbe essere consigliabile inviare le richieste di lettura sulle repliche del primario. In questo modo si condividerà il

carico delle richieste di lettura tra più JVM (Java Virtual Machine). Tuttavia, l'utilizzo delle repliche per le richieste di lettura potrebbe generare delle risposte incongruenti.

Il bilanciamento del carico tra le repliche viene solitamente utilizzato quando i client memorizzano nella cache dati che vengono frequentemente modificati oppure quando questi utilizzano il blocco pessimistico.

Se i dati vengono cambiati in continuazione e quindi invalidati nelle cache locali dei client, come risultato i primari rileveranno da parte dei client una frequenza di richieste get relativamente alta. Allo stesso modo, nella modalità di blocco pessimistico, non esiste alcuna cache locale, quindi tutte le richieste saranno inviate al primario.

Se i dati sono relativamente statici oppure se non si utilizza la modalità pessimistica, l'invio delle richieste di lettura non avrà un grande impatto sulle prestazioni. La frequenza delle richieste get da parte di client con cache piene di dati non è alta.

Quando un client parte per la prima volta, la sua cache locale è vuota. Le richieste di cache ad una cache vuota vengono inoltrate al primario. Il client con il passar del tempo riceve dati, diminuendo così il carico delle richieste. Se viene avviato contemporaneamente un gran numero di client, il carico potrebbe risultare significativo e quindi l'opzione della lettura della replica potrebbe rivelarsi una scelta appropriata per le prestazioni.

Replica lato client

Con eXtreme Scale, è possibile replicare una mappa del server su uno o più client utilizzando la replica asincrona. Un client può richiedere una copia locale in sola lettura della mappa lato server utilizzando il metodo ClientReplicableMap.enableClientReplication.

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

Il primo parametro rappresenta la modalità di replica. Questa modalità può essere continua o istantanea. Il secondo parametro è un array di ID di partizioni che rappresenta le partizioni da cui iniziare la replica dei dati. Se il valore è nullo o un array vuoto, i dati vengono replicati da tutte le partizioni. L'ultimo parametro è un listener per ricevere gli eventi relativi alle repliche dei client. Consultare ClientReplicableMap e ReplicationMapListener nella documentazione API per i dettagli.

Dopo aver abilitato la replica, il server comincia a replicare la mappa sul client. In questo modo il client resterà indietro di poche transazioni rispetto al server in ogni memento.

Tipi di rilevamento failover

WebSphere eXtreme Scale è in grado di rilevare gli errori in modo affidabile.

Funzione di heartbeat

1. I socket vengono lasciati aperti tra Java virtual machine, e se un socket viene chiuso improvvisamente, tale chiusura viene rilevata come errore della Java virtual machine peer. Questo rilevamento individua casi di errori come, ad esempio, l'uscita molto rapida di Java virtual machine dal processo. Tali

operazioni di rilevamento consentono inoltre il ripristino da questi tipi di malfunzionamento solitamente in meno di un secondo.

2. Altri tipi di errore includono: blocco improvviso del sistema operativo, errore del server fisico o errore di rete. Tali errori vengono rilevati attraverso la funzione di heartbeat.

Gli heartbeat vengono inviati periodicamente tra coppie di processi: quando un determinato numero di heartbeat mancano, si suppone che si sia verificato un errore. Questo approccio rileva gli errori in N*M secondi, dove N è il numero di heartbeat mancanti e M è l'intervallo di impostazione degli heartbeat. Non è consentito specificare direttamente M e N; invece, l'impiego di un meccanismo a cursore consente di utilizzare un intervallo di combinazioni M e N verificate.

Errori

Un processo può terminare in modo anomalo per diversi motivi. Il processo può terminare in modo anomalo a causa di un eccessivo utilizzo di risorse, come la dimensione heap massima, o a causa della logica di controllo che ha terminato un processo. Il sistema operativo potrebbe non funzionare correttamente, provocando la perdita di tutti i processi in esecuzione sul sistema. L'hardware potrebbe presentare un errore, sebbene meno frequentemente, come la scheda NIC (network interface card), causando lo scollegamento dalla rete del sistema operativo. In questo contesto, tutti questi errori possono essere ricondotti a uno o due tipi di categorie: errore del processo e perdita della connettività.

Errore del processo

WebSphere eXtreme Scale reagisce molto rapidamente agli errori di processi. Quando si verifica un errore nel processo, il sistema operativo è responsabile della ripulitura di eventuali risorse non più utilizzate dal processo. Questa ripulitura include l'allocazione della porta e la connettività. Quando si verifica un errore nel processo, sulle connessioni che erano utilizzate da quel processo viene immediatamente inviato un segnale per chiudere ogni connessione.

Perdita di connettività

La perdita di connettività si verifica quando il sistema operativo si scollega. Come risultato, il sistema operativo non può inviare segnali ad altri processi. La perdita della connettività può verificarsi per vari motivi ma possono essere divisi in due categorie: errore host e isolamento.

Errore host

Se una macchina host perde potenza, essa diventa immediatamente non disponibile.

Isolamento

Questo scenario rappresenta la condizione di errore più complicata da gestire per il software poiché si presume che il processo non sia disponibile mentre invece lo è.

Errore del contenitore

Gli errori del contenitore vengono scoperti generalmente dai contenitori peer attraverso il meccanismo del gruppo principale. Quando un contenitore o una serie di contenitori va in errore, il servizio catalogo esegue la migrazione dei frammenti

ospitati su quel contenitore o contenitori. Il servizio catalogo cerca prima una replica sincrona prima di eseguire la migrazione ad una replica asincrona. Dopo la migrazione dei frammenti primari nei nuovi contenitori dell'host, il servizio catalogo cerca nuovi contenitori dell'host per le repliche che ora stanno mancando.

Nota: Isolamento del contenitore - il servizio catalogo esegue la migrazione dei frammenti fuori dai contenitori quando si scopre che il contenitore non è disponibile. Se questi contenitori diventano poi disponibili, il servizio catalogo considera i contenitori collocabili proprio come nel normale flusso di avvio.

Latenza rilevamento failover del contenitore

Gli errori possono essere categorizzati in errori soft e hard. Gli errori soft sono di solito causati da un malfunzionamento del processo. Questi errori vengono rilevati dal sistema operativo, che può recuperare le risorse utilizzate, come i socket di rete, molto velocemente. Il tipico rilevamento di errore per quanto riguarda errori soft è inferiore a un secondo. Il rilevamento degli errori hard può impiegare fino a 200 secondi utilizzando la sintonizzazione heartbeat predefinita. Errori di questo tipo includono: rotture fisiche delle macchine, scollegamento dei cavi di rete o errori del sistema operativo. Così, eXtreme Scale deve fare affidamento sull'heartbeat per poter rilevare errori hard che possono essere configurati.

Errori di più contenitori

Una replica non viene mai collocata nello stesso processo del suo elemento primario poiché se il processo viene perso, ciò comporterebbe una perdita sia dell'elemento primario che della replica. La politica di distribuzione definisce un attributo che viene utilizzato dal servizio catalogo per determinare se una replica può essere collocata sulla stessa macchina dell'elemento primario. In un ambiente di sviluppo su una singola macchina, è possibile che si voglia disporre di due contenitori ed eseguire repliche tra di essi. Tuttavia, in produzione, l'utilizzo di una singola macchina non è sufficiente perché la perdita di quell'host comporta la perdita di entrambi i contenitori. Per cambiare tra modalità di sviluppo su una singola macchina e una modalità di produzione con più macchine, disabilitare la modalità di sviluppo nel file di configurazione della politica di distribuzione.

Errore del servizio catalogo

Poiché la griglia del servizio catalogo è una griglia eXtreme Scale, questa utilizza anche il meccanismo del raggruppamento principale nello stesso modo del processo di errore del contenitore. La differenza principale è che il dominio del servizio catalogo utilizza un processo di elezione peer per definire il frammento primario a differenza dell'algoritmo del servizio catalogo che viene utilizzato per i contenitori.

Si noti che il servizio di collocazione ed il servizio di raggruppamento principale sono servizi uno-di-N. Un servizio uno-di-N viene eseguito in un membro del gruppo HA (High Availability). Il servizio di ubicazione e la gestione vengono eseguiti in tutti i membri del gruppo HA (High Availability). Il servizio di collocazione ed il servizio di raggruppamento principale sono singleton poiché sono responsabili della caduta del sistema. Il servizio di posizionamento e di amministrazione sono servizi di sola lettura e sono presenti ovunque per fornire scalabilità.

Il servizio catalogo utilizza la replica per rendersi tollerante all'errore. Se si verifica un errore nel processo del servizio catalogo, il servizio deve essere riavviato per

ripristinare il sistema sul livello di disponibilità desiderato. Se si verifica un errore in tutti i processi che ospitano il servizio catalogo, eXtreme Scale ha una perdita di dati critici. Questo errore comporta un riavvio obbligatorio di tutti i contenitori. Poiché il servizio catalogo può essere eseguito su molti processi, questo errore è un evento improbabile Tuttavia, se si stanno eseguendo tutti i processi su una singola macchina, all'interno di uno chassis a lamella singola oppure da un singolo switch di rete, è più probabile che si verifichi un errore. Provare a rimuovere le modalità comuni di errore dalle macchine che ospitano il servizio catalogo per ridurre la possibilità che si verifichi un errore.

Tabella 13. Rilevamento dell'errore e riepilogo del recupero

Tipo di perdita	Meccanismo di rilevamento	Metodo di recupero
Perdita processo	I/O	Riavvio
Perdita server	Heartbeat	Riavvio
Interruzione di rete	Heartbeat	Ripristino della rete e connessione
Blocco lato server	Heartbeat	Arresto e riavvio del server
Server occupato	Heartbeat	Attesa fino a disponibilità server

Servizio catalogo ad alta disponibilità

Un dominio del servizio catalogo è la griglia dei server di catalogo che si sta utilizzando, che conserva le informazioni di topologia per tutti i contenitori del proprio ambiente eXtreme Scale. Il servizio catalogo controlla il bilanciamento e l'instradamento per tutti i client. Per distribuire eXtreme Scale come spazio di elaborazione del database in memoria, si deve raggruppare il servizio catalogo in una griglia per alta disponibilità.

Componenti del dominio del servizio catalogo

Quando vengono avviati più server di catalogo, uno dei server viene definito server di catalogo principale; questo server accetta gli heartbeat IIOP (Internet Inter-ORB Protocol) e gestisce le modifiche dei dati di sistema in risposta a qualsiasi modifica del servizio catalogo o del contenitore.

Quando i client contattano uno dei server di catalogo, la tabella di instradamento per il dominio del servizio catalogo viene trasmessa ai client mediante il contesto del servizio CORBA (Common Object Request Broker Architecture).

Configurare almeno tre server di catalogo. Se nella propria configurazione vi sono zone, è possibile configurare un server di catalogo per zona.

Quando un server e un contenitore eXtreme Scale contattano uno dei server di catalogo, la tabella di instradamento per il dominio del servizio catalogo viene trasmessa anche al service e al contenitore eXtreme Scale attraverso il contesto del servizio CORBA. Inoltre, se il server di catalogo contattato non è attualmente il server di catalogo principale, la richiesta viene automaticamente reinstradata al server di catalogo principale corrente e la tabella di instradamento per il server di catalogo viene aggiornata.

Nota: La griglia di un server di catalogo e quella di un server contenitore sono molto diverse. Il dominio del servizio catalogo è utilizzato per l'alta disponibilità dei dati di sistema. La griglia del contenitore per l'alta disponibilità dei dati, la

scalabilità e la gestione del carico di lavoro. Pertanto, vi sono due tabelle di instradamento differenti: la tabella di instradamento per il dominio del servizio catalogo e quella per i frammenti della griglia di server.

Le responsabilità del catalogo sono suddivise in una serie di servizi. Il gestore del gruppo principale esegue un raggruppamento peer per il monitoraggio dell'integrità, il servizio di posizionamento effettua l'allocazione, il servizio di amministrazione fornisce accesso alla gestione e il servizio di ubicazione gestisce le località.

Distribuzione dominio del servizio catalogo

Gestore del gruppo principale

Il servizio catalogo utilizza il gestore HA (High Availability) per raggruppare i processi in base al monitoraggio della disponibilità. Ogni raggruppamento dei processi è un gruppo principale. Con eXtreme Scale il gestore del gruppo principale raggruppa insieme i processi in modo dinamico. La dimensione di questi processi è mantenuta su valori ridotti per consentirne la scalabilità. Ogni gruppo principale stabilisce un leader che ha la responsabilità aggiuntiva di inviare lo stato al gestore del gruppo principale quando i singoli membri non funzionano correttamente. Lo stesso meccanismo di stato viene utilizzato per rilevare quando tutti i membri di un gruppo non funzionano correttamente, situazione che provoca la mancata comunicazione con il leader.

Il gestore del gruppo principale è un servizio completamente automatico, responsabile dell'organizzazione dei contenitori in piccoli gruppi di server che vengono poi automaticamente federati per realizzare un ObjectGrid. Quando un contenitore inizialmente contatta un servizio catalogo, attende di essere assegnato ad un gruppo nuovo o esistente. Una distribuzione di eXtreme Scale è costituita da molti gruppi di questo tipo e questo raggruppamento è una funzione di abilitazione chiave della scalabilità. Ogni gruppo è un gruppo di Java virtual machine che utilizza l'heartbeat per monitorare la disponibilità di altri gruppi. In uno di questi gruppi viene stabilito il leader che ha la responsabilità aggiuntiva di inoltrare informazioni di disponibilità al servizio catalogo per consentire la reazione a errori mediante la riallocazione e l'instradamento.

Servizio di posizionamento

Il servizio catalogo gestisce il posizionamento dei frammenti tra la serie di contenitori disponibili. Il servizio di posizionamento è responsabile della gestione dell'equilibrio delle risorse tra risorse fisiche. Il servizio di posizionamento è responsabile dell'allocazione dei singoli frammenti nei relativi contenitori dell'host. Il servizio di posizionamento viene eseguito come uno degli N servizi eletti nella griglia dei dati, quindi esattamente un'istanza del servizio è in esecuzione. Se l'istanza non riesce correttamente, viene eletto un altro processo e subentra al suo posto. Per ridondanza lo stato del servizio catalogo viene replicato tra tutti i server che stanno ospitando il servizio catalogo.

Gestione

Il servizio catalogo è anche il punto di ingresso logico per la gestione del sistema. Il servizio catalogo ospita un Managed Bean (MBean) e fornisce URL JMX (Java Management Extension) per uno dei server che sta gestendo il servizio.

Servizio di ubicazione

Il servizio di ubicazione agisce come un touchpoint per entrambi i client che stanno cercando i contenitori che ospitano l'applicazione ricercata, così come i contenitori che stanno registrando le applicazioni ospitate con il servizio di posizionamento. Il servizio di ubicazione viene eseguito su tutti i membri della griglia per scalare questa funzione.

Il servizio catalogo ospita la logica che è tipicamente inattiva durante gli stati stabili. Di conseguenza, il servizio catalogo influisce minimamente sulla scalabilità. Il servizio viene creato per servire centinaia di contenitori che diventano disponibili contemporaneamente. Per la disponibilità configurare il servizio catalogo in una griglia.

Pianificazione

Dopo l'avvio di un dominio del servizio catalogo, viene eseguito il bind dei membri della griglia. Pianificare attentamente la topologia del dominio del servizio catalogo, poiché non è possibile modificare la configurazione del catalogo al runtime. Distribuire la griglia nel modo più differenziato possibile per impedire errori.

Avvio di un dominio del servizio catalogo

Per ulteriori informazioni sulla creazione di un dominio del servizio catalogo, consultare i dettagli relativi all'avvio di un dominio del servizio catalogo in *Guida alla gestione*.

Connessione di contenitori eXtreme Scale incorporati in WebSphere Application Server ad un dominio del servizio catalogo autonomo

È possibile configurare i contenitori eXtreme Scale incorporati in un ambiente WebSphere Application Server per connettersi ad un dominio del servizio catalogo autonomo.

-  (obsoleto) Nelle release precedenti, si effettuava la connessione dei servizi catalogo nel dominio del servizio catalogo creando una proprietà personalizzata. Questa proprietà può ancora essere utilizzata ma è obsoleta. Per ulteriori informazioni su questa proprietà personalizzata, consultare le informazioni relative all'avvio del processo del servizio catalogo in WebSphere Application Server in *Guida alla gestione*..

  Nota: Collisione del nome server: poiché questa proprietà viene utilizzata per avviare il server di catalogo eXtreme Scale come anche per collegarsi ad esso, i server di catalogo non devono avere lo stesso nome di alcun server WebSphere Application Server.

Per ulteriori informazioni, consultare “Quorum del server di catalogo”.

Quorum del server di catalogo

Il quorum è il numero minimo di server di catalogo necessario per condurre le operazioni di posizionamento per la griglia dei dati. Il numero minimo è rappresentato dall'intera serie dei server di catalogo con cui si è sostituito il quorum.

Termini importanti

Di seguito viene riportato un elenco di termini correlati alle considerazioni sul quorum per WebSphere eXtreme Scale.

- **Brown out**: un brown out è una perdita temporanea della connettività tra uno o più server.
- **Black out**: un black out è una perdita permanente della connettività tra uno o più server.
- **Centro dati**: un centro dati è un gruppo di server geograficamente ubicato generalmente connesso con una LAN (local area network).
- **Zona**: una zona è un'opzione di configurazione utilizzata per raggruppare i server tra loro e condividere alcune caratteristiche fisiche. Alcuni esempi di zone per un gruppo di server sono: un centro dati come descritto in precedenza, una rete d'area, un palazzo, il piano di un palazzo e così via.
- **Heartbeat**: il meccanismo Heartbeat viene utilizzato per determinare se una JVM (Java virtual machine) è in esecuzione o meno.

Topologia

Questa sezione descrive come WebSphere eXtreme Scale opera su una rete che comprende componenti non affidabili. Alcuni esempi di una tale rete includono una rete che si estende su più centri dati.

Spazio indirizzi IP

WebSphere eXtreme Scale richiede una rete dove qualsiasi elemento indirizzabile in essa presente possa collegarsi senza impedimenti ad un qualsiasi altro elemento indirizzabile. Ciò significa che WebSphere eXtreme Scale richiede uno spazio dei nomi dell'indirizzo IP piatto (Flat) e che tutti i firewall consentano il flusso di tutto il traffico tra gli indirizzi IP e le porte utilizzate dalle JVM (Java virtual machine) che ospitano elementi di WebSphere eXtreme Scale.

LAN collegate

Come requisito di WebSphere eXtreme Scale ad ogni LAN viene assegnato un identificatore di zona. In una singola zona WebSphere eXtreme Scale invia segnali heartbeat alle JVM in quantità consistente. Se il servizio catalogo ha il quorum un singolo heartbeat fallito genererà un evento di failover.

Dominio del servizio catalogo e server dei contenitori

Un dominio del servizio catalogo rappresenta una raccolta di JVM simili. Un dominio del servizio catalogo è una griglia composta da server di catalogo ed è a dimensione fissa. Tuttavia, il numero di server contenitore è dinamico. I server contenitore possono essere aggiunti e rimossi su richiesta. In una configurazione con tre centri dati, WebSphere eXtreme Scale richiede una JVM del servizio catalogo per ciascun centro dati.

Il dominio del servizio catalogo utilizza un meccanismo di quorum pieno. A causa di questo meccanismo di quorum pieno, tutti i membri della griglia dei dati devono concordare su qualsiasi azione.

Le JVM del server contenitore sono identificate con un identificativo di zona. La griglia delle JVM contenitore viene automaticamente suddivisa in piccoli gruppi

principali di JVM. Un gruppo principale contiene unicamente JVM della stessa zona. Le JVM di zone diverse non sono mai nello stesso gruppo principale.

Un gruppo principale tenta di rilevare l'errore delle proprie JVM. Le JVM contenitore non dovranno mai estendersi su più LAN connesse tra loro con collegamenti quali una WAN (wide area network). Ciò significa che i gruppi principali non potranno avere contenitori appartenenti alla stessa zona in esecuzione su centri dati differenti.

Ciclo di vita di un server

Avvio di un server di catalogo

I server di catalogo vengono avviati con il comando startOgServer. Il meccanismo di quorum per impostazione predefinita è disabilitato. Per abilitarlo utilizzare l'indicatore abilitato `-quorum` nel comando startOgServer oppure aggiungere la proprietà `enableQuorum=true` nel file delle proprietà. A tutti i server di catalogo devono essere assegnate le stesse impostazioni di quorum.

```
# bin/startOgServer cat0 –serverProps objectGridServer.properties
```

```
File objectGridServer.properties
catalogClusterEndPoints=cat0:cat0.domain.com:6600:6601,
cat1:cat1.domain.com:6600:6601
catalogServiceEndPoints= cat0.domain.com:2809, cat1.domain.com:2809
enableQuorum=true
```

Avvio di un server contenitore

I server contenitore vengono avviati con il comando startOgServer. Quando una griglia dei dati è in esecuzione su più centri dati i server devono utilizzare la tag zona per identificare il centro dati in cui risiedono. L'impostazione della zona sui server della griglia consente a WebSphere eXtreme Scale di monitorare l'integrità dei server contenuti nell'ambito del centro dati, minimizzando il traffico tra i centri dati.

```
# bin/startOgServer gridA0 –serverProps objectGridServer.properties –
objectgridfile xml/objectgrid.xml –deploymentpolicyfile xml/
deploymentpolicy.xml
```

```
File objectGridServer.properties
catalogServiceEndPoints= cat0.domain.com:2809, cat1.domain.com:2809
zoneName=ZoneA
```

Arresto di un server della griglia

I server della griglia vengono arrestati con il comando stopOgServer. Quando si arresta un intero centro dati per manutenzione, passare l'elenco di tutti i server appartenenti a tale zona. Ciò consentirà una transizione di stato pulita da una zona in fase di arresto alla zona o alle zone ancora attive.

```
# bin/stopOgServer gridA0,gridA1,gridA2 –catalogServiceEndPoints
cat0.domain.com:2809,cat1.domain.com:2809
```

Rilevamento errori

WebSphere eXtreme Scale rileva la cessazione di un processo attraverso gli eventi di chiusura anomala dei socket. Il servizio catalogo verrà immediatamente informato del termine di un processo. Attraverso gli heartbeat falliti si rileverà un

black-out. WebSphere eXtreme Scale si protegge dalle condizioni di brown out tra i centri dati con l'implementazione del quorum.

Implementazione degli Heartbeat

Questa sezione descrive come viene implementata la verifica della vitalità in WebSphere eXtreme Scale.

Heartbeat tra i membri del gruppo principale

Il servizio catalogo posiziona delle JVM contenitore in gruppi principali di dimensione limitata. Un gruppo principale tenterà di rilevare l'errore dei propri membri utilizzando due metodi. Se si chiude il socket di una JVM, essa viene considerata inattiva. Ciascun membro invia heartbeat su questi socket ad una frequenza determinata dalla configurazione. Se una JVM non risponde agli heartbeat in un periodo massimo di tempo configurato la JVM viene considerata inattiva.

Un singolo membro di un gruppo principale viene sempre eletto come leader. Il CGL (core group leader) è responsabile di informare periodicamente il servizio del catalogo che il gruppo principale è attivo e di riportare qualsiasi cambiamento relativo all'appartenenza dei membri del gruppo al servizio catalogo. Un modifica dell'appartenenza dei membri al gruppo può essere rappresentata da una JVM in errore o dall'aggiunta di una JVM al gruppo principale.

Se il CGL (core group leader) non è in grado di contattare alcun membro del dominio del servizio catalogo continuerà a provare.

Heartbeat del dominio del servizio catalogo

Il dominio del servizio catalogo assomiglia ad un gruppo principale privato con un'appartenenza dei membri statica ed un meccanismo di quorum. Esso rileva gli errori allo stesso modo di un normale gruppo principale. Tuttavia, il comportamento è modificato in modo da accogliere la logica del quorum. Il servizio catalogo inoltre utilizza una configurazione degli heartbeat meno aggressiva.

Heartbeat del gruppo principale

Il servizio catalogo deve essere informato quando i server contenitore vanno in errore. Ciascun gruppo principale è responsabile di rilevare l'errore di una JVM contenitore e di riportarlo al servizio catalogo tramite il CGL (core group leader). È possibile che si verifichi l'errore di tutti i membri di un gruppo principale. Se ciò accade, è responsabilità del servizio catalogo rilevarlo.

Se il servizio catalogo contrassegna la JVM contenitore come in errore ed il contenitore viene successivamente segnalato come attivo, alla JVM contenitore verrà richiesto di arrestare i server contenitore di WebSphere eXtreme Scale. Una JVM con questo stato non è visibile nelle query xsadmin. Ciò viene riportato dai messaggi nel log della JVM contenitore. È necessario riavviare manualmente queste JVM.

Se si verifica un evento di perdita del quorum, l'invio degli heartbeat viene sospeso finché non lo si ristabilisce.

Comportamento del quorum del servizio catalogo

Solitamente i membri di una servizio catalogo hanno una piena connettività. Il dominio del servizio catalogo è rappresentato da una serie statica di JVM. WebSphere eXtreme Scale prevede che tutti i membri di un servizio catalogo siano in linea. Il servizio catalogo risponde agli eventi del contenitore finché ha il quorum.

Se il servizio catalogo perde il quorum, attende che questo venga ristabilito. In questo periodo di tempo il servizio catalogo ignora gli eventi provenienti dai server contenitore. I server contenitore tentano di inviare nuovamente tutte le richieste rifiutate dal server di catalogo mentre WebSphere eXtreme Scale attende che il quorum venga ristabilito.

Il seguente messaggio indica che si è perduto il quorum. Cercare questo messaggio nel log del proprio servizio catalogo.

```
CWOBJ1254W: Il servizio catalogo è in attesa del quorum.
```

WebSphere eXtreme Scale prevede di perdere il quorum per le seguenti ragioni:

- Errore di un membro JVM del servizio catalogo
- Rete in stato di brown out
- Perdita del centro dati

L'arresto di un'istanza del server di catalogo con `stopOgServer` non causa la perdita del quorum poiché il sistema è a conoscenza del fatto che l'istanza del server è stata arrestata, il che è diverso da un errore della JVM o da un brown out.

Perdita del quorum a causa di un errore di una JVM

Un errore nel server di catalogo causerà la perdita del quorum. In questo caso il quorum deve essere sostituito nel modo più veloce possibile. Il servizio catalogo in errore non può ricongiungersi alla griglia finché il quorum non viene sostituito.

Perdita del quorum a causa del brown out della rete

La progettazione di WebSphere eXtreme Scale prevede la possibilità dei brown out. Un brown out si verifica quando c'è una perdita temporanea della connettività tra i centri dati. Questa condizione ha una natura solitamente transitoria ed i brown out dovrebbero risolversi in secondi o minuti. Mentre WebSphere eXtreme Scale cerca di gestire le normali operazioni durante il brown out, esso viene considerato come un singolo evento errore. Si prevede che l'errore venga risolto e che le normali operazioni possano riprendere senza la necessità da parte di WebSphere eXtreme Scale di intraprendere alcuna azione.

Un brown out di lunga durata può essere classificato come black out solo attraverso l'intervento di un utente. Per poter classificare un brown out come black out, è necessario che il quorum venga sostituito su un lato del brown out.

Ciclo di una JVM del servizio catalogo

Se un server di catalogo viene arrestato con stopOgServer, il quorum viene abbassato ad un server in meno. Ciò significa che i server rimanenti continueranno ad aver il quorum. Il riavvio del server di catalogo riporta il quorum al numero precedente.

Conseguenze della perdita del quorum

Se, nel momento in cui si è perso il quorum, una JVM contenitore era in procinto di andare in errore, il ripristino non avviene finché non si recupera dal brown out oppure, in caso di black out, finché il cliente non esegue il comando di sostituzione del quorum. WebSphere eXtreme Scale considera l'evento di perdita del quorum e l'errore del contenitore come un doppio errore, ciò rappresenta un caso raro. Questo comporta la perdita dell'accesso in scrittura sui dati memorizzati sulla JVM in errore da parte dell'applicazione, finché il quorum non viene ripristinato, momento in cui verrà effettuato anche il normale recupero.

In modo simile, se si cerca di avviare un contenitore durante un evento di perdita del quorum, il contenitore non si avvierà.

Durante l'assenza del quorum è concessa la piena connettività del client. Se durante l'evento di perdita del quorum non si verificano problemi di connettività o errori con i contenitori, i client saranno in grado di interagire completamente con il server contenitore.

Se si verifica un brown out, alcuni client potranno perdere l'accesso alle repliche o alle copie dei dati finché l'evento non si risolve.

Sarà possibile avviare nuovi client, poiché vi dovrebbe essere una JVM del servizio catalogo in ciascun centro dati, quindi sarà possibile per un client raggiungere almeno una JVM del servizio catalogo anche durante un evento di brown out.

Recupero del quorum

Se per un qualsiasi motivo si perde il quorum, quando esso sarà ristabilito verrà eseguito un protocollo di recupero. Quando si verifica un evento di perdita del quorum, tutte le verifiche della vitalità per i gruppi principali vengono sospese ed i report di errore sono ignorati. Una volta ripristinato il quorum il servizio catalogo effettua una verifica della vitalità di tutti i gruppi principali per determinarne immediatamente i membri appartenenti. In questo momento si recupereranno tutti i frammenti precedentemente ospitati sulle JVM contenitore. Se sono stati persi dei frammenti primari, le repliche ancora esistenti saranno promosse a primari. Se i frammenti di replica sono andati perduti, verranno create delle repliche aggiuntive su quelli ancora esistenti.

Sostituzione del quorum

Questa operazione dovrà essere eseguita solo in caso di errore del centro dati. La perdita del quorum dovuta ad un errore della JVM del servizio catalogo o ad un brown out della rete dovrebbe essere recuperato automaticamente una volta riavviata la JVM del servizio catalogo oppure nel momento in cui si risolve il brown out della rete.

Gli amministratori saranno gli unici a conoscenza di un errore del centro dati. WebSphere eXtreme Scale gestisce un brown out ed un black out in modo simile. È necessario informare l'ambiente eXtreme Scale dell'errore utilizzando il comando xsadmin per sostituire il quorum. Ciò informerà il servizio catalogo che il quorum viene raggiunto con l'attuale numero di server di catalogo ed al contempo si avvierà un ripristino completo. Quando si immette un comando di sostituzione del quorum, si sta assicurando che le JVM nel centro dati in errore siano effettivamente in errore e che non recupereranno.

Il seguente elenco prende in considerazione alcuni scenari per la sostituzione del quorum. Si consideri di avere tre server di catalogo: A, B e C.

- Brown out: si supponga di avere un brown out in cui C è temporaneamente isolato. Il servizio catalogo perderà il quorum ed attenderà la fine del brown out, momento in cui C si ricongiungerà al dominio del servizio catalogo ed il quorum verrà ristabilito. Le applicazioni non riscontreranno alcun problema durante tutto questo periodo.
- Errore temporaneo: in questo scenario, C va in errore ed il servizio catalogo perde il quorum, sarà quindi necessario sostituire il quorum. Una volta ristabilito il quorum, C potrà essere riavviato. Una volta riavviato, C si ricongiungerà al dominio del servizio catalogo. Le applicazioni non riscontreranno alcun problema durante tutto questo periodo.
- Errore del centro dati: l'utente verifica che il centro dati sia effettivamente in errore e che sia stato isolato dalla rete. Quindi si immette un comando xsadmin per la sostituzione del quorum. I due centri dati ancora attivi effettueranno un recupero completo sostituendo i frammenti ospitati nel centro dati in errore. Il servizio catalogo è ora in esecuzione con un quorum completo di A e B. L'applicazione potrà riscontrare dei ritardi o delle eccezioni nell'intervallo tra l'inizio del black out e la sostituzione del quorum. Una volta sostituito il quorum, la griglia viene recuperata e si riprendono le normali operazioni.
- Recupero del centro dati: i centri dati ancora attivi sono già in esecuzione con un quorum sostituito. Quando si riavvia il centro dati che contiene C, si dovranno riavviare tutte le JVM in esso contenute. C si ricongiungerà al dominio del servizio catalogo esistente ed il quorum ritornerà quello della situazione normale senza interventi dell'utente.
- Errore del centro dati e brown out: il centro dati contenente C va in errore. Il quorum viene sostituito e ripristinato sui centri dati rimanenti. Se si verifica un brown out tra A e B, si applicano le normali regole di recupero dal brown out. Una volta finito il brown out, il quorum viene ristabilito e si eseguono le operazioni necessarie di recupero.

Comportamento del contenitore

Questa sezione descrive come si comportano le JVM del server contenitore durante la perdita ed il recupero del quorum.

I contenitori ospitano uno o più frammenti. I frammenti sono sia primari che repliche per una specifica partizione. Il servizio catalogo assegna i frammenti ad un contenitore ed esso onorerà questo compito fino a quando non riceverà nuove istruzioni dal servizio catalogo. Questo significa che se un frammento primario non riesce a comunicare con una replica del frammento a causa di un brown out, esso continuerà a provare finché non riceverà nuove istruzioni dal servizio catalogo.

Se si verifica un brown out della rete ed un frammento primario perde la comunicazione con la replica, esso proverà a ricollegarsi finché il servizio catalogo non gli darà nuove istruzioni.

Comportamento della replica sincrona

Durante l'interruzione della connessione il primario potrà accettare nuove transazioni finché vi siano almeno tante repliche in linea quante sono quelle riportate nella proprietà minsync per la serie di mappe. Se si elaborano nuove transazioni sul primario mentre il collegamento alla replica sincrona è interrotto, la replica verrà cancellata e sincronizzata nuovamente con lo stato attuale del primario nel momento in cui il collegamento verrà ristabilito.

Si sconsiglia l'uso di repliche sincrone tra centri dati oppure su collegamenti via WAN.

Comportamento della replica asincrona

Mentre la connessione è interrotta il primario può accettare nuove transazioni. Il primario memorizzerà nel buffer le modifiche fino ad un limite. Se si ristabilisce la connessione prima del raggiungimento del limite, la replica verrà aggiornata con le modifiche memorizzate nel buffer. Se si raggiunge il limite, il primario distruggerà l'elenco memorizzato nel buffer ed al ricollegamento della replica essa verrà ripulita e risincronizzata.

Comportamento del client

I client sono sempre in grado di collegarsi al server di catalogo per avviarsi nella griglia indipendentemente dal fatto che il dominio del servizio catalogo abbia il quorum o meno. Il client proverà a collegarsi a qualsiasi istanza del server di catalogo per ottenere la tabella di instradamento e quindi interagire con la griglia. La connettività della rete potrebbe impedire al client di interagire con alcune porzioni a causa dell'impostazione della rete. Il client può collegarsi alle repliche locali dei dati remoti se è stato configurato per fare ciò. I client non potranno aggiornare un dato se la partizione primaria di tale dato non è disponibile.

Comandi quorum con xsadmin

Questa sezione descrive i comandi xsadmin utili per le situazioni di quorum.

Esecuzioni di query per lo stato del quorum

Lo stato del quorum di un'istanza del server di catalogo può essere interrogato utilizzando il comando xsadmin.

```
xsadmin –ch cathost –p 1099 –quorumstatus
```

Vi sono cinque possibili risultati.

- Il quorum è disabilitato: i server di catalogo sono in esecuzione con il quorum disabilitato. Questo stato indica una modalità di sviluppo oppure con centro dati a server singolo. Non è raccomandata per le configurazioni con più centri dati.
- Il quorum è abilitato ed il server di catalogo ha il quorum: il quorum è abilitato ed il sistema funziona normalmente.
- Il quorum è abilitato ma il server di catalogo è in attesa del quorum: il quorum è abilitato ed è stato perso.
- Il quorum è abilitato ed è stato sostituito: il quorum è abilitato ed è stato sostituito.
- Stato del quorum non valido: quando si verifica un brown out, il servizio catalogo viene suddiviso in due partizioni, A e B. Il server di catalogo A sostituisce il quorum. Quando si risolve la partizione di rete, il server sulla partizione B non sarà valido e richiederà il riavvio della JVM. Si verifica anche se la JVM catalogo in B si riavvia durante un brown out e poi questo termina.

Sostituzione del quorum

Il comando xsadmin può essere utilizzato per sostituire il quorum. È possibile utilizzare una qualsiasi istanza del server di catalogo ancora attiva. Tutte le istanze

ancora attive vengono informate quando ad una di esse viene richiesto di sostituire il quorum. La sintassi per questa operazione è la seguente.

```
xsadmin –ch cathost –p 1099 –overridequorum
```

Comandi di diagnostica

- Stato del quorum: come descritto nella sezione precedente.
- Elenco gruppi principali: visualizza un elenco di tutti i gruppi principali. Vengono visualizzati i membri ed i leader dei gruppi principali.

  ```
  xsadmin –ch cathost –p 1099 –coregroups
  ```
- Arresto server: questo comando rimuove manualmente un server dalla griglia. Non è solitamente necessario poiché i server vengono rimossi automaticamente quando si rileva che sono in errore, ma il comando viene fornito per essere utilizzato con l'aiuto dell'assistenza IBM.

  ```
  xsadmin –ch cathost –p 1099 –g Grid –teardown server1,server2,server3
  ```
- Visualizzazione tabella di instradamento: questo comando visualizza la tabella di instradamento corrente simulando la connessione di un nuovo client alla griglia. Esso convalida inoltre la tabella di instradamento confermando che tutti i server contenitore riconoscono il proprio ruolo nella tabella di instradamento, come ad esempio che tipo di frammento per quale partizione.

  ```
  xsadmin –ch cathost –p 1099 –g myGrid -routetable
  ```
- Visualizzazione dei frammenti non assegnati: se alcuni frammenti non possono essere posizionati sulla griglia sarà possibile utilizzare questo comando per elencarli. Ciò si verifica solo quando il servizio di posizionamento ha una restrizione che previene il posizionamento. Ad esempio, se durante la modalità di produzione, si avviano delle JVM su una singola unità fisica, si potranno posizionare solo i frammenti primari. Quelli di replica non verranno assegnati finché le JVM non saranno avviate su una seconda unità. Il servizio di posizionamento, posiziona le repliche su JVM con indirizzi IP diversi dalle JVM che ospitano i primari. Se non si hanno JVM in una zona, alcuni frammenti potrebbero non essere assegnati.

  ```
  xsadmin –ch cathost –p 1099 –g myGrid –unassigned
  ```
- Impostazioni Set trace: questa serie di comandi definisce le impostazioni di traccia di tutte le JVM che soddisfano il filtro specificato per il comando xsadmin. Questa impostazione modifica le impostazioni di traccia solo finché non si utilizza un altro comando o finché le JVM modificate non vanno in errore o non si arrestano.

  ```
  xsadmin –ch cathost –p 1099 –g myGrid –fh host1 –settracespec
  ObjectGrid*=event=enabled
  ```

  Questo comando abilita la traccia per tutte le JVM su un'unità con il nome host specificato, in questo caso host1.
- Verifica delle dimensioni della mappa: il comando mapsizes è utile per verificare che la distribuzione delle chiavi sia uniforme sui frammenti nelle chiavi. Se alcuni contenitori hanno molte più chiavi rispetto ad altri è probabile che la funzione hash sugli oggetti chiave abbia una scarsa distribuzione.

  ```
  xsadmin -ch cathost -p 1099 -g myGrid -m myMapSet -mapsizes myMap
  ```

Considerazioni sulla sicurezza del trasporto

Poiché i centri dati sono solitamente distribuiti in ubicazioni geograficamente differenti, per motivi di sicurezza gli utenti potranno decidere di abilitare la sicurezza del trasporto tra i centri dati.

Per informazioni su TLS (Transport Layer Security) consultare *Guida alla gestione*.

Repliche e frammenti

Con eXtreme Scale è possibile replicare un database in memoria o un frammento da una JVM (Java virtual machine) ad un'altra. Un frammento rappresenta una partizione posizionata su un contenitore. Su un unico contenitore possono coesistere diversi frammenti che rappresentano diverse partizioni. Ciascuna partizione ha un'istanza che rappresenta un frammento primario ed un numero configurabile di frammenti di replica. I frammenti di replica possono essere sincroni o asincroni. I tipi di frammenti di replica ed il loro posizionamento sono determinati da eXtreme Scale con una politica di distribuzione, che definisce il numero minimo e massimo di frammenti sincroni ed asincroni.

Tipi di frammento

La replica utilizza tre tipi di frammenti:

- Primario
- Replica sincrona
- Replica asincrona

Il frammento primario riceve tutte le operazioni di inserimento, aggiornamento e rimozione. Esso aggiunge e rimuove le repliche, replica i dati sui frammenti di replica e gestisce le transazioni di commit e rollback.

Le repliche sincrone mantengono lo stesso stato dell'elemento primario. Quando un frammento primario replica i dati su una replica sincrona, non si esegue il commit della transazione finché esso non sia stato effettuato sulla replica sincrona.

Le repliche asincrone potrebbero trovarsi o meno allo stesso stato del frammento primario. Quando un frammento primario replica i dati su una replica asincrona, esso non ne attende il commit.

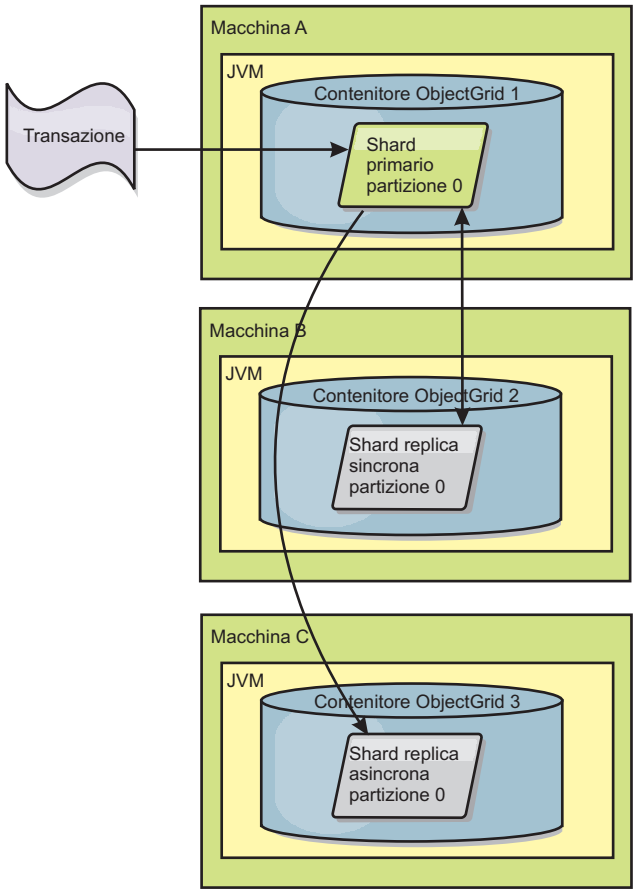


Figura 34. Percorso di comunicazione tra frammenti primari e di replica

Numero minimo di frammenti della replica sincrona

Quando un frammento primario si prepara ad eseguire il commit dei dati, verifica quanti frammenti di replica hanno votato per il commit della transazione. Se la transazione è normalmente in esecuzione sulla replica, esso vota per il commit. Se qualcosa non ha funzionato sulla replica sincrona, esso non vota per il commit. Prima dell'esecuzione del commit del primario, il numero di frammenti della replica sincrona che votano per il commit deve essere almeno uguale a quello delle impostazioni di minSyncReplica della politica di distribuzione. Quando il numero di frammenti della replica sincrona che votano per il commit è basso, il frammento primario non esegue il commit della transazione e ne consegue un errore. Questa azione assicura che il numero richiesto di repliche sincrone sia disponibile con i dati corretti. Le repliche sincrone che hanno ricevuto errori si registreranno nuovamente per correggere il proprio stato. Per ulteriori informazioni su come effettuare nuovamente la registrazione, consultare Ripristino del frammento di replica.

Il frammento primario genera un errore ReplicationVotedToRollbackTransactionException quando le repliche sincrone, ad aver votato per il commit, sono troppo poche.

Replica e programmi di caricamento

Solitamente, un frammento primario scrive le modifiche in modo sincrono su un database tramite il programma di caricamento. Il programma di caricamento ed il

database sono sempre sincronizzati. Quando un frammento primario va in failover su un frammento di replica, il database ed il programma di caricamento potrebbero non essere più sincronizzati. Ad esempio:

- Il frammento primario potrebbe inviare la transazione alla replica e quindi andare in errore prima del commit sul database.
- Il frammento primario potrebbe eseguire il commit sul database e quindi andare in errore prima dell'invio alla replica.

Entrambi gli approcci conseguono in una replica che sarà avanti o indietro di una transazione rispetto al database. Questa situazione non è accettabile. Il programma eXtreme Scale utilizza un protocollo speciale ed un contratto con l'implementazione del programma di caricamento per risolvere questo problema senza un commit in due fasi. Il protocollo viene riportato di seguito:

Lato frammento primario

- Inviare la transazione assieme ai risultati della transazione precedente.
- Scrivere sul database e tentare il commit della transazione.
- Se il database effettua il commit, eseguire il commit su eXtreme Scale. Se il database non esegue il commit, effettuare il rollback della transazione.
- Registrare i risultati.

Lato replica

- Ricevere la transazione e memorizzarla nel buffer.
- Per tutti i risultati, inviarli con la transazione, eseguire il commit delle transazioni memorizzate nel buffer e cancellare qualsiasi transazione di cui si è eseguito il rollback.

Lato replica in failover

- Per tutte le transazioni registrate in memoria, fornire le transazioni al programma di caricamento che tenterà di effettuarne il commit.
- È necessario scrivere il programma di caricamento in modo da rendere ciascuna transazione idempotente.
- Se la transazione è già nel database, il programma di caricamento non effettuerà alcuna operazione.
- Se la transazione non è nel database, il programma di caricamento applicherà la transazione.
- Dopo l'elaborazione di tutte le transazioni, il nuovo frammento primario potrà cominciare a soddisfare le richieste.

Questo protocollo assicura che lo stato del database sia allo stesso livello del frammento primario.

Allocazione dei frammenti (shard): primario e replica

Il servizio catalogo è responsabile del posizionamento dei frammenti. Ciascun ObjectGrid ha un certo numero di partizioni, ognuna delle quali ha un frammento primario e una serie facoltativa di frammenti di replica. Il servizio catalogo non colloca frammenti primari e di replica per la stessa partizione sullo stesso contenitore. Esso, inoltre, non colloca frammenti primari e di replica su contenitori che hanno lo stesso indirizzo IP (a meno che la configurazione non sia in modalità di sviluppo). Il servizio catalogo assegna i frammenti bilanciandoli in modo che vengano distribuiti equamente sui contenitori disponibili.

Se viene avviato un nuovo contenitore, eXtreme Scale recupera dai contenitori relativamente sovraccarichi i frammenti per il nuovo contenitore vuoto. Con questo comportamento, eXtreme Scale stabilisce e gestisce la propria essenziale flessibilità. Tale flessibilità è evidente nella sua efficace capacità di scalabilità orizzontale, sia per l'aumento di scala (scale out) sia per la riduzione di scala (scale in).

Aumento di scala (Scale out)

Aumentare di scala significa che quando ulteriori macchine Java virtual machine o contenitori vengono aggiunti ad una griglia eXtreme Scale, eXtreme Scale cerca di spostare i frammenti esistenti, primari o repliche, dalla vecchia serie di macchine JVM a quella nuova. Tale spostamento consente alla griglia di espandersi per sfruttare processore, rete e memoria delle macchine JVM appena aggiunte. Questo spostamento, inoltre, bilancia la griglia e cerca di fare in modo che ciascuna macchina JVM in essa presente ospiti la stessa quantità di dati. Quando la griglia si espande, ciascun server ospita una più piccola serie secondaria della griglia totale. eXtreme Scale presume che i dati siano distribuiti equamente tra le partizioni. Questa espansione abilita l'aumento di scala (scale out).

Riduzione di scala (Scale in)

Ridurre di scala significa che se una macchina JVM è in errore, eXtreme Scale cerca di riparare i danni. Se la macchina JVM in errore disponeva di una replica, eXtreme Scale sostituisce la replica perduta creandone una nuova su una macchina JVM ancora attiva. Se la macchina JVM disponeva di un frammento primario, eXtreme Scale cerca la replica migliore sulle macchine ancora attive e la promuove come nuovo frammento primario. eXtreme Scale sostituisce poi la replica promossa con la nuova replica creata sui server rimanenti. Per gestire la scalabilità, eXtreme Scale conserva il conteggio delle repliche per le partizioni nel caso di errori relativi ai server.

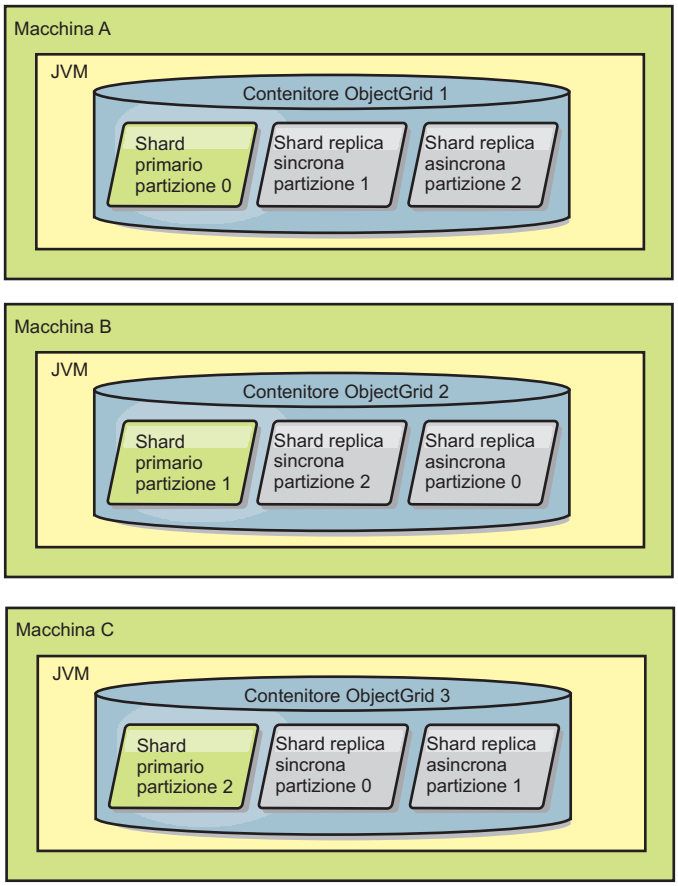


Figura 35. Posizionamento di una mapSet ObjectGrid con una politica di distribuzione di 3 partizioni con il valore 1 per minSyncReplicas, il valore 1 per maxSyncReplicas e il valore 1 per maxAsyncReplicas.

Lettura dalle repliche

È possibile configurare le serie di mappe in modo che un client sia autorizzato a leggere da una replica piuttosto che essere legato unicamente ai frammenti primari.

Di solito risulta vantaggioso utilizzare le repliche come più che semplici potenziali frammenti primari in caso di errori. Ad esempio, le serie di mappe possono essere configurate per consentire l'instradamento delle operazioni di lettura sulle repliche impostando l'opzione replicaReadEnabled sulla MapSet su true. L'impostazione predefinita è false.

Per ulteriori informazioni sull'elemento MapSet, consultare l'argomento relativo al file XML descrittore della politica di distribuzione in *Guida alla gestione*.

L'abilitazione della lettura delle repliche può migliorare le prestazioni distribuendo le richieste di lettura su più JVM (Java™ virtual machine). Se l'opzione non viene abilitata, tutte le richieste di lettura, quali i metodi ObjectMap.get o Query.getResultIterator, vengono instradate sul primario. Quando si imposta replicaReadEnabled su true, alcune richieste get potrebbero restituire dati obsoleti, ciò significa che l'applicazione che utilizza questa opzione deve essere in grado di poter tollerare quest'evenienza. Tuttavia non si verificherà una mancata corrispondenza nella cache. Se i dati non sono sulla replica, la richiesta get viene reindirizzata al primario e nuovamente provata.

L'opzione replicaReadEnabled può essere utilizzata sia con repliche sincrone che asincrone.

Bilanciamento del carico tra repliche

Il bilanciamento del carico tra le repliche viene solitamente utilizzato quando i client memorizzano nella cache dati che vengono frequentemente modificati oppure quando questi utilizzano il blocco pessimistico.

eXtreme Scale, se non altrimenti configurato, invia tutte le richieste di lettura e scrittura sul server primario di un dato gruppo di replica. Il primario deve servire tutte le richieste provenienti dai client. Potrebbe essere consigliabile inviare le richieste di lettura sulle repliche del primario. In questo modo si condividerà il carico delle richieste di lettura tra più JVM (Java Virtual Machine). Tuttavia, l'utilizzo delle repliche per le richieste di lettura potrebbe generare delle risposte incongruenti.

Il bilanciamento del carico tra le repliche viene solitamente utilizzato quando i client memorizzano nella cache dati che vengono frequentemente modificati oppure quando questi utilizzano il blocco pessimistico.

Se i dati vengono cambiati in continuazione e quindi invalidati nelle cache locali dei client, come risultato i primari rileveranno da parte dei client una frequenza di richieste get relativamente alta. Allo stesso modo, nella modalità di blocco pessimistico, non esiste alcuna cache locale, quindi tutte le richieste saranno inviate al primario.

Se i dati sono relativamente statici oppure se non si utilizza la modalità pessimistica, l'invio delle richieste di lettura non avrà un grande impatto sulle prestazioni. La frequenza delle richieste get da parte di client con cache piene di dati non è alta.

Quando un client parte per la prima volta, la sua cache locale è vuota. Le richieste di cache ad una cache vuota vengono inoltrate al primario. Il client con il passar del tempo riceve dati, diminuendo così il carico delle richieste. Se viene avviato contemporaneamente un gran numero di client, il carico potrebbe risultare significativo e quindi l'opzione della lettura della replica potrebbe rivelarsi una scelta appropriata per le prestazioni.

Eventi del ciclo di vita, di ripristino e di errore

I frammenti passano attraverso diversi stati ed eventi per supportare la replica. Il ciclo di vita di un frammento include il passaggio in linea, il runtime, la chiusura, il failover e la gestione degli errori. I frammenti possono essere promossi da un frammento di replica ad un frammento primario per gestire le modifiche dello stato del server.

Eventi del ciclo di vita

Quando i frammenti primario e di replica vengono posizionati ed avviati, passano attraverso una serie di eventi per passare in linea ed alla modalità di ascolto.

Frammento primario

Il servizio catalogo posiziona un frammento primario per una partizione. Inoltre, bilancia le ubicazioni del frammento primario ed avvia il failover per i frammenti primari.

Quando un frammento diventa primario, riceve un elenco di repliche dal servizio catalogo. Il nuovo frammento primario crea un gruppo di repliche e registra tutte le repliche.

Quando il frammento primario è pronto, viene inserito un messaggio di apertura per l'azienda nel file `SystemOut.log` del contenitore su cui il frammento è in esecuzione. Il messaggio di apertura, o messaggio CWOBJ1511I, indica il nome della mappa, il nome della serie di mappe ed il numero di partizione del frammento primario avviato.

```
CWOBJ1511I: mapName:mapSetName:partitionNumber (primario) è aperto per l'azienda.
```

Consultare “Allocazione dei frammenti (shard): primario e replica” a pagina 131 per ulteriori informazioni relative al modo in cui il servizio catalogo posiziona i frammenti.

Frammento di replica

I frammenti di replica sono principalmente controllati dal frammento principale, a meno che il frammento di replica non rilevi un problema. Durante un normale ciclo di vita, il frammento principale posizione, registra ed annulla la registrazione di un frammento di replica.

Quando il frammento primario inizializza un frammento di replica, un messaggio visualizza il log che descrive il punto in cui viene eseguita la replica per indicare che il frammento di replica è disponibile. Il messaggio di apertura, o messaggio or the CWOBJ1511I, indica il nome della mappa, il nome della serie di mappe ed il numero di partizione del frammento di replica. Tale messaggio è riportato di seguito:

```
CWOBJ1511I: mapName:mapSetName:partitionNumber (replica sincrona) è aperto per l'azienda.
```

oppure

```
CWOBJ1511I: mapName:mapSetName:partitionNumber (replica asincrona) è aperto per l'azienda.
```

Frammento di replica asincrono: un frammento di replica asincrono esegue il poll sul primario alla ricerca dei dati. La replica regolerà la tempificazione del poll in modo automatico se non riceve più dati dal primario, il che indica che essa ha raggiunto lo stesso livello del primario. Essa la regolerà inoltre anche se riceve un errore che indica che il primario è in errore o se si verifica un errore di rete.

All'avvio della replica asincrona, il seguente messaggio verrà inserito nel file SystemOut.log per la replica. Questo messaggio potrà essere inserito più di una volta per ciascun messaggio CWOBJ1511. Verrà nuovamente inserito se la replica si collega ad un primario diverso o se vengono aggiunte le mappe del template.

```
CWOBJ1543I: La replica asincrona objectGridName:mapsetName:partitionNumber ha avviato
o continuato ad eseguire la replica dall'elemento primario. Replica per mappe: [mapName]
```

Frammento di replica sincrono: quando il frammento di replica viene avviato per la prima volta, non è ancora in modalità peer. Quando frammento di replica è in modalità peer, riceve i dati dal frammento primario appena i dati arrivano al frammento primario. Prima di passare alla modalità peer, il frammento di replica necessita di una copia di tutti i dati esistenti sul frammento primario.

La replica sincrona copia i dati dal frammento primario in modo simile ad una replica asincrona eseguendo il poll dei dati. Quando essa copia i dati esistenti dal primario passa alla modalità peer ed inizia a ricevere i dati nello stesso momento in cui li riceve il primario.

Quando un frammento di replica passa alla modalità peer, inserisce un messaggio nel file SystemOut.log per la replica. Il tempo fa riferimento alla quantità di tempo richiesta dal frammento di replica per ottenere tutti i propri dati iniziali dal frammento primario. Il tempo può essere indicato con il valore zero o un valore molto basso se il frammento principale non dispone di dati da replicare. Questo messaggio potrà essere inserito più di una volta per ciascun messaggio CWOBJ1511. Verrà nuovamente inserito se la replica si collega ad un primario diverso o se vengono aggiunte le mappe del template.

```
CWOBJ1526I: La replica objectGridName:mapsetName:partitionNumber:mapName entrerà in modalità peer
dopo X secondi.
```

Quando il frammento di replica sincrona è in modalità peer, il frammento primario dovrà effettuare la replica delle transazioni a tutte le repliche sincrone in modalità peer. Il frammento di replica sincrona resta allo stesso livello dei dati del frammento primario. Se nella politica di distribuzione è impostato un numero minimo di repliche sincrone o minSync, tale numero di repliche sincrone dovrà votare per il commit prima che la transazione possa effettuare il commit sul primario con esito positivo.

Eventi di ripristino

La replica è progettata per il ripristino da malfunzionamenti ed eventi di errore. in caso di malfunzionamento di un frammento primario, tale frammento viene sostituito da un'altra replica. Se sui frammenti di replica sono presenti degli errori, il frammento di replica prova ad eseguire il ripristino. Il servizio catalogo controlla l'ubicazione e le transazioni dei nuovi frammenti primari o dei nuovi frammenti di replica.

Il frammento di replica diventa un frammento primario

Un frammento di replica diventa un frammento primario per due motivi. Si è verificato un malfunzionamento o l'arreso del frammento primario oppure è stata effettuata una decisione di bilanciamento per spostare il frammento primario precedente in un nuova ubicazione.

Il servizio catalogo seleziona un nuovo frammento primario dai frammenti di replica sincrona esistenti. Se si dovrà eseguire lo spostamento di un frammento primario e non esistono repliche, si creerà una replica temporanea per completare la transazione. Il nuovo frammento principale registra tutte le repliche esistenti ed accetta le transazioni come nuovo frammento primario. Se i frammenti di replica esistenti dispongono del livello di dati corretto, i dati correnti vengono conservati mentre il frammento di replica esegue la registrazione con il nuovo frammento primario. Le repliche asincrone eseguiranno il poll sul nuovo frammento primario.

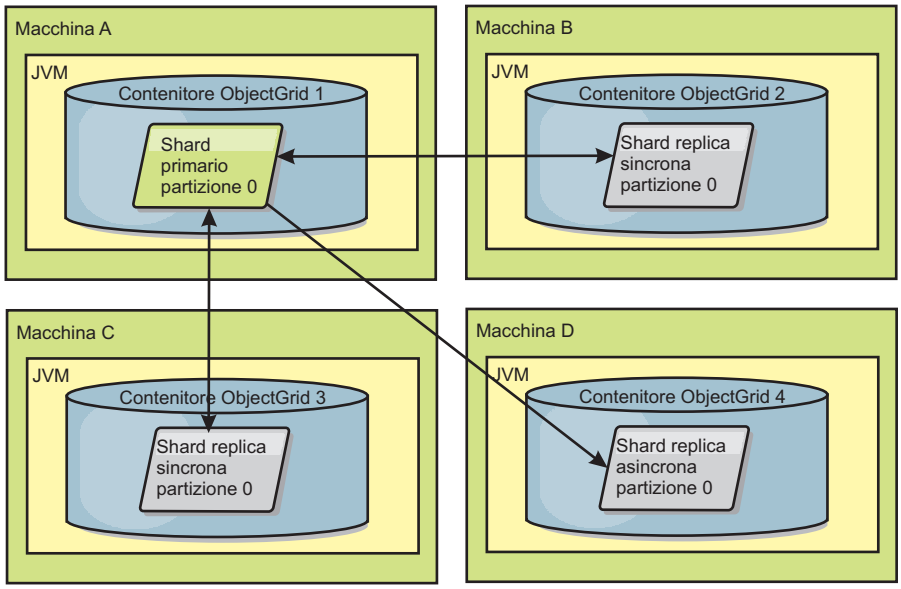


Figura 36. Esempio di posizionamento di una serie di mappe ObjectGrid per la partizione partition0. La politica di distribuzione ha il valore minSyncReplicas uguale a 1, il valore maxSyncReplicas uguale a 2 ed il valore maxAsyncReplicas uguale a 1.

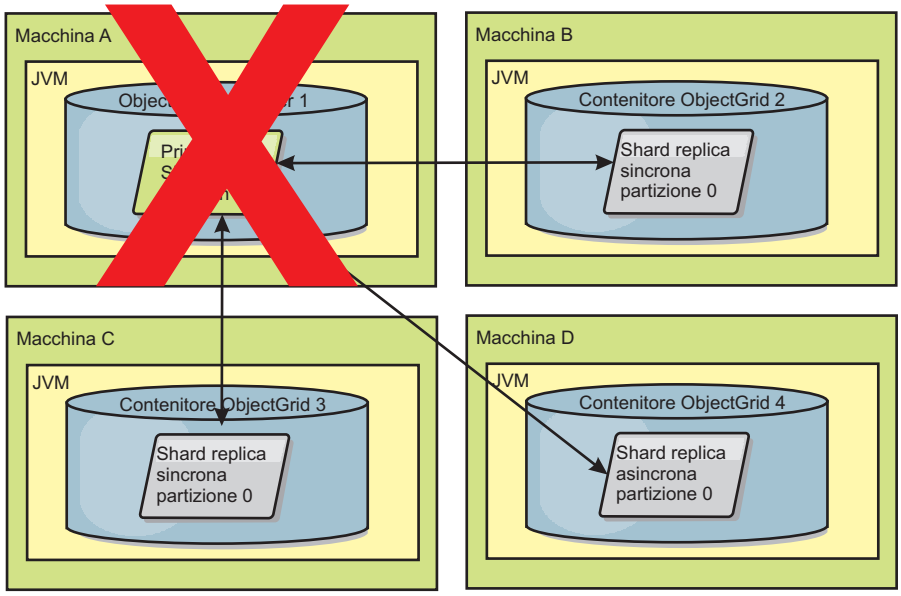


Figura 37. Malfunzionamento del contenitore per il frammento primario

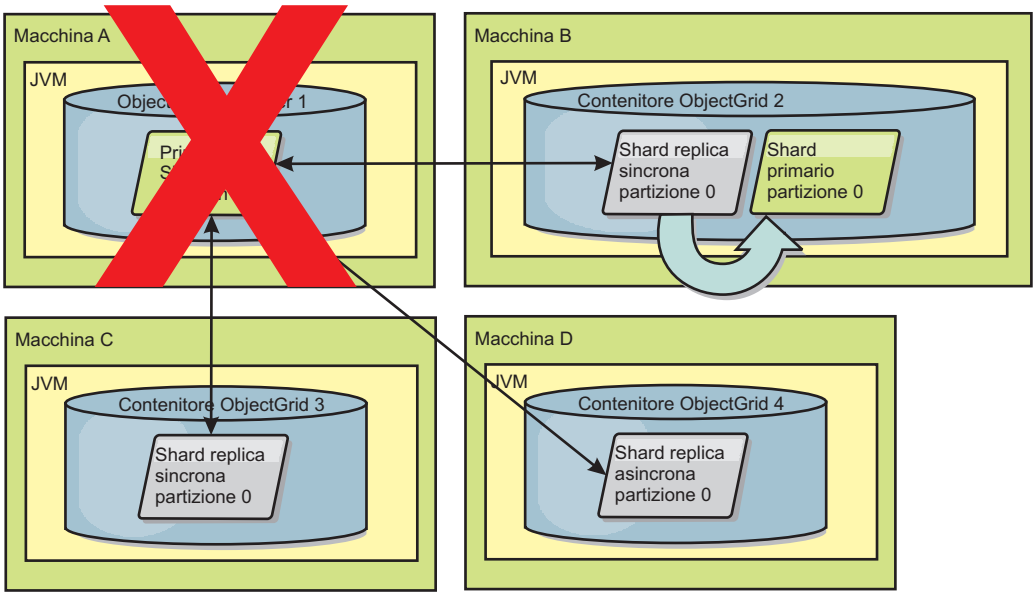


Figura 38. Il frammento di replica sincrona sul contenitore 2 ObjectGrid diventa il frammento primario

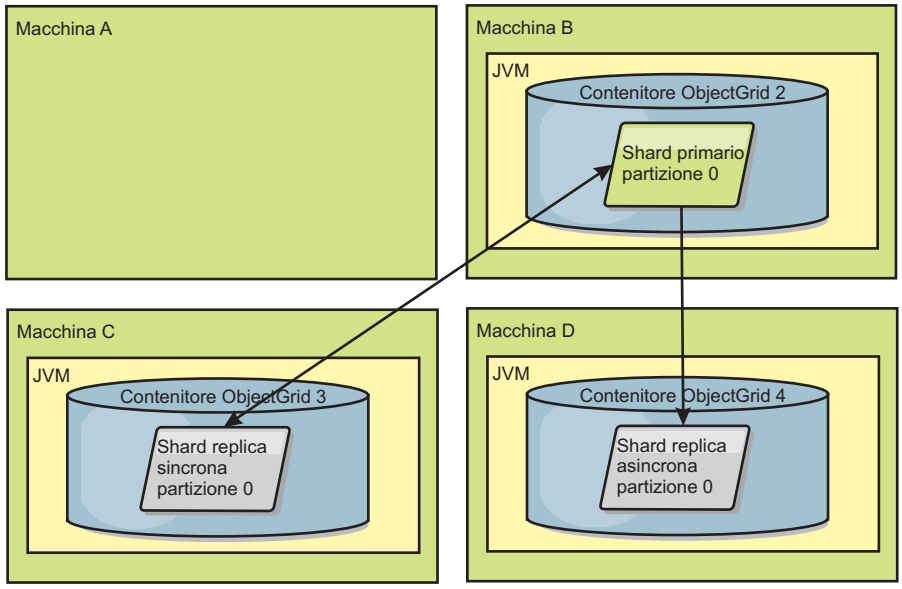


Figura 39. La macchina B contiene il frammento primario. In base al modo in cui è impostata la modalità di correzione automatica ed alla disponibilità dei contenitori, su una macchina potrebbe essere posizionato un nuovo frammento di replica sincrona.

Ripristino del frammento di replica

Un frammento di replica sincrono viene controllato dal frammento primario. Tuttavia, se un frammento di replica rileva un problema, può attivare un evento di nuova registrazione per correggere lo stato dei dati. La replica elimina i dati correnti e riceve una copia aggiornata dal frammento primario.

Quando un frammento di replica avvia un evento di nuova registrazione, la replica stampa un messaggio di log.

```
CWOBJ1524I: Il listener della replica
objectGridName:mapSetName:partition deve registrarsi di nuovo con l'elemento principale.
Motivo: Eccezione elencata
```

Se una transazione causa un errore su un frammento di replica durante l'elaborazione, il frammento di replica si trova in uno stato sconosciuto. La transazione è stata elaborata correttamente sul frammento primario, ma si è verificato un errore sulla replica. Per correggere tale situazione, la replica avvia un evento di nuova registrazione. Con una nuova copia dei dati dal frammento primario, il frammento di replica può continuare. Se si verifica nuovamente lo stesso problema, il frammento di replica non esegue continuamente la nuova registrazione. Per ulteriori dettagli, consultare “Eventi di errore”.

Eventi di errore

Una replica può interrompere la replica dei dati se rileva situazioni di errore da cui non può eseguire il ripristino.

Numero elevato di tentativi di registrazione

Se una replica attiva più volte una nuova registrazione senza eseguire correttamente il commit dei dati, la replica viene arrestata. L'arresto impedisce alla replica di attivare un loop di nuova registrazione infinito. Per impostazione predefinita, un frammento di replica prova ad eseguire la nuova registrazione per tre volte di seguito prima di arrestarsi.

Se un frammento di replica esegue un numero elevato di nuove registrazioni, nel log viene visualizzato il seguente messaggio.

```
CWOBJ1537E: objectGridName:mapSetName:partition ha superato il numero massimo
di volte per la riregistrazione (timesAllowed) senza transazioni corrette..
```

Se la replica non è in grado di eseguire il ripristino mediante la nuova registrazione, potrebbe essersi verificato un problema pervasivo per le transazioni relative al frammento di replica. Un problema possibile potrebbe essere la mancanza di risorse nel percorso di classe se si verifica un errore durante la deserializzazione delle chiavi o dei valori dalla transazione.

Errore durante il passaggio alla modalità peer

Se una replica prova a passare alla modalità peer e si verifica un errore durante l'elaborazione dei dati bulk esistenti dal frammento primario (dati del punto di controllo), la replica viene chiusa. La chiusura impedisce alla replica di essere avviata con dati iniziali non corretti. Poiché, in caso di nuova registrazione, la replica riceve gli stessi dati dal frammento primario, non esegue un ulteriore tentativo.

Se un frammento di replica non riesce a passare alla modalità peer, nel log viene visualizzato il seguente messaggio:

```
CWOBJ1527W La replica objectGridName:mapSetName:partition:mapName non è riuscita ad entrare in modalità peer dopo numerodisecondi secondi.
```

Nel log viene visualizzato un ulteriore messaggio che indica il motivo per cui la replica non è riuscita a passare alla modalità peer.

Errore di ripristino dopo la nuova registrazione o della modalità peer

Se una replica non riesce ad eseguire la nuova registrazione o di passare alla modalità peer, si trova in uno stato inattivo fino a quando non si verifica un nuovo

evento di posizionamento . Un nuovo evento di posizionamento potrebbe essere l'avvio o l'arresto di un nuovo server. È possibile avviare un evento di posizionamento anche utilizzando il metodo triggerPlacement su PlacementServiceMBean Mbean.

Instradamento a zone preferite

Con l'instradamento a zone preferite, eXtreme Scale può dirigere le transazioni su determinate zone in base alle specifiche dell'utente.

WebSphere eXtreme Scale consente di esercitare un controllo significativo su dove devono essere collocati i frammenti di una ObjectGrid.

L'instradamento alle zone preferite consente ai client eXtreme Scale di specificare una tendenza per una particolare zona o serie di zone. Quindi eXtreme Scale tenterà di instradare le transazioni del client alle zone preferite, prima di provare ad instradarle in qualsiasi altra zona.

Requisiti per l'instradamento alle zone preferite

Vi sono alcuni fattori da prendere in considerazione prima di tentare l'instradamento alle zone preferite. Assicurarsi che l'applicazione sia in grado di soddisfare i requisiti dello scenario.

Per poter usufruire dell'instradamento alle zone preferite, è necessario il posizionamento a partizione per contenitore. Questa strategia di posizionamento è una scelta valida per le applicazioni che memorizzano i dati della sessione nella ObjectGrid. La strategia di posizionamento della partizione predefinita per WebSphere eXtreme Scale è a partizione fissa. Le chiavi vengono codificate con hash al commit della transazione per determinare quale partizione ospita la coppia chiave-valore della mappa quando si utilizza un posizionamento a partizione fissa.

Il posizionamento a partizione per contenitore assegna i dati dell'utente ad una partizione casuale al momento del commit della transazione tramite il SessionHandle. È necessario essere in grado di ricreare il SessionHandle per poter richiamare i dati dall'ObjectGrid.

Poiché è possibile utilizzare le zone per un maggiore controllo sull'ubicazione in cui gli elementi primari e le relative repliche risiederanno nel dominio, una distribuzione su più zone è vantaggiosa quando i dati risiedono in più ubicazioni fisiche. La separazione geografica degli elementi primari e delle repliche è un modo per far sì che la perdita catastrofica di un centro dati non abbia conseguenze sulla disponibilità del dati.

Quando i dati vengono distribuiti in una topologia a più zone, è probabile che anche i client vengano distribuiti nella topologia. L'instradamento dei client alla relativa zona locale o centro dati locale ha l'ovvio vantaggio nelle prestazioni con una ridotta latenza di rete. Utilizzare questo scenario quando possibile.

Configurazione della topologia per l'instradamento alle zone preferite

Considerare il seguente scenario. Si hanno 2 centri dati: Chicago e Londra. Per poter ridurre al minimo i tempi di risposta dei client, si desidera che i client leggano e scrivano i dati nel relativo centro dati locale.

I frammenti primari ObjectGrid devono essere collocati in ciascun centro dati, in modo che le transazioni possano essere scritte localmente da ciascuna ubicazione. Inoltre, i client dovranno essere a conoscenza delle zone per poter effettuare l'instradamento alla zona locale.

Il posizionamento per contenitore individua i nuovi frammenti primari su ciascun contenitore avviato. Le repliche vengono posizionate in base alla zona e alle regole di posizionamento specificate dalla politica di distribuzione. Per impostazione predefinita, una replica viene posizionata in una zona diversa rispetto al relativo elemento primario. Considerare la seguente politica di distribuzione per il seguente scenario.

```
<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
 xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">
 <objectgridDeployment objectgridName="universe">
  <mapSet name="mapSet1" placementStrategy="PER_CONTAINER"
   numberOfPartitions="3" maxAsyncReplicas="1">
   <map ref="planet" />
  </mapSet>
 </objectgridDeployment>
</deploymentPolicy>
```

Ogni contenitore avviato con la politica di distribuzione riceverà 3 nuovi elementi primari. Ogni elemento primario avrà 1 replica asincrona. Avviare ogni contenitore con il nome zona appropriate. Utilizzare il parametro -zone se si stanno avviando i contenitori con lo script startOgServer.

Per un server di contenitori di Chicago:

- UNIX Linux

  ```
  startOgServer.sh s1 -objectGridFile ../xml/universeGrid.xml
  -deploymentPolicyFile ../xml/universeDp.xml
  -catalogServiceEndpoints MyServer1.company.com:2809
  -zone Chicago
  ```

- Windows

  ```
  startOgServer.bat s1 -objectGridFile ../xml/universeGrid.xml
  -deploymentPolicyFile ../xml/universeDp.xml
  -catalogServiceEndpoints MyServer1.company.com:2809
  -zone Chicago
  ```

Se i contenitori sono in esecuzione in WebSphere Application Server, è necessario creare un gruppo di nodi e denominarlo con il prefisso “ReplicationZone”. I server in esecuzione sui nodi di tali gruppi di nodi verranno collocati nella zona appropriata. Ad esempio, i server in esecuzione su un nodo di Chicago potrebbero essere un gruppo di nodi denominato “ReplicationZoneChicago”.

Gli elementi primari della zona Chicago avranno le repliche nella zona Londra. Gli elementi primari della zona Londra avranno le repliche nella zona Chicago.

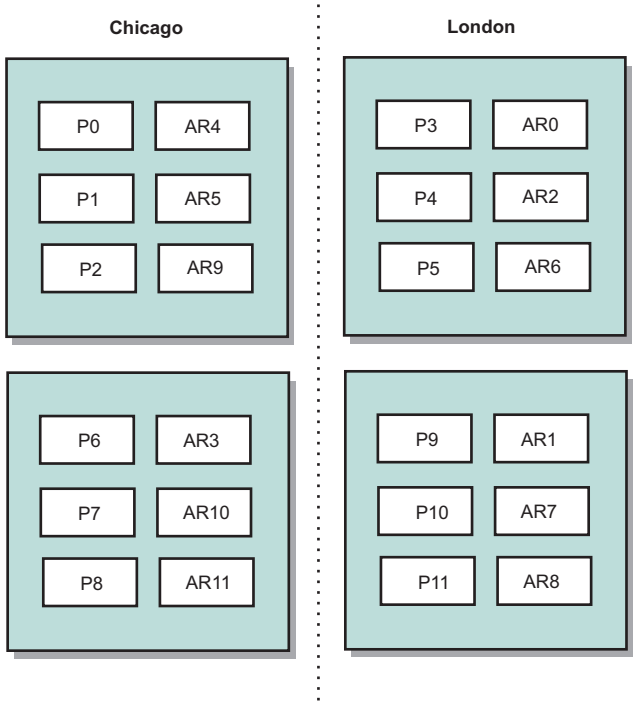


Figura 40. Elementi primari e repliche nelle zone

Impostare le zone preferite per i client. Questa informazione può essere fornita in uno dei differenti modi riportati di seguito. Il modo più semplice è di fornire un file delle proprietà client al client JVM. Creare un file denominato objectGridClient.properties ed assicurarsi che sia incluso nel percorso classe. Per ulteriori informazioni, consultare l'argomento relativo al file delle proprietà del client nel manuale *Guida alla gestione*.

Includere la proprietà preferZones nel file. Impostare il valore della proprietà sulla zona appropriata. I client in Chicago devono avere quanto segue nel file objectGridClient.properties.

```
preferZones=Chicago
```

Il file delle proprietà per i client di Londra deve contenere

```
preferZones=London
```

Questa proprietà indica ad ogni client di instradare le transazioni alla zona locale, se possibile. I dati inseriti in un elemento primario nella zona locale verranno replicati in modo asincrono nella zona esterna.

Utilizzo di SessionHandle per effettuare l'instradamento alla zona locale

La strategia di posizionamento per contenitore non utilizza un algoritmo basato su hash per determinare l'ubicazione delle coppie chiave-valore nell'ObjectGrid. L'ObjectGrid deve usufruire di SessionHandle per assicurare che le transazioni vengano instradate nell'ubicazione corretta quando si utilizza questa strategia di posizionamento. Quando una transazione ha effettuato il commit, un SessionHandle viene collegato alla sessione se non ne è stato già impostato uno. In alternativa, il SessionHandle può essere collegato alla sessione richiamando

Session.getSessionHandle prima di effettuare il commit della transazione. I seguenti frammenti di codice mostrano un SessionHandle che viene collegato prima di effettuare il commit della transazione.

```
Session ogSession = objectGrid.getSession();

// binding the SessionHandle
SessionHandle sessionHandle = ogSession.getSessionHandle();

ogSession.begin();
ObjectMap map = ogSession.getMap("planet");
map.insert("planet1", "mercury");

// tran is routed to partition specified by SessionHandle
ogSession.commit();
```

Si presupponga che il codice precedente fosse in esecuzione su un client nel centro dati di Chicago. Poiché l'attributo preferZones è stato impostato su Chicago per questo client, questa transazione verrebbe instradata su una delle partizioni primarie nella zona Chicago, partizione 0, 1, 2, 6, 7 o 8.

Questo SessionHandle è il percorso per tornare alla partizione che memorizza i dati sottoposti a commit. Il SessionHandle deve essere riutilizzato o ricreato e deve essere impostato sulla sessione, per poter tornare alla partizione che contiene i dati con commit.

```
ogSession.setSessionHandle(sessionHandle);
ogSession.begin();

// value returned will be "mercury"
String value = map.get("planet1");
ogSession.commit();
```

Poiché questa transazione riutilizza il SessionHandle tcreato durante la transazione insert, la transazione get verrà instradata alla partizione che contiene i dati inseriti. Questa transazione non sarà in grado di richiamare i dati inseriti precedentemente se il SessionHandle non è stato impostato.

In che modo gli errori di contenitore e di zona influiscono sull'instradamento basato sulle zone

In normali circostanze, un client con la proprietà preferZones impostata avrà tutte le transazioni instradate sulla zona o le zone specificate. Tuttavia, la perdita di un contenitore comporterà la promozione di una replica in elemento primario in una zona esterna. Un client che in precedenza veniva instradato sulle partizioni della zona locale potrebbe essere forzato all'instradamento sulla zona esterna. per poter richiamare i dati inseriti precedentemente.

Considerare il seguente scenario. Un contenitore nella zona Chicago viene perso. Precedentemente conteneva elementi primari per le partizioni 0, 1 e 2. I nuovi elementi primari per queste partizioni saranno nella zona Londra in quanto questa zona ospitava le repliche di quelle partizioni.

Tutti i client di Chicago che utilizzano un SessionHandle che punta ad una delle partizioni sottoposte a failover ora verranno instradati su Londra. I client di Chicago che utilizzano nuovi SessionHandle verranno instradati su elementi primari con base su Chicago.

Allo stesso modo, se si perde l'intera zona di Chicago, tutte le repliche contenute nella zona Londra diverranno elementi primari. In questo caso, tutti i client di Chicago instraderanno le proprie transazioni su Londra.

Topologie della replica della griglia multi-master (AP)

Utilizzando la funzione di replica asincrona multi-master, due o più griglie possono diventare una copia speculare dell'altra. Questo mirroring viene effettuato utilizzando repliche asincrone tra link che connettono insieme le griglie. Ciascuna griglia è ospitata all'interno di un "dominio" completamente indipendente che possiede un proprio servizio catalogo, server contenitori e un nome dominio univoco. Utilizzando la funzione di replica asincrona, è possibile utilizzare dei link per interconnettere una collezione di questi domini e successivamente sincronizzare i domini utilizzando la replica su questi link. eXtreme Scale abilita a costruire quasi qualsiasi topologia perché la definizione dei link tra i domini è lasciata all'utente.

Domini: Griglie con specifiche caratteristiche

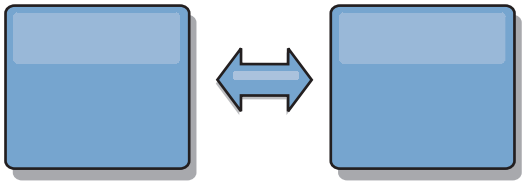
Le griglie utilizzate nelle topologie di replica multi-master sono conosciute come *domini*. Ciascun dominio deve avere le seguenti caratteristiche:

- deve avere un servizio catalogo privato con un nome dominio univoco
- deve avere lo stesso nome griglia delle altre griglie nel dominio
- deve avere lo stesso numero di partizioni delle altre griglie nel dominio
- Deve essere una griglia a FIXED_PARTITION (le griglie PER_CONTAINER non possono essere replicate)
- deve avere lo stesso numero di partizioni (potrebbe avere o meno lo stesso numero e gli stessi tipi di repliche)
- deve avere gli stessi tipi di dati che sono stati replicati dalle altre griglie nel dominio
- deve avere lo stesso nome mapset, i nomi mappe e i template della mappa dinamica delle altre griglie nel dominio.

Dopo aver avviato i domini nella topologia, verrà replicata ogni griglia avente le caratteristiche precedentemente indicate. Considerare che la relativa politica di replica sarà ignorata.

Link di connessione ai domini

Un'infrastruttura di griglia di replica è un grafico connesso di domini con link bidirezionali tra di essi. Un link consente due domini per scambiarsi le modifiche dei dati. Ad esempio, la topologia più semplice è una coppia di domini con un singolo link tra di essi. I domini sono denominati partendo dalla lettera “A” e poi proseguendo con la lettera “B” e così via procedendo da sinistra. Il link potrebbe incrociare una WAN (Wide area network) coprendo grandi distanze. Se il link viene interrotto, le modifiche ai dati possono ancora essere effettuate in entrambi i domini. Le modifiche vengono riconciliate successivamente, quando il link riconnette i domini. I link automaticamente tentano di eseguire nuovamente la connessione se la connessione di rete viene interrotta.

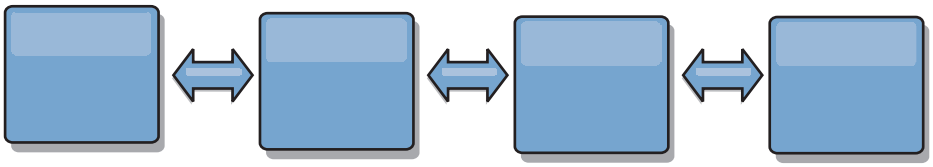


Dopo aver impostato i link eXtreme Scale tenterà prima di rendere ogni dominio identico e successivamente tenterà di mantenere le condizioni di identicità quando le modifiche avvengono in qualsiasi dominio. L'obiettivo di eXtreme Scale è per

ciascun dominio di essere un esatto specchio di ogni altro dominio connesso dai link. I link di replica tra i domini aiutano a garantire che qualsiasi modifica effettuata in un dominio venga copiata negli altri domini.

Topologie lineari

Sebbene è tra le più semplici topologie, un topologia lineare dimostra alcune qualità dei link. Prima, non è necessario per un dominio essere connesso direttamente ad ogni altro dominio per poter ricevere le modifiche. Il dominio B trarrà le modifiche dal dominio A. Il dominio C riceve le modifiche dal dominio A attraverso il dominio B che connette i domini A e C. Analogamente, il dominio D riceve le modifiche dagli altri domini attraverso il dominio C. Questa funzionalità diffonde il carico della distribuzione delle modifiche lontano dall'origine delle modifiche.



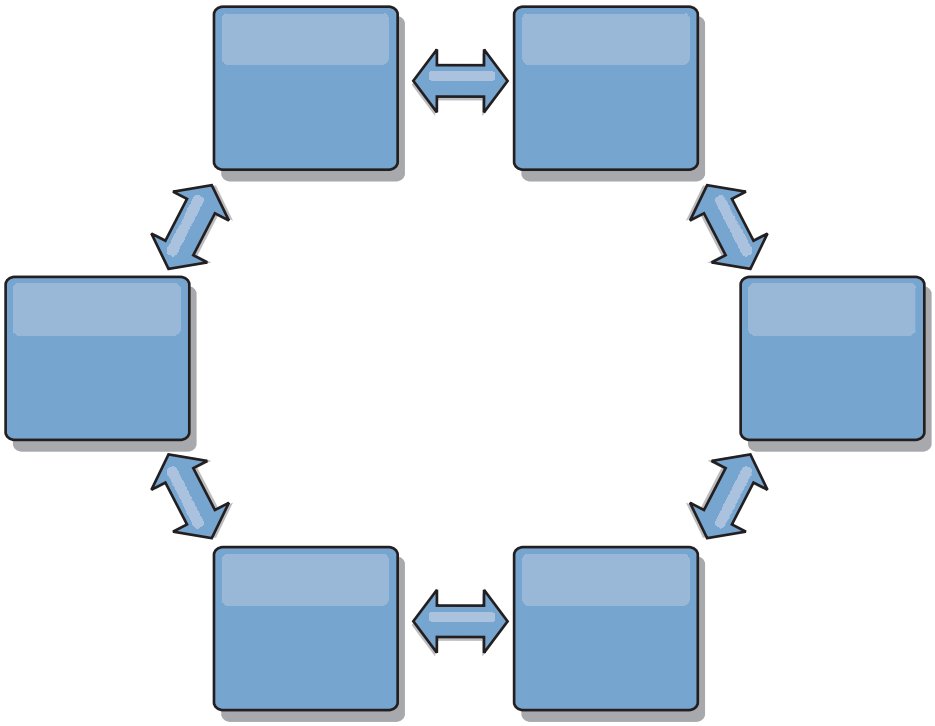
Considerare che se il dominio C è in errore, si verificano i seguenti eventi.

1. Il dominio D dovrebbe essere orfano fino a quando il dominio C non è riavviato.
2. Il dominio C dovrebbe sincronizzare se stesso con il dominio B, che è una copia del dominio A
3. Il dominio D dovrebbe utilizzare il dominio C per sincronizzare se stesso con le modifiche avvenute nei domini A e B mentre il dominio D è rimasto orfano (mentre il dominio C era giù)

Alla fine, i domini A, B, C e D dovrebbero diventare nuovamente tutti identici l'uno con gli altri.

Topologie ad anello

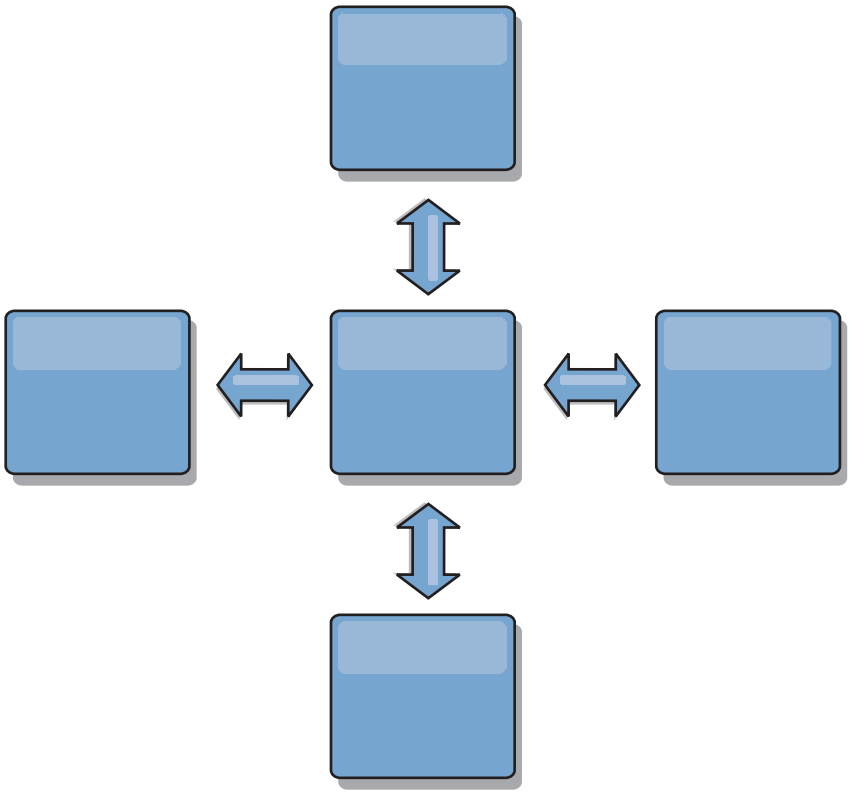
Le topologie ad anello sono un esempio di una topologia più adattabile. Sebbene un dominio o un singolo link può essere in errore, i domini superstiti possono ancora ricevere modifiche spostandosi intorno all'anello lontano dall'errore. Ciascun dominio dispone di due link agli altri domini. Ciascun dominio ha al massimo due link, non importa quanto grande sia la topologia ad anello. La latenza nel propagare le modifiche può essere considerevole poiché le modifiche da un particolare dominio potrebbero avere la necessità di spostarsi attraverso diversi domini prima che tutti i domini vedano tutte le modifiche. Una topologia lineare presenta lo stesso problema.



Immaginare una topologia ad anello più sofisticata con un dominio root al centro dell'anello. Il dominio root agisce come una clearing house centrale mentre gli altri domini agiscono come clearing house remote per le modifiche che si verificano nel dominio root. Il dominio root può arbitrare le modifiche tra i domini. Se una topologia ad anello contiene più di un anello intorno ad un dominio root, il dominio root può solo arbitrare le modifiche tra i domini nell'anello più interno. Tuttavia, i risultati dell'arbitraggio si estendono ai domini negli altri anelli.

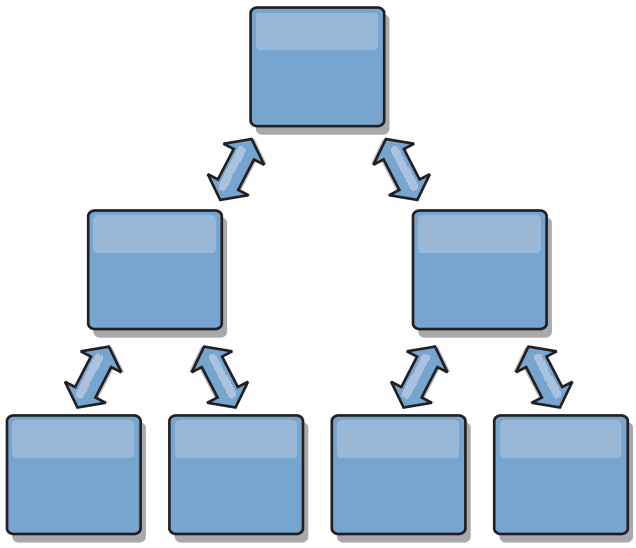
Topologie Hub e spoke

Una topologia hub e spoke ha una migliore latenza il che significa che le modifiche viaggiano attraverso al massimo un dominio intermedio (l'hub), ma presenta altri problemi. Essa ha un dominio centrale che agisce come un hub. Questo "dominio hub" è connesso ad ogni "dominio spoke" utilizzando un link. Chiaramente, l'onere della distribuzione delle modifiche tra i domini resta a carico dell'hub. L'hub agisce come una clearing house per i conflitti, un'impostazione che può essere importante in determinati scenari. In un ambiente con un elevato tasso di aggiornamento, l'hub, per essere adeguato, potrebbe avere la necessità di essere eseguito su più hardware rispetto agli spoke. eXtreme Scale è progettato per scalare in modo lineare, il che significa che è possibile ingrandire l'hub come richiesto senza difficoltà. Tuttavia, se l'hub va in errore, le modifiche non verranno distribuite fino a quando l'hub non viene riavviato. Alcune modifiche nei domini spoke verranno distribuite dopo che l'hub sia stato riconnesso.



Topologie a struttura ad albero

Un ultimo esempio di topologia è una struttura ad albero diretta non ciclica. Non ciclica significa che non esistono cicli o loop. Diretta significa che i link esistono solo tra parent e child. Questa configurazione può essere utile per topologie con così tanti domini che non è agevole avere un hub centrale connesso a ciascun spoke, o nei casi in cui si ha la necessità di avere la possibilità di aggiungere dei domini child senza aggiornare il dominio root.



Questa topologia può ancora avere una clearing house nel dominio root, ma il secondo livello può agire come clearing house remote per le modifiche che si verificano al di sotto del dominio stesso. Il dominio root può arbitrare le modifiche

tra i domini solo al secondo livello. Sono possibili anche strutture ad albero N-ary. Una struttura ad albero N-ari ha N child a ciascun livello. Ciascun dominio ha un'estensione di N.

Considerazioni sull'arbitraggio nella progettazione della topologia

I conflitti di modifica potrebbero verificarsi se gli stessi record possono essere modificati contemporaneamente in due posti. Impostare ciascun dominio in modo da avere circa la stessa quantità di CPU, memoria, risorse di rete. Si potrebbe osservare che i domini che eseguono la gestione del conflitto di modifica (arbitraggio) utilizzano più risorse degli altri domini. I conflitti vengono rilevati automaticamente. Essi vengono risolti utilizzando uno dei due meccanismi:

- **Arbitro del conflitto predefinito.** Il protocollo predefinito prevede di utilizzare le modifiche provenienti dal dominio con il nome lessicale più basso. Ad esempio, se il dominio A e il dominio B generano un conflitto per un record, le modifiche dal dominio B saranno ignorate. Il dominio A conserva la propria versione e il record nel dominio B viene modificato per corrispondere al record del dominio A. Questo si applica anche alle applicazioni in cui gli utenti o le sessioni sono normalmente legati o presentano affinità con una delle griglie.
- **Arbitro del conflitto personalizzato.** Le applicazioni possono fornire un arbitro personalizzato. Quando un dominio rileva un conflitto, esso richiama l'arbitro. Per informazioni relative allo sviluppo di un 'buon' arbitro personalizzato, consultare Sviluppo di arbitri personalizzati per la replica multi-master..

Per topologie in cui sono possibili i conflitti, considerare di utilizzare una topologia hub e spoke oppure una topologia a struttura ad albero. Queste due topologie sono idonee ad evitare continui continui il che può accadere quando:

1. più domini sperimentano un conflitto
2. ciascun dominio risolve il conflitto localmente, apportando delle correzioni
3. le correzioni entrano in conflitto producendo correzioni delle correzioni
4. e così via, poiché le correzioni si propagano tra i vari domini, tentando di realizzare la sincronia.

Per evitare continui conflitti, scegliere un dominio specifico – un *dominio di arbitraggio* – come gestore del conflitto per un sottoinsieme di domini. Ad esempio, una topologia hub e spoke potrebbe utilizzare l'hub come gestore del conflitto. Il gestore del conflitt spoke ignora qualsiasi conflitto rilevato dai domini spoke. Il dominio hub creerà le correzioni, evitando il controllo delle correzioni del conflitto. Il dominio designato a gestire i conflitti deve avere un link a tutti i domini per cui è responsabile di risolvere i conflitti. In una topologia a struttura ad albero, ogni dominio interno parent risolve i conflitti per i propri child diretti. Al contrario, se si utilizza una topologia ad anello, non è possibile designare un dominio nell'anello come dominio risolutore dei conflitti.

La tabella di seguito riportata riepiloga gli approcci di arbitraggio che sono maggiormente compatibili con le varie topologie. .

Tabella 14. Approcci di arbitraggio. Questa tabella stabilisce se l'arbitraggio dell'applicazione è compatibile con le varie tecnologie.

Topologia	Arbitraggio dell'applicazione	Note
Una linea di due domini	Sì	Scegliere un dominio come arbitro.

Tabella 14. Approcci di arbitraggio (Continua). Questa tabella stabilisce se l'arbitraggio dell'applicazione è compatibile con le varie tecnologie.

Topologia	Arbitraggio dell'applicazione	Note
Una linea di tre domini	Sì	Il dominio in mezzo deve essere l'arbitro. Pensare al dominio in mezzo come l'hub in una semplice topologia hub e spoke.
Una linea di più di tre domini	Sì	L'arbitraggio dell'applicazione non è supportato.
Un hub con N spoke	Sì	L'hub con i link a tutti gli spoke deve essere il dominio di arbitraggio.
Un anello di N domini	Sì	L'arbitraggio dell'applicazione non è supportato
Una struttura ad albero diretta, non ciclica (struttura ad albero N-ari)	Sì	Tutti i nodi root devono aribitrare solo i loro discendenti diretti.

Considerazioni sui link nella progettazione della topologia

Idealmente, una topologia comprende il minimo numero di link mentre l'ottimizzazione cerca un compromesso tra la latenza della modifica, la tolleranza di errori e le caratteristiche delle prestazioni.

- **Latenza di modifica**

  La latenza di modifica viene determinata dal numero di domini intermedi che una modifica deve attraversare prima di arrivare ad un dominio specifico.

  Una topologia ha la migliore latenza di modifica quando essa elimina i domini intermedi creando dei link per ciascun dominio ad ogni altro dominio. Tuttavia, un dominio deve eseguire l'attività di replica in proporzione al relativo numero di link. Per topologie larghe, il picco di link da definire potrebbe comportare un onere amministrativo.

  La velocità con cui una modifica viene copiata negli altri domini dipende da ulteriori fattori tra i quali:

  – CPU e larghezza di banda della rete nel dominio origine
  – il numero di domini intermedi e i link tra il dominio origine e il dominio di destinazione
  – la CPU e le risorse di rete disponibili nei domini origine, di destinazione e intermedi.

- **Tolleranza degli errori**

  La tolleranza degli errori è determinata dal numero di percorsi esistenti tra i due domini per la replica della modifica.

  Se esiste un singolo link tra i domini, se il link va in errore, le modifiche non verranno propagate. Se il singolo link da un dominio ad un altro dominio passa attraverso dei domini intermedi, le modifiche non verranno propagate se qualsiasi dominio intermedio è giù.

  Considerare la topologia lineare con tre domini A, B e C:

  ```
  A <-> B <-> C
  ```

  Se perdura qualsiasi di queste condizioni, il dominio C non vedrà le modifiche del dominio A:

  – Il dominio A è su e il dominio B è giù

– Il link tra A e B è giù
– Il link tra B e C è giù

Al contrario, una topologia ad anello consente a ciascun dominio di trarre le modifiche da altre direzioni.

```
A <-> B <-> C <-> back to A
```

Ad esempio, se il dominio B è giù, il dominio C ancora può trarre le modifiche direttamente dal dominio A.

Una progettazione hub e spoke è suscettibile di far andare giù l'hub poiché sono forzate a passare attraverso l'hub stesso. Tuttavia, è meritevole ricordare che un singolo dominio è ancora una griglia completamente tollerante agli errori grossolani che potrebbero presentarsi come problemi del centro dati fisico o WAN.

- **Prestazioni**

  Il numero di link definiti in un dominio influisce sulle prestazioni. Più link utilizzano più risorse e il risultato potrebbe essere un calo delle prestazioni. La possibilità di trarre le modifiche per un dominio A attraverso gli altri domini effettivamente riduce il carico del dominio A nelle repliche delle relative transazioni ovunque. *Il carico di distribuzione della modifica in un dominio è limitato al numero di link che esso utilizza. Esso non ha niente a che fare con quanti domini sono presenti nella topologia.* Questa proprietà fornisce la scalabilità e consente all'onere della distribuzione della modifica di essere condiviso tra i domini nella topologia, piuttosto che caricare l'onere su un singolo dominio.

  Un dominio può trarre le modifiche indirettamente attraverso altri domini. Considerare una topologia lineare con cinque domini.

  ```
  A <=> B <=> C <=> D <=> E
  ```

  – A trae le modifiche da B, C, D ed E attraverso B
  – B trae direttamente le modifiche da A e C e le modifiche da D ed E attraverso C.
  – C trae direttamente le modifiche da B e D e le modifiche da A attraverso B ed E attraverso D.
  – D trae direttamente le modifiche da C ed E e le modifiche da A e B attraverso C.
  – E trae direttamente le modifiche da D e le modifiche da A, B e C attraverso D.

  Il carico di distribuzione nei domini A ed E è più basso perché ciascuno di essi ha un solo link ad un singolo dominio. Il carico di distribuzione nei domini B, C, e D è doppio del carico nei domini A ed E perché ciascuno dei domini B, C e D hanno un link a due domini. Questa distribuzione di carichi potrebbe rimanere costante anche se la linea contenesse 1000 domini perché il carico dipende dal numero di link di ciascun dominio e non dal numero complessivo di domini nella topologia.

Considerazioni sulla prestazione

Considerare le limitazioni di seguito riportate quando si utilizzano le topologie di replica multi-master.

- **Ottimizzazione della distribuzione della modifica** (trattata in precedenza)
- **Latenza della replica** (trattata in precedenza)
- **Le prestazioni del link di replica** eXtreme Scalecreano un singolo socket TCP/IP tra ogni coppia di JVM. Tutto il traffico tra queste JVM si verifica sopra quel socket compreso la replica multi-master. Poiché i domini vengono ospitati

su almeno N JVM contenitori fornendo almeno N link TCP per domini peer, i domini con il maggior numero di contenitori avrà livelli di prestazioni della replica più elevati. Più contenitori significa più CPU e risorse di rete.

- **Il supporto dell'ottimizzazione di 'sliding window ' TCP e l'** Abilitazione di RFC 1323 su entrambi gli estremi di un link consente di avere più dati in andata e ritorno (round trip) risultando in un livello di prestazioni più elevato. La tecnica espande la capacità della finestra di un fattore pari a circa 16,000.

  Ricordarsi che i socket TCP utilizzano il meccanismo 'sliding window' per controllare il flusso di masse di dati il che tipicamente limita il socket a 64 KB per un intervallo di andata e ritorno (round trip). Se l'intervallo di andata e ritorno è 100 ms, la larghezza di banda è limitata a 640 KB al secondo senza ulteriore ottimizzazione. L'utilizzo completo della larghezza di banda disponibile su un link potrebbe richiedere un'ottimizzazione specifica in base al sistema operativo. La maggior parte di sistemi operativi prevede i parametri di ottimizzazione incluso le opzioni RFC 1323 per migliorare la velocità di trasmissione per i link ad elevata latenza.

  Diversi fattori possono influire sulle prestazioni della replica

  – La velocità a cui eXtreme Scale può trarre le modifiche.
  – La velocità a cui eXtreme Scale può soddisfare le richieste di replica.
  – La capacità 'sliding window'
  – L'ottimizzazione del buffer di rete su entrambi i lati di un link per consentire a eXtreme Scale di trarre le modifiche sopra i socket in modo più veloce possibile.
- **Serializzazione dell'oggetto** Tutti i dati devono essere serializzabili. Se un dominio non utilizza COPY_TO_BYTES, il dominio deve utilizzare la serializzazione Java oppure ObjectTransformers per ottimizzare le prestazioni della serializzazione.
- **Compressione** eXtreme Scale per impostazione predefinita, comprime tutti i dati inviati tra i domini. Non esiste un'opzione per disabilitare la compressione nell'attuale release.
- **Ottimizzazione della memoria** *L'utilizzo della memoria per una topologia di replica multi-master è largamente indipendente dal numero di domini nella topologia.*

  L'abilitazione della replica multi-master aggiunge un sovraccarico fisso per la voce della mappa per gestire il controllo di versioni. Ciascun contenitore tiene anche traccia di un quantitativo fisso di dati per ciascun dominio nella topologia. Una topologia con due domini utilizza approssimativamente la stessa memoria di una topologia con 50 domini. eXtreme Scale non utilizza log di ripetizione o code simili nella sua implementazione il che significa che se un link di replica non è disponibile per un intervallo considerevole di tempo, non si verifica la crescita della dimensione della struttura dati, piuttosto resta in attesa di riprendere la replica quando il link si riavvia.

Più centri dati con FIXED_PARTITION

Ora è possibile utilizzare una griglia FIXED_PARTITION tra due o più centri dati. Ciascun centro dati ha la necessità di avere il proprio dominio, in termini di replica multi-master. Ciascun centro dati può leggere e scrivere i dati rispetto al dominio locale. Queste modifiche si propagheranno agli altri centri dati utilizzando i link che sono stati definiti.

Client completamente replicati

Questa variazione nella topologia interessa una coppia di server eXtreme Scale che sono in esecuzione come hub. Ciascun client crea una singola griglia contenitore autonoma con un catalogo nella JVM client. Un client utilizza la propria griglia per connettersi al catalogo hub causando che il client si sincronizza con l'hub appena ottiene una connessione all'hub.

Alcune modifiche effettuate dal client sono locali al client e vengono replicate in modo asincrono all'hub. L'hub agisce come dominio di arbitraggio distribuendo le modifiche a tutti i client connessi. La topologia dei client completamente replicati fornisce una buona cache L2 per il programma di definizione relazionale dell'oggetto come OpenJPA. Le modifiche verranno distribuite velocemente tra le JVM client attraverso l'hub. Finché la dimensione della cache può essere contenuta entro lo spazio dell'heap disponibile dei client, questa topologia costituisce una buona architettura per questo stile L2.

Utilizzare, se necessario, più partizioni per scalare il dominio hub su più JVM. Poiché tutti i dati ancora devono adattarsi ad una singola JVM client , utilizzando più partizioni si incrementa la capacità dell'hub di distribuire ed arbitrare le modifiche, ma esso non modifica la capacità di un singolo dominio.

Limitazioni

Considerare le seguenti limitazioni quando si decide se e in che modo utilizzare le topologie di replica multi-master.

- **Considerare la configurazione dei programmi di caricamento della classe con più domini.**

  I domini devono avere accesso a tutte le classi che vengono utilizzate come chiavi e valori. Qualsiasi dipendenza deve essere riflessa in tutti i percorsi di classe per le JVM del contenitore della griglia per tutti i domini. Se un plug-in CollisionArbiter recupera il valore per una voce di cache, le classi per i valori devono essere presenti nel dominio che sta richiamando l'arbitro.

- **L'utilizzo dei programmi di caricamento non è consigliato**

  I programmi di caricamento possono essere utilizzati per interfacciare le modifiche tra una griglia e un database. >È improbabile che tutte le griglie (domini) in una topologia siano collocate geograficamente con lo stesso database. La latenza WAN e gli altri fattori potrebbero rendere questo utilizzo non desiderabile.

  Il precaricamento della griglia è un altro problema che richiede una progettazione attenta. Generalmente, quando una griglia è riavviata, essa viene nuovamente precaricata. Il precaricamento non è necessario o addiruttura desiderabile allorché si utilizza una replica multi-master. Appena un dominio è in linea, esso automaticamente ricarica se stesso con i contenuti del dominio a cui è collegato. Di conseguenza, non è necessario inizializzare un precaricamento manuale per una griglia che è un dominio in una topologia di replica multi-master.

  I programmi di caricamento generalmente rispettano le regole di inserimento ed aggiornamento. Utilizzando la replica multi-master, gli inserimenti devono essere trattati come unioni. Quando i dati sono tratti in remoto dopo che un dominio è stato riavviato, i dati esistenti saranno ‘inseriti' nel dominio locale. Poiché questi dati potrebbero già essere presenti nel database locale, un tipico inserimento potrebbe andare in errore con un'eccezione di chiave duplicata nel database. Invece devono essere utilizzate le semantiche di unione.

eXtreme Scale può essere configurato per eseguire un precaricamento basato sul frammento utilizzando i metodi di precaricamento nei plug-in Loader. Non utilizzare questa tecnica nella topologia di replica multi-master. Invece, utilizzare un precaricamento basato sul client quando la topologia è avviata (inizialmente). Consentire alla topologia multi-master di aggiornare qualsiasi dominio riavviato con una copia corrente di quanto è stato memorizzato in altri domini nella topologia. Dopo che i domini sono stati avviati, la topologia multi-master è responsabile di mantenerli in sincronia.

- **Non è supportato EntityManager**

  Una serie di mappe contenenti un'entità mappa non è replicata tra i domini.
- **Le mappe array di byte non sono supportate**

  Una serie di mappe contenenti una mappa che è configurata con COPY_TO_BYTES non è replicata tra i domini.
- **Write-behind non è supportato**

  Una serie di mappe contenenti una mappa che è configurata con il supporto write-behind non è replicata tra i domini.

JMS per la distribuzione delle modifiche della transazione

Utilizzare JMS (Java Message Service) per modifiche di transazioni distribuite tra livelli differenti o in ambienti su piattaforme miste.

JMS è un protocollo ideale per modifiche distribuite tra livelli differenti o in ambienti su piattaforme miste. Ad esempio, alcune applicazioni che utilizzano eXtreme Scale potrebbero essere distribuite su IBM WebSphere Application Server Community Edition, Apache Geronimo o Apache Tomcat, laddove altre applicazioni potrebbero essere eseguite su WebSphere Application Server Versione 6.x. JMS è ideale per le modifiche distribuite tra peer eXtreme Scale in questi diversi ambienti. Il sistema di trasporto messaggi del gestore HA (High Availability) è molto veloce ma può distribuire modifiche solo a Java virtual machine che sono in un gruppo principale singolo. JMS è più lento ma consente impostazioni di client dell'applicazione più ampie e varie per condividere un ObjectGrid. JMS è ideale quando si condividono dati in un ObjectGrid tra un client fat Swing ed un'applicazione distribuita su WebSphere Extended Deployment.

Il Client Invalidation Mechanism incorporato e il Peer-to-Peer Replication sono esempi di distribuzione delle modifiche transazionali basate su JMS. Consultare le informazioni sulla configurazione della replica peer-to-peer con JMS in*Guida alla gestione* per ulteriori informazioni.

Implementazione di JMS

JMS viene implementato per modifiche di transazioni distribuite mediante un oggetto Java che si comporta come un ObjectGridEventListener. Questo oggetto può propagare lo stato nei seguenti quattro modi:

1. Invalidazione: qualunque voce eliminata, aggiornata o cancellata viene rimossa su tutti i peer Java virtual machine quando viene ricevuto il messaggio.
2. Invalidazione condizionale: la voce viene eliminata solo se la versione locale è la stessa o è più vecchia rispetto alla versione sul publisher.
3. Invio: qualunque voce eliminata, aggiornata, cancellata o inserita viene aggiunta o sovrascritta su tutti i peer Java virtual machine quando viene ricevuto il messaggio JMS.

4. Invio condizionale: la voce viene solo aggiornata o aggiunta sul lato ricevente se la voce locale è meno recente della versione che è stata pubblicata.

Ascolto di modifiche per la pubblicazione

Il plug-in implementa l'interfaccia ObjectGridEventListener per intercettare l'evento transactionEnd. Quando eXtreme Scale richiama questo metodo, il plug-in prova a convertire l'elenco LogSequence per ogni mappa toccata dalla transazione in un messaggio JMS e poi lo pubblica. Il plug-in può essere configurato per pubblicare modifiche per tutte le mappe o per una serie secondaria di mappe. Gli oggetti LogSequence vengono elaborati per le mappe abilitate alla pubblicazione. La classe ObjectGrid di LogSequenceTransformer serializza un LogSequence filtrato per ogni mappa in un flusso. Dopo la serializzazione di tutti i LogSequences in un flusso, viene creato un ObjectMessage JMS e viene pubblicato in un argomento ben noto.

Ascolto di messaggi JMS e loro applicazione nell'ObjectGrid locale

Lo stesso plug-in avvia anche un thread che si avvita in un loop, ricevendo tutti i messaggi che vengono pubblicati in un argomento ben noto. Quando arriva un messaggio, passa il contenuto del messaggio alla classe LogSequenceTransformer dove viene convertito in una serie di oggetti LogSequence. Quindi, viene avviata una transazione no-write-through. Ogni oggetto LogSequence viene fornito al metodo Session.processLogSequence, che aggiorna le mappe in locale con le modifiche. Il metodo processLogSequence capisce la modalità di distribuzione. Viene eseguito il commit della transazione e la cache locale ora riflette le modifiche. Per ulteriori informazioni sull'utilizzo di JMS per la distribuzione delle modifiche della transazione, consultare le informazioni sulla distribuzione delle modifiche tra Java Virtual Machines peer in *Guida alla gestione*.

Serie di mappe per la replica

La replica è abilitata tramite l'associazione delle BackingMap con una MapSet.

Una MapSet è una raccolta di mappe catalogate in base alla chiave di partizione. Questa chiave di partizione viene derivata dalla chiave della mappa singola prelevando dal suo modulo hash il numero di partizioni. Quindi, se un gruppo di mappe all'interno della MapSet ha una chiave di partizione X, tali mappe saranno memorizzate nella partizione X corrispondente nella griglia. Se un altro gruppo ha una chiave di partizione Y, tutte le mappe verranno memorizzate nella partizione Y e così via. Inoltre, tutti i dati nelle mappe vengono replicati in base alle politiche definite sulla MapSet, il che è valido solo per le topologie di eXtreme Scale distribuite (non necessario per le istanze locali).

Consultare “Partizionamento” a pagina 91 per ulteriori dettagli.

Alle MapSet vengono assegnati il numero di partizioni ed una politica di replica. La configurazione di replica della MapSet identifica semplicemente il numero di frammenti di replica sincroni ed asincroni che una MapSet deve avere per il frammento primario. Ad esempio, se vi dovrà essere una replica sincrona ed una asincrona, ciascuna BackingMap assegnata alla MapSet avrà un frammento di replica automaticamente distribuito all'interno della serie di contenitori disponibili per eXtreme Scale. La configurazione di replica può anche consentire ai client di leggere i dati da server replicati in modo sincrono. Ciò potrà distribuire il carico

delle richieste di lettura su più server in eXtreme Scale. La replica ha unicamente un impatto sul modello di programmazione quando si esegue il precaricamento delle BackingMap.

Per i dettagli relativi alle diverse opzioni di configurazione, consultare quanto segue:

Capitolo 6. Panoramica sull'elaborazione della transazione

Sessioni ed elaborazione della transazione

WebSphere eXtreme Scale utilizza le transazioni come meccanismo di interazione con i dati.

Per interagire con i dati, il thread dell'applicazione necessita di una sessione propria. Quando l'applicazione desidera utilizzare l'ObjectGrid su un thread, chiama uno dei metodi ObjectGrid.getSession per ottenere un thread. Con la sessione, l'applicazione può utilizzare i dati memorizzati nelle mappe ObjectGrid.

Quando un'applicazione utilizza un oggetto Session, la sessione deve essere nel contesto di una transazione. Una transazione inizia ed esegue il commit o inizia ed esegue il rollback utilizzando i metodi begin, commit, e rollback sull'oggetto Session. Le applicazioni possono anche operare in modalità di commit automatico, in cui Session si avvia ed esegue il commit di una transazione automaticamente ogni volta che viene eseguita un'operazione sulla mappa. La modalità di commit automatico non può raggruppare più operazioni in una singola transazione, pertanto è l'opzione più lenta se si sta creando un batch di più operazioni in una singola transazione. Tuttavia, per le transazioni che contengono una sola operazione, il commit automatico è l'opzione più veloce.

Transazioni

Le transazioni hanno molti vantaggi per la memorizzazione e la manipolazione dei dati. È possibile utilizzare le transazioni per proteggere la griglia da modifiche simultanee, per applicare più modifiche come un'unità contemporanea, per replicare i dati e per implementare il ciclo di vita per i blocchi sulle modifiche.

Quando viene avviata una transazione, WebSphere eXtreme Scale assegna una speciale mappa delle differenze per contenere le modifiche o le copie correnti delle coppie chiave-valore utilizzate dalla transazione. In genere, quando si accede ad una coppia chiave-valore, il valore viene copiato prima che l'applicazione riceva il valore. La mappa delle differenze tiene traccia di tutte le modifiche per operazioni quali l'inserimento, l'aggiornamento, il richiamo, la rimozione e così via. Le chiavi non vengono copiate perché ritenute immutabili. Se viene specificato un oggetto ObjectTransformer, quell'oggetto viene utilizzato per copiare il valore. Se la transazione utilizza un blocco ottimistico, viene tenuta traccia anche delle immagini precedenti dei valori per il confronto quando viene eseguito il commit della transazione.

Se viene eseguito il rollback di una transazione, le informazioni della mappa delle differenze vengono eliminate, e i blocchi sulle voci vengono rilasciati. Quando viene eseguito il commit di una transazione, le modifiche vengono applicate alle mappe ed i blocchi vengono rilasciati. Se viene utilizzato il blocco ottimistico, eXtreme Scale confronta le versioni delle immagini precedenti dei valori con i valori contenuti nella mappa. Affinché possa essere seguito il commit della transazione, questi valori devono corrispondere. Questo confronto consente uno schema di blocco di più versioni, ma questo comporta la creazione di due copie quando la transazione accede alla voce. Tutti i valori vengono copiati di nuovo e la nuova copia viene memorizzata nella mappa. WebSphere eXtreme Scale esegue

questa copia per proteggersi dalla modifica del riferimento applicazione da parte dell'applicazione sul valore successivo ad un commit.

È possibile evitare di utilizzare diverse copie delle informazioni. L'applicazione può salvare una copia utilizzando un blocco pessimistico invece del blocco ottimistico come costo della limitazione della simultaneità. Anche la copia del valore durante il commit può essere evitata se l'applicazione decide di non modificare un valore dopo un commit.

Vantaggi delle transazioni

Utilizzare le transazioni per i seguenti motivi:

Utilizzando le transazioni, è possibile:

- Eseguire il rollback delle modifiche se si verifica un'eccezione o la logica di business deve annullare le modifiche di stato.
- Per applicare più modifiche come un'unità atomica al momento del commit.
- Mantenere e rilasciare blocchi sui dati per applicare più modifiche come un'unità atomica al momento del commit.
- Proteggere un thread da modifiche simultanee.
- Implementare un ciclo di vita per i blocchi sulle modifiche.
- Produrre un'unità atomica di replica.

Dimensione della transazione

Le transazioni di dimensioni maggiori sono più efficienti, specialmente per la replica. Tuttavia, le transazioni di dimensioni maggiori possono influire negativamente sulla simultaneità poiché i blocchi sulle voci vengono mantenuti per un periodo di tempo più lungo. Se si utilizzano transazioni di dimensioni maggiori, è possibile accrescere le prestazioni della replica. Questo accrescimento delle prestazioni è importante quando si precarica una mappa. Provare differenti dimensioni di batch per determinare cosa funziona meglio per lo scenario.

Transazioni più grandi sono utili anche con i programmi di caricamento. Se viene utilizzato un programma di caricamento che può eseguire il batch SQL, sono possibili significativi miglioramenti delle prestazioni a seconda della transazione e delle riduzioni significative del carico sul lato database. Questo miglioramento delle prestazioni dipende dall'implementazione di Loader.

Modalità di commit automatico

Se non è stata avviata attivamente alcuna transazione, quando un'applicazione interagisce con un oggetto ObjectMap viene eseguita un'operazione automatica di inizio e di commit per conto dell'applicazione. Questa operazione automatica di inizio e di commit funziona, ma impedisce il funzionamento corretto del rollback e del blocco. Questo ha un impatto sulla velocità di replica sincrona a causa della dimensione molto piccola della transazione. Se si sta usando un'applicazione gestore di entità, non utilizzare la modalità di commit automatico poiché gli oggetti che vengono cercati con il metodo EntityManager.find divengono immediatamente non gestiti alla restituzione del metodo e non utilizzabili.

Coordinatori di transazioni esterne

In genere le transazioni iniziano con il metodo session.begin e terminano con il metodo session.commit. Tuttavia, quando eXtreme Scale è incorporato, le transazioni potrebbero essere avviate e terminate da un coordinatore di transazioni esterne. Se si utilizza un coordinatore di transazioni esterne, non è necessario richiamare il metodo session.begin e terminare con il metodo session.commit. Per ulteriori informazioni su eXtreme Scale e l'interazione di transazione esterne, consultare *Guida alla programmazione*. Se si utilizza WebSphere Application Server, è possibile utilizzare il plug-in WebSphereTranscationCallback. Per ulteriori informazioni sui plug-in disponibili con WebSphere eXtreme Scale, consultare *Guida alla programmazione*.

Attributo CopyMode

È possibile ottimizzare il numero di copie definendo l'attributo CopyMode degli oggetti BackingMap o ObjectMap.

È possibile ottimizzare il numero di copie definendo l'attributo CopyMode degli oggetti BackingMap o ObjectMap. La modalità di copia ha i seguenti valori:

- COPY_ON_READ_AND_COMMIT
- COPY_ON_READ
- NO_COPY
- COPY_ON_WRITE
- COPY_TO_BYTES

Il valore COPY_ON_READ_AND_COMMIT è quello predefinito. Il valore COPY_ON_READ esegue la copia quando vengono inizialmente richiamati i dati, ma non durante il commit. Questa modalità è sicura se l'applicazione non modifica un valore dopo aver eseguito il commit di una transazione. Il valore NO_COPY non copia i dati, ed è una modalità sicura solo per i dati di sola lettura. Se i dati non cambiano mai, non è necessario copiarli per motivi di isolamento.

Prestare attenzione quando si utilizza il valore di attributo NO_COPY con mappe che possono essere aggiornate. WebSphere eXtreme Scale utilizza la copia al primo tocco per consentire il rollback della transazione. L'applicazione ha cambiato solo la copia, e come risultato, eXtreme Scale elimina la copia. Se viene utilizzato il valore di attributo NO_COPY, e l'applicazione modifica il valore sottoposto a commit, il completamento di un rollback non è possibile. La modifica del valore sottoposto a commit porta a problemi con gli indici, la replica e così via, poiché gli indici e le repliche si aggiornano quando viene eseguito il commit della transazione. Se si modificano i dati sottoposti a commit e poi si esegue il rollback della transazione, che di fatto non viene eseguito, gli indici non vengono aggiornati e la replica non viene eseguita. Altri thread possono immediatamente individuare le modifiche senza commit, anche se hanno i blocchi. Utilizzare il valore di attributo NO_COPY solo per le mappe di sola lettura o per le applicazioni che completano la copia appropriata prima di modificare il valore. Se si utilizza il valore attributo NO_COPY e si chiama il supporto IBM con un problema di integrità dei dati, viene richiesto di riprodurre il problema con la modalità di copia impostata su COPY_ON_READ_AND_COMMIT.

Il valore COPY_TO_BYTES memorizza i valori nella mappa in una formato serializzato. Alla lettura, eXtreme Scale deserializza il valore da un formato serializzato e al commit memorizza il valore in un formato serializzato. Con questo metodo viene eseguita una copia si alla lettura che al commit.

La modalità di copia predefinita per una mappa può essere configurata sull'oggetto BackingMap. È possibile anche cambiare la modalità di copia sulle mappe prima di avviare una transazione utilizzando il metodo ObjectMap.setCopyMode.

Segue un esempio di un frammento di mappa di backup da un file `objectgrid.xml` che mostra come impostare la modalità di copia per una determinata mappa di backup. Questo esempio presume che si stia utilizzando `cc` come spazio dei nomi `objectgrid/config`.

```
<cc:backingMap name="RuntimeLifespan" copyMode="NO_COPY"/>
```

Per ulteriori informazioni, consultare le informazioni relative alle migliori pratiche per copyMode in *Guida alla programmazione*.

Blocco della voce della mappa

Un ObjectGrid BackingMap supporta diverse strategie di blocco per le mappe al fine di mantenere la congruenza nella voce della cache.

Ciascuna BackingMap può essere configurata per utilizzare una delle seguenti strategie di blocco.

1. Modalità di blocco ottimistica
2. Modalità di blocco pessimistica
3. None

La strategia di blocco predefinita è OTTIMISTICA. Utilizzare il blocco ottimistico quando i dati vengono modificati non frequentemente. I blocchi sono conservati solo per un breve periodo mentre i dati sono letti dalla cache e copiati alla transazione. Quando la cache della transazione è sincronizzata con la cache principale, qualsiasi oggetto di cache che è stato aggiornato, viene controllato rispetto alla versione originale. Se la verifica non riesce, allora la transazione esegue il rollback e si genera un'eccezione OptimisticCollisionException.

La strategia di blocco PESSIMISTICA acquisisce i blocchi per le voci della cache e dovrebbe essere utilizzata quando i dati vengono modificati frequentemente. Qualche volta una voce dela cache è in lettura viene acquisito un blocco e conservato condizionatamente fino a quando non è completata la transazione. La durata di alcuni blocchi può essere regolata utilizzando i livelli di isolamento della transazione per la sessione.

Se non è richiesto il blocco perché i dati non vengono mai aggiornati o vengono aggiornati solo durante i periodi di attesa, è possibile disabilitare il blocco utilizzando la strategia di blocco NONE. Questa strategia è molto veloce perché non viene richiesto un gestore blocco. La strategia di blocco NONE è ideale per le tabelle di ricerca o per le mappe in sola lettura.

Per ulteriori informazioni relative alle strategie di blocco, consultare nella sezione *Panoramica sul prodotto*.

Specifica di una strategia di blocco

L'esempio di seguito riportato dimostra come la strategia di blocco possa essere impostata sulle BackingMaps map1, map2 e map3 , laddove ciascuna mappa utilizza una diversa strategia di blocco. Il primo frammento di codice mostra in che modo utilizzare XML per la configurazione della strategia di blocco e il secondo frammento di codice mostra un approccio programmatico.

Approccio XML

BackingMap configuration - XML example
```
<?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="test">
            <backingMap name="map1"
                lockStrategy="PESSIMISTIC" numberOfLockBuckets="31"/>
            <backingMap name="map2"
                lockStrategy="OPTIMISTIC" numberOfLockBuckets="409"/>
            <backingMap name="map3"
                lockStrategy="NONE"/>
        </objectGrid>
    </objectGrids>
</objectGridConfig>
```

Approccio programmatico

Configurazione BackingMap - esempio programmatico
```
import com.ibm.websphere.objectgrid.BackingMap;
import com.ibm.websphere.objectgrid.LockStrategy;
import com.ibm.websphere.objectgrid.ObjectGrid;
import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
...
ObjectGrid og =
 ObjectGridManagerFactory.getObjectGridManager().createObjectGrid("test");
BackingMap bm = og.defineMap("map1");
bm.setLockStrategy( LockStrategy.PESSIMISTIC );
bm.setNumberOfLockBuckets(31);
bm = og.defineMap("map2");
bm.setNumberOfLockBuckets(409);
bm.setLockStrategy( LockStrategy.OPTIMISTIC );
bm = og.defineMap("map3");
bm.setLockStrategy( LockStrategy.NONE );
```

Per evitare un'eccezione java.lang.IllegalStateException, il metodo setLockStrategy deve essere richiamato prima di utilizzare i metodi Initialize o getSession in un'istanza locale ObjectGrid instance.

Per ulteriori informazioni, consultare l'argomento relative alle strategie di blocco nella sezione *Panoramica sul prodotto*.

Configurazione di un gestore blocco

Quando vengono utilizzate sia una strategia di blocco PESSIMISTICA che una OTTIMISTICA, un gestore blocco viene creato per la BackingMap. Il gestore blocco utilizza una mappa di hash per tenere traccia delle voci che sono bloccate da una o più transazioni. Se esistono molte voci della mappa nell'associazione hash, più bucket di blocco possono condurre ad una migliore prestazione. better performance. Il rischio di conflitti di sincronizzazione Java è più basso quanto più cresce il numero di bucket. Più bucket di blocco conduce anche ad una maggiore concorrenza. Gli esempi precedenti mostrano in che modo un'applicazione può impostare il numero di bucket di blocco da utilizzare per una determinata istanza BackingMap.

Per evitare un'eccezione java.lang.IllegalStateException, il metodo setNumberOfLockBuckets deve essere richiamato prima di richiamare i metodi Initialize o getSession su un'istanza ObjectGrid. Il parametro del metodo setNumberOfLockBuckets è un primitivo Java intero che specifica il numero di bucket di blocco da utilizzare. L'utilizzo di un numero primo può consentire una

distribuzione uniforme delle voci della mappa sui bucket di bloccco. Un buon punto di partenza per migliorare le prestazioni è impostare il numero di bucket di blocco al 10 per cento circa del numero previsto di voci di BackingMap.

LockDeadlockException

Di seguito è riportato un esempio di codice che mostra il rilevamento delle eccezioni e il messaggio che ne risulta, viene successivamente visualizzato.

```
try {
...
} catch (ObjectGridException oe) {
System.out.println(oe);
}
```

Il risultato è:

```
com.ibm.websphere.objectgrid.plugins.LockDeadlockException: _Message
```

Questo messaggio rappresenta la stringa che viene passata come parametro quando l'eccezione è creata e attivata.

Causa dell'eccezione

Il tipo più comune di eccezione di deadlock avviene quando si utilizza la strategia di blocco pessimisticoe due client separati ciascuno avente un blocco condiviso su un particolare oggetto. Successivamente, entrambi i client tentano di promuovere un blocco esclusivo su quell'oggetto. Il diagramma di seguito riportato illustra tale situazione, compreso i blocchi della transazione che causano 'eccezione.

Il frammento di codice Java di seguito riportato dimostra in che modo passare un file di configurazione XML per creare un ObjectGrid.

Questo è una visualizzazione dell'estratto di ciò che accade nel proprio programma quando si verifica l'eccezione. In un'appplicazione con molti thread che aggiornano lo stesso ObjecMap, è possibile incorrere in questa situazione. Di seguito è riportato un esempio di due client che esguono i blocchi del codice transazione come illustrato nella figura precedente.

Soluzioni possibili

È possibile ritrovare la situazione illustrata nella figura 1 quando numerosi thread avviano le transazioni in una particolare mappa. In questo caso, l'eccezione è attivata per evitare che il programma sia sospeso. È possibile avvisare se stessi e aggiungere codice al blocco rilevato per ulteriori dettagli sulla causa. Poiché questa eccezione si vede solo in una strategia di blocco pessimistico, una semplice soluzione è semplificare l'utilizzo della strategia di blocco. Se si acquisisce una strategia di blocco pessimistico, tuttavia è possibile utilizzare il metodo getForUpdate invece del metodo Get. Ciò elimina la ricezione delle eccezioni per la situazione descritta precedentemente.

Strategie di blocco

Le strategie di blocco includono le strategie pessimistica, ottimistica e none Per scegliere una strategia di blocco, è necessario considerare questioni come la percentuale di ciascun tipo di operazione che si ha se oppure se si utilizza o meno un programma di caricamento e così via.

I blocchi sono collegati dalle transazioni. È possibile specificare le seguenti impostazioni di blocco:

- **Nessun blocco**: l'esecuzione senza l'impostazione di blocco è l'opzione più veloce. Se si stanno utilizzando dati di sola lettura, il blocco potrebbe non essere necessario.
- **Blocco pessimistico**: acquisisce i blocchi sulle voci e mantiene i blocchi fino al commit. Questa strategia di blocco fornisce una buona congruenza a spese della velocità di trasmissione.
- **Blocco ottimistico**: prende un'immagine precedente di ogni record che la transazione tocca e la confronta con i valori correnti della voce quando viene seguito il commit della transazione. Se i valori della voce cambiano, la transazione esegue il rollback. Non vengono mantenuti blocchi fino al commit. Questa strategia di blocco fornisce una migliore simultaneità della strategia pessimistica, con il rischio del rollback della transazione e il dispendio di memoria per la creazione di un'ulteriore copia della voce.

Impostare la strategia di blocco sulla BackingMap. Non è possibile modificare la strategia di blocco di ogni transazione. Di seguito è riportato un esempio di frammento XML che mostra come impostare la modalità di blocco su una mappa utilizzando il file XML, presupponendo che `cc` sia lo spazio dei nomi per lo spazio dei nomi `objectgrid/config`:

```
<cc:backingMap name="RuntimeLifespan" lockStrategy="PESSIMISTIC" />
```

Blocco pessimistico

Utilizzare la strategia di blocco pessimistico per le mappe in lettura e scrittura quando altre strategie di blocco non sono possibili. Quando una mappa ObjectGrid è configurata per utilizzare una strategia di blocco pessimistico, si ottiene un blocco della transazione pessimistica per una voce della mappa quando una transazione prende prima la voce da BackingMap. Il blocco pessimistico è conservato fino a quando l'applicazione non completa la transazione. Generalmente la strategia di blocco pessimistico viene utilizzata nelle seguenti situazioni:

- Quando viene configurata la BackingMap con o senza un programma di caricamento e le informazioni sul controllo versioni non sono disponibili.
- Quando viene utilizzata la BackingMap direttamente da un'applicazione che ha necessità di una guida da eXtreme Scale per il controllo della concorrenza.
- Quando le informazioni sul controllo versioni sono disponibili, ma le transazioni di aggiornamento sono frequentemente in conflitto con le voci di backup risultando in un aggiornamento pessimistico non riuscito.

Poiché la strategia di blocco pessimistico ha il maggiore impatto sulle prestazioni e la scalabilità, questa strategia dovrebbe essere utilizzata per le mappe in lettura e scrittura solo quando altre strategie di blocco non sono praticabili. Ad esempio, queste situazioni potrebbero includere quando si verifica di frequente che un aggiornamento ottimistico non riesce o quando il recupero da un errore ottimistico è difficile da gestire per un'applicazione.

Blocco ottimistico

La strategia di blocco ottimistico presuppone che nessuna delle due transazioni possa tentare di aggiornare la stessa voce della mappa durante l'esecuzione contemporanea. A causa di questa convinzione, la modalità blocco non ha necessità di essere conservata per il ciclo di vita della transazione poiché è improbabile che più di una transazione possa aggiornare la voce della mappa contemporaneamente. La strategia di blocco ottimistico è generalmente utilizzata nelle seguenti situazioni:

- quando viene configurata la BackingMap con o senza un programma di caricamento e le informazioni sul controllo versioni sono disponibili.
- Quando una BackingMap ha per lo più transazioni che eseguono operazioni di lettura. Le operazioni di inserimento, aggiornamento e rimozione delle voci della mappa non si verificano spesso nella BackingMap.
- Quando una BackingMap viene inserita, aggiornata o rimossa più frequentemente di quanto non sia letta, ma le transazioni raramente sono in conflitto sulla stessa voce della mappa.

Come la strategia di blocco pessimistico, i metodi nell'interfaccia ObjectMap determinano in che modo eXtreme Scale tenta automaticamente di acquisire una modalità blocco per la voce della mappa che sta per essere acceduta. Tuttavia, esistono le seguenti differenze tra le strategie pessimistica e ottimistica:

- Come la strategia di blocco pessimistico, una modalità blocco S viene acquisita dai metodi Get e getAll quando viene richiamato il metodo. Tuttavia, con un blocco pessimistico, la modalità blocco S non è conservata fino a quando la transazione non è completa. Invece la modalità blocco S viene rilasciata prima che il metodo ritorni all'applicazione. Lo scopo per acquisire la modalità blocco è tale per cui eXtreme Scale può garantire che solo i dati su cui è stato eseguito il commit dalle altre transazioni è visibile alla transazione corrente. Dopo che eXtreme Scale ha verificato che è stato eseguito il commit dei dati, la modalità blocco S viene rilasciata. Al momento del commit viene eseguito un controllo sulla versione ottimistico per garantire che nessuna altra transazione abbia modificato la voce della mappa dopo che la transazione corrente ha rilasciato la propria modalità blocco S. Se non viene eseguito il fetch di una voce dalla mappa prima che essa sia aggiornata, invalidata o eliminata, eXtreme Scale il runtime implicitamente esegue il fetch della voce della mappa. Questa operazione implicita di caricamento viene eseguita per ottenere il valore corrente al momento in cui è stato richiesto di modificare la voce.
- Diversamente dalla strategia di blocco pessimistico, i metodi getForUpdate e getAllForUpdate vengono gestiti esattamente come i metodi Get e getAll quando viene utilizzata la strategia di blocco ottimistico. Cioè una modalità blocco S viene acquisita all'avvio del metodo e la modalità blocco S viene rilasciata prima di ritornare all'applicazione.

Tutti gli altri metodi ObjectMap vengono gestiti esattamente come viene gestita la strategia di blocco pessimistico. Cioè quando viene richiamato il metodo Commit, si ottiene una modalità blocco X per la voce della mappa che è stata inserita, aggiornata, rimossa,toccata o invalidata e la modalità blocco X è conservata fino a quando la transazione non completa l'elaborazione del commit.

La strategia di blocco ottimistico presuppone che nessuna transazione in esecuzione contemporaneamente tenti di aggiornare la stessa voce della mappa. A causa di questo assunto, la modalità blocco non necessita di essere conservata per il ciclo di vita della transazione poiché è improbabile che più di una transazione possa aggiornare contemporaneamente la voce della mappa. Tuttavia, poiché non è conservata una modalità di blocco, un'altra transazione concomitante potrebbe potenzialmente aggiornare la voce della mappa dopo che la transazione corrente ha rilasciato la modalità blocco S.

Per gestire questa possibilità, eXtreme Scale ottiene un blocco X al momento del commit ed esegue un controllo sulla versione ottimistico per verificare che nessun'altra transazione abbia modificato la voce della mappa dopo che la transazione corrente abbia letto la voce della mappa da BackingMap. Se un'altra transazione modifica la voce della mappa, il controllo sulla versione non riesce e si

verifica un'eccezione OptimisticCollisionException. Questa eccezione forza la transazione corrente ad eseguire il roll back e l'applicazione deve tentare l'intera transazione nuovamente. La strategia di blocco ottimistico è molto utile quando una mappa è per lo più letta ed è improbabile che si verifichino aggiornamenti alla stessa voce della mappa.

Nessun blocco

Quando una BackingMap viene configurata per non utilizzare alcuna strategia di blocco, non si ottiene alcun blocco della transazione per una voce della mappa.

L'utilizzo di nessuna strategia di blocco è utile quando un'applicazione JavaBeans™ utilizza Hibernate per ottenere i dati permanenti. In questo scenario, la BackingMap viene configurata senza un programma di caricamento e il gestore della persistenza utilizza la BackingMap come un cache di dati. In questo scenario, il gestore della persistenza fornisce un controllo sulla concorrenza tra le transazioni che accedono le stesse voci della mappa.

WebSphere eXtreme Scale non ha necessità di ottenere alcun blocco sulla transazione a scopo di controllo della concorrenza. Questa situazione presuppone che il gestore della persistenza non rilasci i propri blocchi sulla transazione prima di aggiornare la mappa ObjectGrid con le modifiche sulle quali è stato eseguito il commit. Se il gestore della persistenza rilascia i suoi blocchi, allora deve essere utilizzata una strategia di blocco ottimistico o pessimistico. Ad esempio, supponiamo che il gestore della persistenza di un contenitore EJB stia aggiornando una mappa ObjectGrid con i dati sui quali è stato eseguito il commit nella transazione gestita dal contenitore EJB. Se l'aggiornamento della mappa ObjectGrid si verifica prima che siano rilasciati i blocchi della transazione del gestore della persistenza, allora è possibile non utilizzare alcuna strategia di blocco. Se si verifica l'aggiornamento della mappa ObjectGrid dopo che sono stati rilasciati i blocchi della transazione del gestore della persistenza, allora devi utilizzare sia la strategia di blocco ottimistico che quella di blocco pessimistico.

Un altro scenario in cui non può essere utilizzata alcuna strategia di blocco, si ha quando l'applicazione utilizza direttamente una BackingMap e un programma di caricamento è configurato per la mappa. In questo scenario, il programma di caricamento utilizza il supporto del controllo della concorrenza che viene fornito da RDBMS (Sistema di gestione del database relazionale) utilizzando sia JDBC (Java database connectivity) che Hibernate per accedere i dati in un database relazionale. L'implementazione del programma di caricamento può utilizzare sia un approccio ottimistico che uno pessimistico. Un programma di caricamento che utilizza un blocco ottimistico o un approccio con controllo sulla versione guida nel raggiungere i risultati migliori nella concorrenza e nelle prestazioni. Per ulteriori informazioni relative all'approccio di blocco ottimistico, consultare la sezione OptimisticCallback nelle per le informazioni relative alle considerazioni sul programma di caricamento in *Guida alla gestione*. Se si sta utilizzando un programma di caricamento che utilizza il supporto di blocco pessimistico di un backend sottostante si potrebbe desiderare di utilizzare il parametro forUpdate che viene passato nel metodo Get dell'interfaccia del programma di caricamento. Impostare questo parametro su true se è stato utilizzato il metodo getForUpdate dell'interfaccia ObjectMap da un'applicazione per ottenere i dati. Il programma di caricamento può utilizzare questo parametro per determinare se richiedere un blocco aggiornabile per la riga che si sta leggendo. Ad esempio, il DB2 ottiene un blocco aggiornabile quando l'istruzione select contiene una clausola `FOR UPDATE`. Questo approccio offre la stessa possibilità di prevenire un deadlock che è descritta in "Blocco pessimistico" a pagina 163.

Per ulteriori informazioni, consultare l'argomento relativo alla gestione dei blocchi in *Guida alla programmazione* o blocco della voce della mappa in *Guida alla gestione*.

JMS per la distribuzione delle modifiche della transazione

Utilizzare JMS (Java Message Service) per modifiche di transazioni distribuite tra livelli differenti o in ambienti su piattaforme miste.

JMS è un protocollo ideale per modifiche distribuite tra livelli differenti o in ambienti su piattaforme miste. Ad esempio, alcune applicazioni che utilizzano eXtreme Scale potrebbero essere distribuite su IBM WebSphere Application Server Community Edition, Apache Geronimo o Apache Tomcat, laddove altre applicazioni potrebbero essere eseguite su WebSphere Application Server Versione 6.x. JMS è ideale per le modifiche distribuite tra peer eXtreme Scale in questi diversi ambienti. Il sistema di trasporto messaggi del gestore HA (High Availability) è molto veloce ma può distribuire modifiche solo a Java virtual machine che sono in un gruppo principale singolo. JMS è più lento ma consente impostazioni di client dell'applicazione più ampie e varie per condividere un ObjectGrid. JMS è ideale quando si condividono dati in un ObjectGrid tra un client fat Swing ed un'applicazione distribuita su WebSphere Extended Deployment.

Il Client Invalidation Mechanism incorporato e il Peer-to-Peer Replication sono esempi di distribuzione delle modifiche transazionali basate su JMS. Consultare le informazioni sulla configurazione della replica peer-to-peer con JMS in*Guida alla gestione* per ulteriori informazioni.

Implementazione di JMS

JMS viene implementato per modifiche di transazioni distribuite mediante un oggetto Java che si comporta come un ObjectGridEventListener. Questo oggetto può propagare lo stato nei seguenti quattro modi:

1. Invalidazione: qualunque voce eliminata, aggiornata o cancellata viene rimossa su tutti i peer Java virtual machine quando viene ricevuto il messaggio.
2. Invalidazione condizionale: la voce viene eliminata solo se la versione locale è la stessa o è più vecchia rispetto alla versione sul publisher.
3. Invio: qualunque voce eliminata, aggiornata, cancellata o inserita viene aggiunta o sovrascritta su tutti i peer Java virtual machine quando viene ricevuto il messaggio JMS.
4. Invio condizionale: la voce viene solo aggiornata o aggiunta sul lato ricevente se la voce locale è meno recente della versione che è stata pubblicata.

Ascolto di modifiche per la pubblicazione

Il plug-in implementa l'interfaccia ObjectGridEventListener per intercettare l'evento transactionEnd. Quando eXtreme Scale richiama questo metodo, il plug-in prova a convertire l'elenco LogSequence per ogni mappa toccata dalla transazione in un messaggio JMS e poi lo pubblica. Il plug-in può essere configurato per pubblicare modifiche per tutte le mappe o per una serie secondaria di mappe. Gli oggetti LogSequence vengono elaborati per le mappe abilitate alla pubblicazione. La classe ObjectGrid di LogSequenceTransformer serializza un LogSequence filtrato per ogni mappa in un flusso. Dopo la serializzazione di tutti i LogSequences in un flusso, viene creato un ObjectMessage JMS e viene pubblicato in un argomento ben noto.

Ascolto di messaggi JMS e loro applicazione nell'ObjectGrid locale

Lo stesso plug-in avvia anche un thread che si avvita in un loop, ricevendo tutti i messaggi che vengono pubblicati in un argomento ben noto. Quando arriva un messaggio, passa il contenuto del messaggio alla classe LogSequenceTransformer dove viene convertito in una serie di oggetti LogSequence. Quindi, viene avviata una transazione no-write-through. Ogni oggetto LogSequence viene fornito al metodo Session.processLogSequence, che aggiorna le mappe in locale con le modifiche. Il metodo processLogSequence capisce la modalità di distribuzione. Viene eseguito il commit della transazione e la cache locale ora riflette le modifiche. Per ulteriori informazioni sull'utilizzo di JMS per la distribuzione delle modifiche della transazione, consultare le informazioni sulla distribuzione delle modifiche tra Java Virtual Machines peer in *Guida alla gestione*.

Transazioni su partizione singola e sulle partizioni della griglia

La distinzione principale tra WebSphere eXtreme Scale e le soluzioni di memorizzazione dei dati tradizionali come i database relazionali o i database in memoria è rappresentata dall'utilizzo delle partizioni, che consentono alla cache di scalare in modo lineare. I tipi importanti di transazione da considerare sono le transazioni su partizione singola e su tutte le partizioni (sulla griglia).

In generale, le interazioni con la cache possono essere suddivise in transazioni su partizione singola o transazioni sulla griglia, come riportato di seguito.

Transazioni su partizione singola

Le transazioni su partizione singola rappresentano il metodo preferito per l'interazione con le cache ospitate da WebSphere eXtreme Scale. Quando una transazione è limitata ad una partizione singola, per impostazione predefinita è limitata ad una singola Java virtual machine e quindi ad un singolo computer server. Un server può completare un numero *M* di tali transazioni al secondo e, se si dispone di *N* computer, è possibile completare `M*N` transazioni al secondo. Se le proprie attività vengono incrementate ed è necessario raddoppiare il numero di transazioni al secondo, è possibile raddoppiare *N* acquistando altri computer. Quindi, è possibile soddisfare le richieste di capacità senza modificare l'applicazione, aggiornare l'hardware o portare l'applicazione fuori linea.

Oltre a consentire alla cache di scalare in modo così significativo, le transazioni su partizione singola incrementano al massimo la disponibilità della cache. Ciascuna transazione dipende solo da un computer. Anche in caso di malfunzionamento di uno degli altri `(N-1)` computer, il successo o il tempo di risposta della transazione non vengono modificati. Quindi, se vengono utilizzati 100 computer e si verifica un malfunzionamento di uno di essi, viene eseguito il rollback solo dell'1 percento delle transazioni in esecuzione al momento del malfunzionamento del server. In seguito al malfunzionamento del server, WebSphere eXtreme Scale assegna nuovamente le partizioni ospitate dal server malfunzionante agli altri 99 computer. Durante tale periodo, prima che l'operazione venga completata, gli altri 99 computer possono ancora completare le transazioni. Vengono bloccate solo le transazioni che coinvolgono le partizioni in fase di riassegnazione. Una volta completato il processo di failover, la cache può continuare a funzionare, completamente operativa, al 99 percento della propria capacità di trasmissione originale. Dopo che il server malfunzionante viene sostituito e restituito alla griglia, la cache ritorna al 100 percento della capacità di trasmissione.

Transazioni sulla griglia

In termini di prestazioni, disponibilità e scalabilità, le transazioni sulla griglia sono l'opposto delle transazioni su partizione singola. Le transazioni sulla griglia accedono a tutte le partizioni e quindi a tutti i computer nella configurazione. A ciascun computer nella griglia viene richiesto di ricercare alcuni dati e di restituire i risultati. La transazione non può essere completata fino a quando non hanno risposto tutti i computer e quindi il livello di prestazione dell'intera griglia è limitato dal computer più lento. L'aggiunta di altri computer non rende più veloce il computer più lento e quindi non migliora il livello di prestazione della cache.

Le transazioni sulla griglia hanno un effetto simile sulla disponibilità. Facendo riferimento all'esempio precedente, se vengono utilizzati 100 server e si verifica il malfunzionamento di un server, viene eseguito il rollback del 100 percento delle transazioni in esecuzione al momento del malfunzionamento del server. Dopo il malfunzionamento del server, WebSphere eXtreme Scale inizia a riassegnare le partizioni ospitate da tale server agli altri 99 computer. Durante questo periodo di tempo, prima che venga completato il processo di failover, la griglia non è in grado di elaborare alcuna di tali transazioni. Una volta completato il processo di failover, la cache può continuare a funzionare, ma con una capacità ridotta. Se ciascun computer nella griglia gestiva 10 partizioni, 10 dei rimanenti 99 computer ricevono almeno una partizione supplementare in seguito al processo di failover. L'aggiunta di una partizione supplementare incrementa il carico di lavoro di tale computer di almeno il 10 percento. Poiché il livello di prestazione della griglia è limitato alla velocità di trasmissione del computer più lento in una transazione sulla griglia, il livello di prestazione viene ridotto in media del 10 percento.

È preferibile utilizzare le transazioni su partizione singola invece delle transazioni su griglia per eseguire lo scale out con una cache di oggetti distribuita, ad alta disponibilità come WebSphere eXtreme Scale. L'ottimizzazione delle prestazioni di questi tipi di sistemi richiede l'utilizzo di tecniche differenti dalle metodologie relazionali tradizionali, ma è possibile convertire le transazioni su griglia in transazioni su partizione singola scalabili.

Migliori pratiche per la creazione di modelli di dati scalabili

Le migliori pratiche per la creazione di applicazioni scalabili con prodotti come WebSphere eXtreme Scale includono due categorie: principi fondamentali e suggerimenti per l'implementazione. I principi fondamentali sono le idee principali da catturare nella progettazione dei dati. Un'applicazione che non osserva tali principi non sarà in grado di scalare correttamente, anche per le proprie transazioni principali. I suggerimenti per l'implementazione vengono applicati per transazioni problematiche in un'applicazione altrimenti ben progettata che osserva i principi generali per i modelli di dati scalabili.

Principi fondamentali

Alcuni degli importanti mezzi di ottimizzazione della scalabilità sono concetti di base o principi da tenere in considerazione.

Duplicazione invece della normalizzazione

Il concetto principale da ricordare quando si utilizzano prodotti come WebSphere eXtreme Scale è che tali prodotti sono progettati per distribuire i dati su un numero elevato di computer. Se l'obiettivo è il completamento della maggior parte o di tutte le transazioni su una singola partizione, il progetto del modello di dati deve garantire che tutti i dati che potrebbero

essere richiesti dalla transazione si trovino nella partizione. Nella maggior parte dei casi, ciò è possibile solo duplicando i dati.

Ad esempio, considerare un'applicazione come una bacheca messaggi. Due transazioni molto importanti per una bacheca messaggi mostrano tutti i messaggi inviati da un determinato utente e tutti i messaggi relativi ad un determinato argomento. Innanzitutto, considerare il modo in cui tali transazioni funzionano con un modello di dati normalizzato che contiene un record utente, un record dell'argomento ed un record del messaggio che contiene il testo reale. Se i messaggi sono partizionati con i record utente, la visualizzazione degli argomenti diventa una transazione su griglia e viceversa. Gli argomenti e gli utenti non possono essere partizionati insieme perché hanno una relazione molti-a-molti.

Il modo migliore per consentire a questa bacheca messaggi di scalare è quello di duplicare i messaggi, memorizzando una copia con il record dell'argomento ed una copia con il record utente. Quindi, la visualizzazione dei messaggi da un utente è una transazione su partizione singola, la visualizzazione dei messaggi su un argomento è una transazione su partizione singola e l'aggiornamento o eliminazione di un messaggio è una transazione su due partizioni. Tutte e tre tali transazioni scaleranno linearmente con l'incremento del numero di computer nella griglia.

Scalabilità piuttosto che risorse

L'ostacolo maggiore da superare quando si considerano modelli di dati denormalizzati è l'impatto di tali modelli sulle risorse. La conservazione di due, tre o più copie di alcuni dati può dare l'impressione di utilizzare un numero troppo elevato di risorse per essere pratica. In questo scenario, ricordare quanto riportato di seguito: ogni anno, le risorse hardware diventano meno costose. Secondo, e più importante, WebSphere eXtreme Scale elimina la maggior parte dei costi nascosti associati alla distribuzione di ulteriori risorse.

Misurare le risorse in termini di costi piuttosto che in termini di hardware, come megabyte e processori. Gli archivi di dati che utilizzano dati relazionali normalizzati generalmente devono essere ubicati sullo stesso computer. Tale collocazione obbligatoria indica che è necessario acquistare un singolo computer enterprise di capacità maggiori piuttosto che diversi computer di capacità minori. Con l'hardware enterprise, non è raro il caso in cui un computer in grado di completare un milione di transazioni al secondo costi molto più della soma dei costi di 10 computer in grado, ciascuno, di eseguire 100000 transazioni al secondo.

Inoltre, esiste un costo aziendale relativo all'aggiunta di risorse. Un business crescente alla fine esaurisce le proprie capacità. Quando si esaurisce la capacità, è necessario passare ad un computer più potente e veloce oppure creare un secondo ambiente di produzione. In entrambi i casi, derivano ulteriori costi a causa di affari non realizzati o della gestione di una capacità quasi doppia rispetto a quella necessaria durante il periodo di transizione.

Con WebSphere eXtreme Scale, non è necessario chiudere l'applicazione per aggiungere capacità. Se i propri progetti per l'anno successivo richiedono una capacità superiore del 10 percento, incrementare del 10 percento il numero di computer nella griglia. È possibile incrementare tale percentuale senza rendere indisponibile l'applicazione e senza acquistare capacità supplementare.

Evitare le trasformazioni dei dati

Quando si utilizza WebSphere eXtreme Scale, i dati devono essere memorizzati in un formato direttamente utilizzabile dalla logica di business. L'utilizzo di un formato più primitivo per i dati rappresenta un costo. È necessario eseguire la trasformazione quando vengono eseguite la scrittura e la lettura dei dati. Con i database relazionali tale trasformazione è necessaria, in quanto i dati vengono resi persistenti su disco abbastanza frequentemente, ma con WebSphere eXtreme Scale, non è necessario eseguire tali trasformazioni. Per la maggior parte, i dati sono memorizzati nella memoria e possono essere memorizzati nel formato esatto richiesto dall'applicazione.

L'osservazione di questa semplice regola consente di denormalizzare i dati in base al primo principio. Il tipo più comune di trasformazione per i dati di business è rappresentato dalle operazioni JOIN, necessarie per convertire dati normalizzati in una serie di risultati che soddisfi le necessità dell'applicazione. La memorizzazione dei dati nel formato corretto evita, in modo implicito, l'esecuzione di tali operazioni JOIN e produce un modello di dati denormalizzato.

Eliminare le query illimitate

Anche se i dati sono strutturati correttamente, le query illimitate non vengono scalate correttamente. Ad esempio, non utilizzare una transazione che richiede un elenco di tutti gli elementi ordinati in base al valore. Questa transazione potrebbe funzionare inizialmente, quando il numero totale di elementi è uguale a 1000; quando, però, il numero totale di elementi raggiunge 10 milioni, la transazione restituisce tutti gli elementi. Se viene eseguita questa transazione, i due risultati più probabili sono la scadenza della transazione o la visualizzazione di un errore di memoria esaurita del client.

L'opzione migliore è quella di modificare la logica di business in modo che possano essere restituiti solo i primi 10 o 20 elementi. Tale modifica della logica rende gestibile la dimensione della transazione, indipendentemente dal numero di elementi presenti nella cache.

Definizione dello schema

Il vantaggio principale della normalizzazione dei dati consiste nel fatto che il sistema del database può occuparsi della congruenza dei dati in modo trasparente. Quando i dati vengono denormalizzati per la scalabilità, tale gestione della congruenza dei dati automatica non esiste più. Per garantire la congruenza dei dati, è necessario implementare un modello di dati che possa essere utilizzato nel livello dell'applicazione oppure come plug-in sulla griglia distribuita.

Considerare l'esempio relativo alla bacheca messaggi. Se una transazione rimuove un messaggio da un argomento, è necessario rimuovere il messaggio duplicato sul record utente. Senza un modello di dati, è possibile che uno sviluppatore scriva il codice dell'applicazione per rimuovere il messaggio dall'argomento, dimenticando di rimuovere il messaggio dal record utente. Tuttavia, se lo sviluppatore utilizzasse un modello di dati invece di interagire direttamente con la cache, il metodo removePost sul modello di dati potrebbe estrarre l'ID utente dal messaggio, ricercare il record utente e rimuovere il messaggio duplicato in modo trasparente.

In alternativa, è possibile implementare un listener che viene eseguito sulla partizione reale che rileva la modifica all'argomento e modifica automaticamente il record utente. Un listener può essere utile perché la modifica al record utente potrebbe essere eseguita in locale se la partizione dispone del record utente oppure, anche se il record utente si trova su una partizione differente, la transazione viene eseguita tra i server invece che tra client e server. È probabile che la connessione di rete tra i server sia più rapida rispetto alla connessione di rete tra il client ed il server.

Evitare i conflitti

Evitare scenari in cui è presente un contatore globale. La griglia non scala se un singolo record viene utilizzato un numero di volte sproporzionato in confronto agli altri record. Le prestazioni della griglia saranno limitate dalle prestazioni del computer che contiene il record indicato.

In queste situazioni, provare a suddividere il record, in modo che sia gestito per partizione. Ad esempio, considerare una transazione che restituisce il numero totale di voci nella cache distribuita. Invece di impostare il sistema in modo che tutte le operazioni di inserimento e rimozione accedono ad un singolo record, utilizzare, su ciascuna partizione, un listener che tiene traccia delle operazioni di inserimento e rimozione. Grazie alla traccia del listener, le operazioni di inserimento e rimozione possono diventare operazioni su partizione singola.

La lettura del contatore diventa un'operazione sulla griglia, ma per la maggior parte è già inefficiente come operazione sulla griglia perché le proprie prestazioni sono collegate a quelle del computer che contiene il record.

Suggerimenti per l'implementazione

Per ottenere la massima scalabilità, è possibile considerare i suggerimenti riportati di seguito.

Utilizzare gli indici di ricerca inversi

Considerare un modello di dati denormalizzato in modo appropriato in cui i record del cliente sono partizionati in base al numero ID del cliente. Tale metodo di partizionamento rappresenta la scelta logica in quanto quasi tutte le operazioni di business eseguite con il record del cliente utilizzano il numero ID del cliente. Tuttavia, un'importante transazione che non utilizza il numero ID del cliente è la transazione di login. Per il login, è più comune utilizzare nomi utente o indirizzi e-mail invece dei numeri ID del cliente.

L'approccio semplice allo scenario di login è quello di utilizzare una transazione sulla griglia per individuare il record del cliente. Come illustrato in precedenza, tale approccio non esegue la scalabilità.

L'opzione successiva potrebbe essere quella di eseguire il partizionamento sul nome utente o l'e-mail. Questa opzione non è pratica, in quanto tutte le operazioni basate sull'ID cliente diventano transazioni sulla griglia. Inoltre, i clienti del sito potrebbero desiderare di modificare il proprio nome utente o indirizzo e-mail. I prodotti come WebSphere eXtreme Scale richiedono che il valore utilizzato per partizionare i dati resti costante.

La soluzione corretta consiste nell'utilizzo di un indice di ricerca inverso. Con WebSphere eXtreme Scale, è possibile creare una cache nella stessa griglia distribuita della cache che contiene tutti i record utente. Questa

cache è altamente disponibile, partizionata e scalabile. Tale cache può essere utilizzata per associare un nome utente o un indirizzo e-mail ad un ID cliente. Questa cache converte l'operazione di login in un'operazione su due partizioni invece di un'operazione sulla griglia. Tale scenario non è pratico come una transazione su partizione singola, ma il livello di prestazione viene scalato linearmente con l'aumento del numero di computer.

Calcolo al momento della scrittura

I valori calcolati comunemente, come le medie o i totali, possono essere costosi da creare perché tali operazioni generalmente richiedono la lettura di un numero di voci elevato. Poiché nella maggior parte delle applicazioni le letture sono più comuni delle scritture, è preferibile calcolare tali valori al momento della scrittura e memorizzare il risultato nella cache. Questa operazione rende le operazioni di scrittura più rapide e scalabili.

Campi facoltativi

Considerare un record utente che contenga un numero di telefono dell'ufficio, di casa e mobile. Per ciascun utente, potrebbero essere definiti tutti, nessuno o qualsiasi combinazione di tali numeri. Se i dati fossero normalizzati, esisterebbero una tabella utente ed una tabella dei numeri telefonici. I numeri telefonici di un determinato utente potrebbero essere individuati utilizzando un'operazione JOIN tra le due tabelle.

La denormalizzazione di tale record non richiede la duplicazione dei dati, perché la maggior parte degli utenti non condivide i numeri di telefono. Al contrario, deve essere possibile che alcuni alloggiamenti siano vuoti nel record utente. Invece di utilizzare una tabella dei numeri di telefono, aggiungere tre attributi a ciascun record utente, uno per ciascun tipo di numero di telefono. Tale aggiunta di attributi elimina l'operazione JOIN e converte l'operazione di ricerca di un numero di telefono per un utente in un'operazione su partizione singola.

Posizionamento di relazioni molti-a-molti

Considerare un'applicazione che tenga traccia dei prodotti e dei negozi in cui i prodotti vengono venduti. Un singolo prodotto viene venduto in molti negozi ed un singolo negozio vende molti prodotti. Si supponga che tale applicazione tracci 50 grandi venditori. Ciascun prodotto viene venduto in un massimo di 50 negozi, ciascuno dei quali vende migliaia di prodotti.

Conservare un elenco dei negozi all'interno dell'entità del prodotto (disposizione A) invece di conservare un elenco dei prodotti all'interno di ciascuna entità del negozio (disposizione B). Analizzando alcune delle transazioni che questa applicazione deve eseguire, è possibile comprendere per quale motivo la disposizione A è più scalabile.

Per prima cosa, si analizzino gli aggiornamenti. Con la disposizione A, la rimozione di un prodotto dall'inventario di un negozio blocca l'entità del prodotto. Se la griglia contiene 10000 prodotti, per eseguire l'aggiornamento è necessario bloccare solo 1/10000 della griglia. Con la disposizione B, la griglia contiene solo 50 negozi, per cui per completare l'aggiornamento è necessario bloccare solo 1/50 della griglia. Quindi, anche se entrambe queste operazioni possono essere considerate operazioni su partizione singola, la disposizione A esegue lo scale out in modo più efficiente.

Ora, considerando le letture con la disposizione A, la ricerca dei negozi in cui viene venduto un prodotto è una transazione su partizione singola che scala ed è rapida perché la transazione trasmette solo una piccola quantità di dati. Con la disposizione B, la transazione diventa una transazione sulla griglia, perché è necessario accedere a ciascuna entità del negozio per verificare se il prodotto viene venduto in tale negozio, rivelando un enorme vantaggio per le prestazioni per la disposizione A.

Scalabilità con dati normalizzati

Un utilizzo valido delle transazioni sulla griglia è quello di scalare l'elaborazione dei dati. Se una griglia dispone di 5 computer e viene distribuita una transazione sulla griglia che esamina circa 100000 record su ciascun computer, la transazione esamina 500000 record. Se il computer più lento nella griglia è in grado di eseguire 10 di tali transazioni al secondo, la griglia è in grado di esaminare 5000000 record al secondo. Se i dati nella griglia vengono raddoppiati, ciascun computer deve esaminare 200000 record e ciascuna transazione esamina 1000000 record. Tale incremento di dati riduce il livello di prestazione del computer più lento a 5 transazioni al secondo, riducendo di conseguenza livello di prestazione della griglia a 5 transazioni al secondo. La griglia esamina 5000000 di record al secondo.

In questo scenario, raddoppiando il numero di computer, ogni computer può tornare al proprio precedente carico di lavoro esaminando 100000 record, consentendo al computer più lento di elaborare 10 di tali transazioni al secondo. Il livello di prestazione della griglia resta invariato a 10 richieste al secondo, ma ora ciascuna transazione elabora 1000000 record, per cui la griglia ha raddoppiato la propria capacità in termini di elaborazione dei record, portandola a 10000000 al secondo.

Per le applicazioni come i motori di ricerca, che devono eseguire operazioni di scalabilità in termini di elaborazione dei dati per adattarsi all'incremento di Internet ed in termini di livello di prestazione per adattarsi al numero sempre crescente di utenti, è necessario creare più griglie, con una condivisione delle richieste sulle griglie. Se è necessario aumentare il livello di prestazione, aggiungere altri computer ed aggiungere un'altra griglia alle richieste di servizio. Se è necessario aumentare l'elaborazione dei dati, aggiungere altri computer e tenere costante il numero di griglie.

Capitolo 7. Panoramica sulla sicurezza

WebSphere eXtreme Scale può proteggere l'accesso ai dati, incluso consentire l'integrazione con provider della sicurezza esterni.

Nota: In un archivio dati esistente non in cache come un database, è probabile che vi siano funzioni integrate di sicurezza che potrebbe non essere necessario configurare o abilitare attivamente. Tuttavia, dopo aver inserito i dati nella cache con eXtreme Scale, è necessario prendere in considerazione l'importante situazione che ne risulta, ovvero che le funzioni di sicurezza del backend non sono più attive. È possibile configurare la sicurezza eXtreme Scale sui livelli necessari, in modo che sia protetta anche la nuova architettura dei dati inserita nella cache.
Di seguito è riportato un breve riepilogo delle funzioni di sicurezza di eXtreme Scale. Per informazioni più dettagliate sulla configurazione della sicurezza, consultare *Guida alla gestione* e *Guida alla programmazione*.

Concetti base per la sicurezza distribuita

La sicurezza eXtreme Scale distribuita si basa su tre concetti chiave:

Autenticazione attendibile
: La possibilità di stabilire l'identità del richiedente. WebSphere eXtreme Scale supporta sia l'autenticazione client-server che server-server.

Autorizzazione
: La possibilità di fornire autorizzazioni per concedere i diritti di accesso al richiedente. WebSphere eXtreme Scale supporta autorizzazioni diverse per varie operazioni.

Trasporto sicuro
: La trasmissione sicura dei dati su una rete. WebSphere eXtreme Scale supporta i protocolli TLS/SSL (Transport Layer Security/Secure Sockets Layer).

Autenticazione

WebSphere eXtreme Scale supporta un framework client server distribuito. È disponibile un'infrastruttura di sicurezza client server per proteggere l'accesso ai server eXtreme Scale. Ad esempio, quando viene richiesta l'autenticazione dal server eXtreme Scale, un client eXtreme Scale deve fornire le credenziali per effettuare l'autenticazione con il server. Queste credenziali possono essere una coppia nome utente e password, un certificato client, un ticket Kerberos o dati presentati in un formato concordato dal client e dal server.

Autorizzazione

Le autorizzazioni WebSphere eXtreme Scale si basano su soggetti e autorizzazioni. È possibile utilizzare JAAS (Java Authentication and Authorization Services) per autorizzare l'accesso oppure è possibile collegare un approccio personalizzato, ad esempio TAM /Tivoli Access Manager). per gestire le autorizzazioni. È possibile fornire le seguenti autorizzazioni ad un client o a un gruppo:

Autorizzazione per la mappa
: Per eseguire operazioni insert, read, update, evict o delete sulle mappe.

Autorizzazione per ObjectGrid
Per eseguire query di oggetti o entità e query di flusso sugli oggetti ObjectGrid.

Autorizzazione per l'agent DataGrid
Consente la distribuzione degli agent DataGrid su un ObjectGrid.

Autorizzazione per la mappa lato server
Per replicare una mappa server sul lato client o creare un indice dinamico per la mappa server.

Autorizzazione all'amministrazione
Per eseguire attività di amministrazione.

Sicurezza del trasporto

Per proteggere la comunicazione client server, WebSphere eXtreme Scale supporta TLS/SSL. Questi protocolli forniscono la sicurezza del livello di trasporto con autenticità, integrità e riservatezza per una connessione sicura tra un client e un server eXtreme Scale.

Sicurezza della griglia

In un ambiente sicuro, un server deve esser in grado di verificare l'autenticità di un altro server. A questo scopo WebSphere eXtreme Scale utilizza un meccanismo di stringhe di chiavi segrete condivise. Questo meccanismo di chiavi segrete è simile ad una password condivisa. Tutti i server eXtreme Scale concordano una stringa segreta condivisa. Quando un server si unisce alla griglia, gli viene richiesta la stringa segreta. Se la stringa segreta del server che si unisce corrisponde a quella del server principale, il server può unirsi alla griglia. Diversamente, la richiesta di unione viene respinta.

L'invio di un segreto in testo chiaro non è sicuro. L'infrastruttura della sicurezza di eXtreme Scale fornisce un plug-in SecureTokenManager per consentire al server di proteggere il segreto prima di inviarlo. È possibile scegliere come implementare l'operazione sicura. WebSphere eXtreme Scale fornisce un'implementazione, in cui viene implementata l'operazione sicura che codifica e firma il segreto.

Sicurezza JMX (Java Management Extensions) in una topologia di distribuzione dinamica

La sicurezza JMX MBean è supportata in tutte le versioni di eXtreme Scale. I client degli MBean del server catalogo e degli MBean del server contenitore possono essere autenticati, e l'accesso alle operazioni MBean può essere rafforzato.

Sicurezza eXtreme Scale locale

La sicurezza eXtreme Scale locale è diversa dal modello di eXtreme Scale distribuito in quanto l'applicazione crea direttamente l'istanza ed utilizza un'istanza ObjectGrid. Le istanze dell'applicazione e di eXtreme Scale sono sulla stessa JVM /Java virtual machine). Poiché in questo modello non esiste un concetto client-server, l'autenticazione non è supportata. Le applicazioni devono gestire la propria autenticazione, e quindi passare l'oggetto Subject autenticato a eXtreme Scale. Tuttavia, il meccanismo di autorizzazione utilizzato per il modello di programmazione eXtreme Scale locale è uguale a quello utilizzato per il modello client-server.

Configurazione e programmazione

Per ulteriori informazioni sulla configurazione e la programmazione della sicurezza, consultare *Guida alla gestione* e *Guida alla programmazione*.

Capitolo 8. Panoramica servizi dati REST

Il servizio dati REST di WebSphere eXtreme Scale è un servizio HTTP Java compatibile con Microsoft WCF Data Services (formalmente ADO.NET Data Services) ed implementa Open Data Protocol (OData). Microsoft WCF Data Services è compatibile con questa specifica quando si utilizza Visual Studio 2008 SP1 e .NET Framework 3.5 SP1.

Requisiti di compatibilità

Il servizio dati REST consente a qualsiasi client HTTP di accedere ad una griglia di dati. Il servizio dati REST è compatibile con il supporto WCF Data Services fornito con Microsoft .NET Framework 3.5 SP1. Le applicazioni RESTful possono essere sviluppate con la ricca strumentazione fornita da Microsoft Visual Studio 2008 SP1. La figura fornisce una panoramica del modo in cui il WCF Data Services interagisce con client e database.

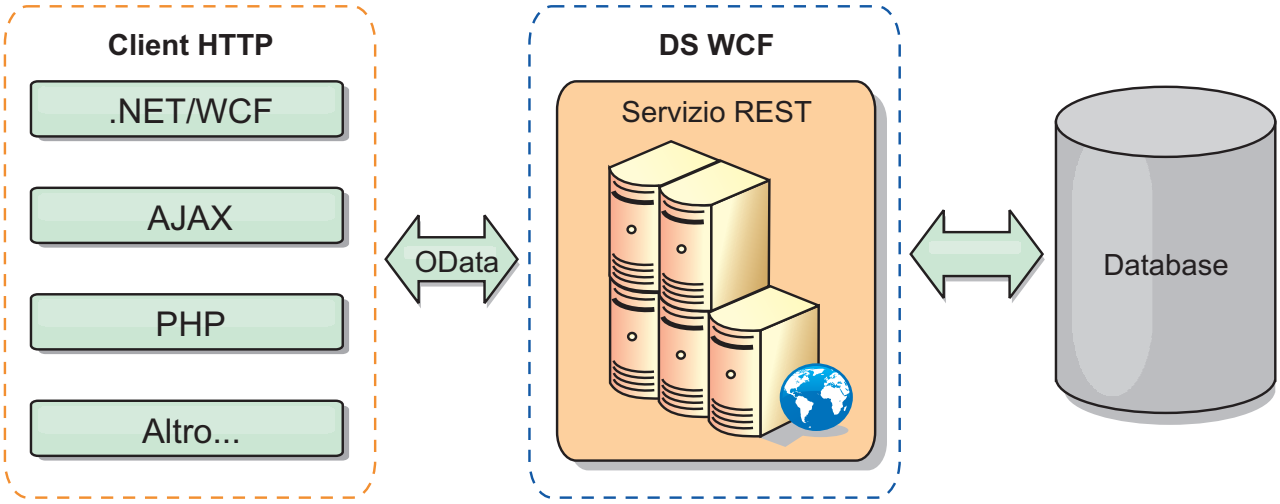


Figura 41. WCF Data Services di Microsoft

WebSphere eXtreme Scale include una serie di API ricche di funzioni per i client Java. Come mostrato nella seguente figura, il servizio dati REST è un gateway tra i client HTTP e la griglia WebSphere eXtreme Scale che comunica con la griglia tramite un client WebSphere eXtreme Scale. Il servizio dati REST è un servlet Java che consente distribuzioni flessibili per piattaforme Java comuni, piattaforme Enterprise Edition (JEE), come WebSphere Application Server. Il servizio dati REST comunica con la griglia WebSphere eXtreme Scale utilizzando le API WebSphere eXtreme Scale Java. Consente ai client di WCF Data Services o a qualunque altro client in grado di farlo, di comunicare con HTTP e XML.

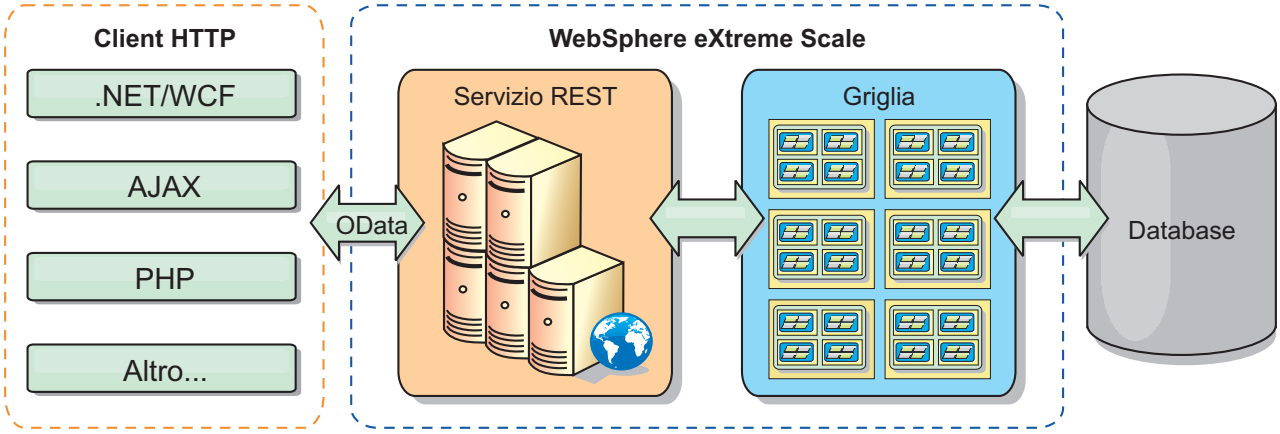


Figura 42. Servizio dati REST WebSphere eXtreme Scale

Fare riferimento a "Esempio e supporto didattico del servizio dati REST" a pagina 219, oppure utilizzare i seguenti collegamenti per saperne di più su WCF Data Services.

- 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

Funzioni

Questa versione del servizio dati REST di eXtreme Scale supporta le seguenti funzioni:

- Modeling automatico delle entità API di eXtreme Scale EntityManager come entità di WCF Data Services, che include il seguente supporto:
  - Conversione del tipo dati Java in tipo EDM (Entity Data Model)
  - Supporto associazione entità
  - Supporto associazione root schema e chiave, richiesto per le griglie di dati partizionati.

  Per ulteriori informazioni, consultare Entity model.
- Formato payload dati XML Atom Publish Protocol (AtomPub o APP) e JASON (JavaScript™ Object Notation).
- Operazioni CURD (Create, Read, Update e Delete) tramite i rispettivi metodi di richiesta HTTP: POST, GET, PUT e DELETE. Inoltre, è supportata l'estensione Microsoft MERGE.
- Query semplici, con l'uso di filtri
- Richiamo batch e richieste di serie di modifiche
- Griglia partizionata per l'alta disponibilità.
- Interoperabilità con i client delle API di eXtreme Scale EntityManager
- Supporto per server Web JEE standard
- Contemporaneità ottimistica

- Autorizzazione e autenticazione utente tra il servizio dati REST e la griglia dati di eXtreme Scale

Problemi noti e limitazioni

- Le richieste con tunnel non sono supportate.

Capitolo 9. Panoramica sull'integrazione Spring framework

Spring è un noto framework per lo sviluppo di applicazioni Java. WebSphere eXtreme Scale fornisce il supporto per consentire a Spring la gestione delle transazioni eXtreme Scale e la configurazione dei client e dei server che compongono la griglia di dati in memoria distribuita.

Transazioni native gestite da Spring

Spring fornisce transazioni gestite da contenitori che sono simili ad un server delle applicazioni Java Platform, Enterprise Edition. Tuttavia, il meccanismo Spring può collegare implementazioni diverse. WebSphere eXtreme Scale fornisce l'integrazione del gestire transazioni che consente a Spring di gestire i cicli di vita delle transazioni ObjectGrid. Per i dettagli consultare le informazioni relative alle transazioni native in *Guida alla programmazione*.

Bean di estensione gestiti da Spring e supporto spazio dei nomi

eXtreme Scale si integra con Spring per consentire i bean in stile Spring definiti per i punti di estensione o i plug-in. Questa funzione fornisce configurazioni più sofisticate e e una maggiore flessibilità per la configurazione dei punti di estensione.

In aggiunta ai bean di estensione gestiti da Spring, eXtreme Scale fornisce uno spazio dei nomi Spring denominato "objectgrid". I bean e le implementazioni integrate vengono predefiniti in questo spazio dei nomi e questo comporta che è più facile per gli utenti configurare eXtreme Scale. Per ulteriori dettagli su questi argomenti ed un esempio di come avviare un server contenitore eXtreme Scale utilizzando le configurazioni Spring, consultare Bean di estensione Spring e supporto spazio dei nomi.

Supporto dell'ambito shard (frammento)

Con la configurazione Spring in stile tradizionale, un bean ObjectGrid può essere un tipo singleton o un tipo prototipo. ObjectGrid supporta anche un nuovo ambito denominato ambito "shard" (frammento). Se è definito un bean come ambito shard, viene creato un unico bean per frammento. Tutte le richieste di bean con un ID o più ID che corrispondono a quella definizione di bean nello stesso frammento avranno come risultato un'istanza di quel bean specifico restituita dal contenitore Spring.

Il seguente esempio mostra che è definito un bean com.ibm.ws.objectgrid.jpa.plugins.JPAPropFactoryImpl con l'ambito impostato su shard. Quindi, vene creata una sola istanza della classe JPAPropFactoryImpl per frammento.

```
<bean id="jpaPropFactory" class="com.ibm.ws.objectgrid.jpa.plugins.JPAPropFactoryImpl" scope="shard" />
```

Spring Web Flow

Spring Web Flow memorizza lo stato della sessione nella sessione HTTP per impostazione predefinita. Se un'applicazione Web è configurata per utilizzare eXtreme Scale per la gestione delle sessioni, viene utilizzata automaticamente da

Spring per memorizzare lo stato e viene resa tollerante agli errori nello stesso modo della sessione.

Creazione package

Le estensioni eXtreme Scale Spring sono nel file `ogspring.jar`. Questo file JAR (Java archive) deve essere nel percorso di classe, perché il supporto Spring funzioni. Se un'applicazione JEE in esecuzione in un WebSphere Extended Deployment ha potenziato WebSphere Application Server Network Deployment, l'applicazione deve posizionare il file `spring.jar` ed i relativi file associati nei moduli EAR (Enterprise Archive). È necessario inoltre posizionare il file `ogspring.jar` nella stessa ubicazione.

Capitolo 10. Supporti didattici, esempi e campioni

Sono disponibili diversi supporti didattici, esempi e campioni di WebSphere eXtreme Scale.

Supporti didattici

Attualmente sono disponibili i seguenti supporti didattici.

- Supporto didattico per ObjectMap
- "Supporto didattico gestore entità: panoramica"
-
-

Esempi

Gli argomenti in basso descrivono le funzioni principali di WebSphere eXtreme Scale.

- Consultare l'esempio di API Data Grid in *Guida alla programmazione*
- Consultare i dettagli sulla configurazione di distribuzioni locali in *Guida alla gestione*

Campioni di esempio

Alcuni campioni di esempio per illustrare come utilizzare le API ObjectGrid nei vari ambienti vengono forniti con il prodotto WebSphere eXtreme Scale.

Articoli con supporti didattici ed esempi

Tabella 15. Articoli disponibili per funzione

Articolo	**Funzioni**
Building grid-ready applications	API ObjectMap, API EntityManager, Query, Agent, Java SE e EE, Statistiche, Partizionamento, Amministrazione/ Operazioni, Eclipse
Scalable grid-style computing and data processing	API EntityManager, Agent
Building a scalable, resilient, high-performance database alternative	API ObjectMap, Replica, Partitionamento, Amministrazione/Operazioni, Eclipse
Enhancing xsadmin for WebSphere eXtreme Scale	Amministrazione
Redbook: User's Guide	Tutti gli argomenti

Supporto didattico gestore entità: panoramica

Il Supporto didattico per il gestore entità mostra come utilizzare WebSphere eXtreme Scale per memorizzare informazioni sull'ordine in un sito Web. È possibile creare una semplice applicazione Java Platform, Standard Edition 5 che utilizza eXtreme Scale in memoria, locale. Le entità utilizzano annotazioni e generics di Java SE 5.

Prima di iniziare

Assicurarsi di aver soddisfatto i seguenti requisiti prima di iniziare il supporto didattico:

- È necessario avere Java SE 5.
- È necessario avere il file `objectgrid.jar` nel proprio percorso di classe.

Supporto didattico gestore entità: creazione di una classe di entità

Il primo passo del supporto didattico per il gestore entità mostra come creare un ObjectGrid locale con un'entità creando una classe di entità, registrando il tipo di entità tramite eXtreme Scale e memorizzando un'istanza di entità nella cache.

Informazioni su questa attività

Procedura

1. Creare l'oggetto Order. Per identificare l'oggetto come un'entità ObjectGrid, aggiungere l'annotazione @Entity. Quando si aggiunge questa annotazione, tutti gli attributi serializzabili nell'oggetto vengono automaticamente resi persistenti in eXtreme Scale, tranne se si utilizzano le annotazioni sugli attributi per sostituire gli attributi. L'attributo orderNumber viene annotato con @Id per indicare che questo attributo è la chiave primaria. Un esempio di un oggetto Order viene riportato di seguito:

   Order.java

   ```
   @Entity
   public class Order
   {
       @Id String orderNumber;
       Date date;
       String customerName;
       String itemName;
       int quantity;
       double price;
   }
   ```

2. Eseguire l'applicazione eXtreme Scale Hello World per mostrare le operazioni di entità. Il seguente programma di esempio può essere lanciato in modalità autonoma per mostrare le operazioni di entità. Utilizzare il programma in un progetto Eclipse Java che contenga il file `objectgrid.jar` aggiunto al percorso di classe. Segue un esempio di di una semplice applicazione Hello world che utilizza eXtreme Scale:

   Application.java

   ```
   package emtutorial.basic.step1;

   import com.ibm.websphere.objectgrid.ObjectGrid;
   import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
   import com.ibm.websphere.objectgrid.Session;
   import com.ibm.websphere.objectgrid.em.EntityManager;

   public class Application
   {

       static public void main(String [] args)
           throws Exception
       {
           ObjectGrid og =
       ObjectGridManagerFactory.getObjectGridManager().createObjectGrid();
           og.registerEntities(new Class[] {Order.class});

           Session s = og.getSession();
           EntityManager em = s.getEntityManager();

           em.getTransaction().begin();

           Order o = new Order();
   ```

```
        o.customerName = "John Smith";
        o.date = new java.util.Date(System.currentTimeMillis());
        o.itemName = "Widget";
        o.orderNumber = "1";
        o.price = 99.99;
        o.quantity = 1;

        em.persist(o);
        em.getTransaction().commit();

        em.getTransaction().begin();
        o = (Order)em.find(Order.class, "1");
        System.out.println("Found order for customer: " + o.customerName);
        em.getTransaction().commit();
    }
}
```

Questa applicazione di esempio esegue le seguenti operazioni:

a. Inizializza un eXtreme Scale locale con un nome generato automaticamente.
b. Registra le classi di entità con l'applicazione utilizzando le API registerEntities, sebbene l'utilizzo delle API registerEntities non sia sempre necessario.
c. Richiama una Sessione e un riferimento al gestore entità per la Sessione.
d. Associa ogni sessione eXtreme Scale ad un singolo EntityManager e EntityTransaction. EntityManager a questo punto viene utilizzata.
e. Il metodo registerEntities crea un oggetto BackingMap chiamato Order e associa i metadati per l'oggetto Order con l'oggetto BackingMap. Questi metadati includono gli attributi chiave e non chiave, insieme ai nome e tipi di attributo.
f. Inizia una transazione e crea un'istanza Order. La transazione viene popolata con alcuni valori e resa persistente tramite il metodo EntityManager.persist, che identifica l'entità come in attesa di essere inclusa nella mappa ObjectGrid associata.
g. Viene eseguito quindi il commit della transazione e l'entità viene inclusa in ObjectMap.
h. Viene creata un'altra transazione e l'oggetto Order viene richiamato utilizzando la chiave 1. Il cast del tipo sul metodo EntityManager.find è necessario, perché la capability dei nomi generici di Java SE 5 non viene utilizzata per assicurare che il file `objectgrid.jar` lavori su Java SE 1.4 Java virtual machine e successive.

Supporto didattico gestore entità: creazione relazioni di entità

Creare una relazione semplice tra entità creando due classi di entità con una relazione, registrando le entità con l'ObjectGrid e memorizzando le istanze di entità nella cache.

Procedura

1. Creare l'entità customer, che viene utilizzata per memorizzare i dettagli del cliente indipendentemente dall'oggetto Order. Segue un esempio di entità customer:

Customer.java

```
@Entity
public class Customer
{
    @Id String id;
    String firstName;
    String surname;
    String address;
    String phoneNumber;
}
```

Questa classe include le informazioni sul cliente come nome, indirizzo e numero di telefono.

2. Creare l'oggetto Order, che è simile all'oggetto Order nell'argomento “Supporto didattico gestore entità: creazione di una classe di entità” a pagina 186. Segue un esempio di oggetto Order:

   Order.java

   ```
   @Entity
   public class Order
   {
       @Id String orderNumber;
       Date date;
       @ManyToOne(cascade=CascadeType.PERSIST) Customer customer;
       String itemName;
       int quantity;
       double price;
   }
   ```

   In questo esempio, un riferimento a un oggetto Customer sostituisce l'attributo customerName. Il riferimento ha un'annotazione che indica una relazione molti a uno. Una relazione molti a uno indica che ogni ordine ha un cliente, ma più ordini possono fare riferimento allo stesso cliente. Il modificatore di annotazione a cascata indica che se il gestore entità rende persistente l'oggetto Order, deve rendere persistente anche l'oggetto Customer. Se si sceglie di non impostare l'opzione di persistenza a cascata, che è l'opzione predefinita, è necessario rendere persistente manualmente l'oggetto Customer con l'oggetto Order.

3. Utilizzando le entità, definire le mappe per l'istanza di ObjectGrid. Ogni mappa viene definita per una specifica entità e un'entità viene denominata Order e l'altra Customer. La seguente applicazione di esempio illustra come memorizzare e recuperare un ordine cliente:

   Application.java

   ```
   public class Application
   {
       static public void main(String [] args)
           throws Exception
       {
           ObjectGrid og =
        ObjectGridManagerFactory.getObjectGridManager().createObjectGrid();
           og.registerEntities(new Class[] {Order.class});

           Session s = og.getSession();
           EntityManager em = s.getEntityManager();

           em.getTransaction().begin();

           Customer cust = new Customer();
           cust.address = "Main Street";
           cust.firstName = "John";
           cust.surname = "Smith";
           cust.id = "C001";
           cust.phoneNumber = "5555551212";

           Order o = new Order();
           o.customer = cust;
           o.date = new java.util.Date();
           o.itemName = "Widget";
           o.orderNumber = "1";
           o.price = 99.99;
           o.quantity = 1;

           em.persist(o);
           em.getTransaction().commit();

           em.getTransaction().begin();
           o = (Order)em.find(Order.class, "1");
           System.out.println("Found order for customer: "
   ```

```
    + o.customer.firstName + " " + o.customer.surname);
       em.getTransaction().commit();
    }
}
```

Questa applicazione è simile all'applicazione di esempio del passo precedente. Nel precedente esempio, viene registrata solo una singola classe Order. WebSphere eXtreme Scale rileva e automaticamente include il riferimento all'entità Customer e viene inserito il riferimento a un'istanza Customer per `John Smith` dal nuovo oggetto Order. Di conseguenza, il nuovo cliente viene reso automaticamente persistente, in quanto la relazione tra i due ordini include il modificatore a cascata che richiede che ogni oggetto sia persistente. Quando viene trovato l'oggetto Order, il gestore entità trova automaticamente l'oggetto Customer associato e inserisce un riferimento all'oggetto.

Supporto didattico gestore entità: schema entità Order

Creare quattro entità utilizzando sia relazioni singole che bidirezionali, elenchi ordinati e relazioni chiavi esterne. Le API EntityManager vengono utilizzate per rendere persistenti e trovare le entità. Sulla base delle entità Order e Customer incontrate nelle sezioni precedenti del supporto didattico, questo passo del supporto didattico aggiunge due ulteriori entità: le entità Item e OrderLine.

Informazioni su questa attività

Figura 43. Schema entità Order. Un'entità Order ha un riferimento ad un cliente e zero o più OrderLines. Ogni entità OrderLine contiene un riferimento a un singolo elemento e include la quantità ordinata.

Procedura

1. Creare un'entità cliente che è simile agli esempi precedenti.

   Customer.java
   ```
   @Entity
   public class Customer
   {
       @Id String id;
       String firstName;
       String surname;
       String address;
       String phoneNumber;
   }
   ```

2. Creare un'entità Item, che detiene le informazioni su un prodotto incluso nell'inventario dell'archivio, come la descrizione, la quantità e il prezzo del prodotto.

   Item.java
   ```
   @Entity
   public class Item
   {
       @Id String id;
       String description;
       long quantityOnHand;
       double price;
   }
   ```

3. Creare l'entità OrderLine. Ogni Order ha zero o più OrderLines, che identifica la quantità di ogni elemento nell'ordine. La chiave per OrderLine è una chiave composta da Order che possiede OrderLine e un numero intero che assegna un numero alla riga dell'ordine. Aggiungere un modificatore di persistenza a cascata ad ogni relazione sulle proprie entità.

OrderLine.java

```
@Entity
public class OrderLine
{
    @Id @ManyToOne(cascade=CascadeType.PERSIST) Order order;
    @Id int lineNumber;
    @OneToOne(cascade=CascadeType.PERSIST) Item item;
    int quantity;
    double price;
}
```

4. Creare un oggetto Order finale, che abbia un riferimento a Customer per l'ordine e un insieme di oggetti OrderLine.

   Order.java

   ```
   @Entity
   public class Order
   {
       @Id String orderNumber;
       java.util.Date date;
       @ManyToOne(cascade=CascadeType.PERSIST) Customer customer;
       @OneToMany(cascade=CascadeType.ALL, mappedBy="order")
      @OrderBy("lineNumber") List<OrderLine> lines;
   }
   ```

   ALL a cascata viene utilizzato come modificatore per le righe. Questo modificatore segnala l'EntityManager rendere a cascata sia l'operazione PERSIST che REMOVE. Ad esempio, se l'entità Order viene resa persistente o rimossa, anche tutte le entità OrderLine vengono rese persistenti o rimosse.

   Se un'entità OrderLine viene rimossa dall'elenco righe nell'oggetto Order, il riferimento viene interrotto. Tuttavia, l'entità OrderLine non viene rimossa dalla cache. È necessario utilizzare l'API di rimozione EntityManager per rimuovere le entità dalla cache. L'operazione REMOVE non viene utilizzata nell'entità Customer o nell'entità Item da OrderLine. Di conseguenza, l'entità Customer rimane anche se Order o Item vengono rimossi quando OrderLine viene eliminato.

   Il modificatore mappedBy indica una relazione inversa con l'entità di destinazione. Il modificatore identifica l'attributo che nell'entità di destinazione fa riferimento all'entità di origine e il lato appartenente alla relazione uno-a-uno o molti-a-molti. Di solito, è possibile omettere il modificatore. Tuttavia, un errore viene visualizzato per indicare che deve essere specificato se WebSphere eXtreme Scale non può scoprirlo automaticamente. Un'entità OrderLine che contiene due attributi di tipo Order in una relazione molti-a-uno di solito causa quest'errore.

   L'annotazione @OrderBy specifica l'ordine in cui ogni entità OrderLine deve essere nell'elenco righe. Se l'annotazione non viene specificata, le righe vengono visualizzate con un ordine arbitrario. Sebbene le righe vengano aggiunte all'entità Order lanciando ArrayList, che mantiene l'ordine, EntityManager non riconosce necessariamente l'ordine. Quando si lancia il metodo find per recuperare l'oggetto Order dalla cache, l'oggetto list non è un oggetto ArrayList.

5. Creare l'applicazione. Il seguente esempio illustra l'oggetto Order finale, che ha un riferimento a Customer per l'ordine e un insieme di oggetti OrderLine.
   a. Trovare gli elementi da ordinare, che diventano successivamente entità Managed.
   b. Creare OrderLine e collegarla ad ogni Item.
   c. Creare Order e associarlo ad ogni OrderLine e al cliente.
   d. Rendere l'ordine persistente, che automaticamente rende persistente ogni OrderLine.

e. Eseguire il commit della transazione, che separa ogni entità e sincronizza lo stato delle entità con la cache.

f. Stampare le informazioni sull'ordine. Le entità OrderLine vengono automaticamente ordinate tramite l'ID OrderLine.

Application.java

```
static public void main(String [] args)
        throws Exception
    {
        ...

        // Add some items to our inventory.
        em.getTransaction().begin();
        createItems(em);
        em.getTransaction().commit();

        // Create a new customer with the items in his cart.
        em.getTransaction().begin();
        Customer cust = createCustomer();
        em.persist(cust);

        // Create a new order and add an order line for each item.
        // Each line item is automatically persisted since the
    // Cascade=ALL option is set.
        Order order = createOrderFromItems(em, cust, "ORDER_1",
     new String[]{"1", "2"}, new int[]{1,3});
        em.persist(order);
        em.getTransaction().commit();

        // Print the order summary
        em.getTransaction().begin();
        order = (Order)em.find(Order.class, "ORDER_1");
        System.out.println(printOrderSummary(order));
        em.getTransaction().commit();
    }

    public static Customer createCustomer() {
        Customer cust = new Customer();
        cust.address = "Main Street";
        cust.firstName = "John";
        cust.surname = "Smith";
        cust.id = "C001";
        cust.phoneNumber = "5555551212";
        return cust;
    }

    public static void createItems(EntityManager em) {
        Item item1 = new Item();
        item1.id = "1";
        item1.price = 9.99;
        item1.description = "Widget 1";
        item1.quantityOnHand = 4000;
        em.persist(item1);

        Item item2 = new Item();
        item2.id = "2";
        item2.price = 15.99;
        item2.description = "Widget 2";
        item2.quantityOnHand = 225;
        em.persist(item2);

    }

    public static Order createOrderFromItems(EntityManager em,
   Customer cust, String orderId, String[] itemIds, int[] qty) {
```

```
        Item[] items = getItems(em, itemIds);

        Order order = new Order();
        order.customer = cust;
        order.date = new java.util.Date();
        order.orderNumber = orderId;
        order.lines = new ArrayList<OrderLine>(items.length);
    for(int i=0;i<items.length;i++){
      OrderLine line = new OrderLine();
            line.lineNumber = i+1;
            line.item = items[i];
            line.price = line.item.price;
            line.quantity = qty[i];
            line.order = order;
            order.lines.add(line);
        }
        return order;
    }

    public static Item[] getItems(EntityManager em, String[] itemIds) {
        Item[] items = new Item[itemIds.length];
        for(int i=0;i<items.length;i++){
    items[i] = (Item) em.find(Item.class, itemIds[i]);
        }
        return items;
    }
```

Il passo successivo consiste nel cancellare un'entità. L'interfaccia EntityManager dispone di un metodo remove che contrassegna un oggetto come eliminato. L'applicazione deve rimuovere l'entità da tutti gli insiemi di relazioni prima di chiamare il metodo remove. Modificare i riferimenti e lanciare il metodo remove o em.remove(object), come passo finale.

Supporto didattico gestore entità: aggiornamento elementi

Se si desidera modificare un'entità, è possibile trovare l'istanza, aggiornare l'istanza e tutte le entità citate ed eseguire il commit della transazione.

Procedura

Aggiornare gli elementi. Il seguente esempio mostra in che modo trovare l'istanza Order, modificare l'istanza e tutte le entità citate ed eseguire il commit della transazione.

```
public static void updateCustomerOrder(EntityManager em) {
    em.getTransaction().begin();
    Order order = (Order) em.find(Order.class, "ORDER_1");
    processDiscount(order, 10);
    Customer cust = order.customer;
    cust.phoneNumber = "5075551234";
    em.getTransaction().commit();
}

public static void processDiscount(Order order, double discountPct) {
    for(OrderLine line : order.lines) {
        line.price = line.price * ((100-discountPct)/100);
    }
}
```

Il livellamento della transazione sincronizza tutte le entità gestite con la cache. Quando viene eseguito il commit di una transazione, si verifica automaticamente un livellamento. In tal caso, Order diventa un'entità gestita. Anche tutte le entità presenti come riferimento in Order, Customer, e OrderLine diventano entità gestite. Quando la transazione viene livellata, ognuna delle entità vengono controllate per determinare se sono state modificate. Quelle modificate vengono aggiornate nella cache. Quando la transazione ha terminato, tramite commit o roll back, le entità

vengono separate e tutte le modifiche che vengono effettuate nelle entità non vengono riflesse nella cache.

Supporto didattico gestore entità: aggiornamento e rimozione elementi con un indice

È possibile utilizzare un indice per trovare, aggiornare e rimuovere entità.

Procedura

Aggiornare e rimuovere entità utilizzando un indice. È possibile utilizzare un indice per trovare, aggiornare e rimuovere entità. Nei seguenti esempi, la classe di entità Order viene aggiornata per utilizzare l'annotazione @Index. L'annotazione @Index segnala a WebSphere eXtreme Scale di creare un indice di intervallo per un attributo. Il nome dell'indice è uguale al nome dell'attributo ed è sempre un tipo di indice MapRangeIndex.

Order.java

```
@Entity
public class Order
{
    @Id String orderNumber;
    @Index java.util.Date date;
    @OneToOne(cascade=CascadeType.PERSIST) Customer customer;
    @OneToMany(cascade=CascadeType.ALL, mappedBy="order")
    @OrderBy("lineNumber") List<OrderLine> lines;  }
```

Il seguente esempio mostra come annullare tutti gli ordini inoltrati entro l'ultimo minuto. Trovare l'ordine utilizzando un indice, aggiungere gli elementi in base all'ordine in cui si trovavano nell'inventario e rimuovere l'ordine e gli elementi di riga associati dal sistema.

```
public static void cancelOrdersUsingIndex(Session s)
  throws ObjectGridException {
    // Cancel all orders that were submitted 1 minute ago
    java.util.Date cancelTime = new
   java.util.Date(System.currentTimeMillis() - 60000);
    EntityManager em = s.getEntityManager();
    em.getTransaction().begin();
    MapRangeIndex dateIndex = (MapRangeIndex)
   s.getMap("Order").getIndex("date");
    Iterator<Tuple> orderKeys = dateIndex.findGreaterEqual(cancelTime);
  while(orderKeys.hasNext()) {
   Tuple orderKey = orderKeys.next();
   // Find the Order so we can remove it.
   Order curOrder = (Order) em.find(Order.class, orderKey);
   // Verify that the order was not updated by someone else.
   if(curOrder != null && curOrder.date.getTime() >= cancelTime.getTime()) {
    for(OrderLine line : curOrder.lines) {
     // Add the item back to the inventory.
     line.item.quantityOnHand += line.quantity;
     line.quantity = 0;
    }
   em.remove(curOrder);
   }
  }
 em.getTransaction().commit();
}
```

Supporto didattico gestore entità: aggiornamento e rimozione elementi tramite query

È possibile aggiornare e rimuovere entità utilizzando una query.

Procedura

Aggiornare e rimuovere entità utilizzando una query.

```
Order.java
@Entity
public class Order
{
    @Id String orderNumber;
    @Index java.util.Date date;
    @OneToOne(cascade=CascadeType.PERSIST) Customer customer;
    @OneToMany(cascade=CascadeType.ALL, mappedBy="order")
   @OrderBy("lineNumber") List<OrderLine> lines;
}
```

The order entity class is the same as it is in the previous example. The class still provides the @Index annotation, because the query string uses the date to find the entity. The query engine uses indices when they can be used.

```
public static void cancelOrdersUsingQuery(Session s) {
        // Cancel all orders that were submitted 1 minute ago
        java.util.Date cancelTime =
     new java.util.Date(System.currentTimeMillis() - 60000);
        EntityManager em = s.getEntityManager();
        em.getTransaction().begin();

        // Create a query that will find the order based on date.  Since
        // we have an index defined on the order date, the query
    // will automatically use it.
        Query query = em.createQuery("SELECT order FROM Order order
     WHERE order.date >= ?1");
        query.setParameter(1, cancelTime);
        Iterator<Order> orderIterator = query.getResultIterator();
    while(orderIterator.hasNext()) {
     Order order = orderIterator.next();
     // Verify that the order wasn't updated by someone else.
     // Since the query used an index, there was no lock on the row.
     if(order != null && order.date.getTime() >= cancelTime.getTime()) {
       for(OrderLine line : order.lines) {
        // Add the item back to the inventory.
        line.item.quantityOnHand += line.quantity;
        line.quantity = 0;
       }
       em.remove(order);
     }
   }
  em.getTransaction().commit();
}
```

COme nel precedente esempio, il metodo cancelOrdersUsingQuery intende annullare tutti gli ordini inoltrati nel precedente minuto. Per annullare l'ordine, trovare l'ordine utilizzando una query, aggiungendo gli elementi in base all'ordine in cui si trovavano nell'inventario e rimuovere l'ordine e gli elementi di riga associati dal sistema.

Supporto didattico ObjectQuery

Con i passi di seguito riportati, è possibile sviluppare un ObjectGrid locale in memoria che può memorizzare le informazioni sull'ordinamento per un sito Web; tali passi, inoltre, illustrano il modo in cui utilizzare ObjectQuery per eseguire query sui dati presenti nella griglia.

Prima di iniziare

Accertarsi che il file `objectgrid.jar` sia presente nel percorso di classe.

Informazioni su questa attività

Ciascun passo contenuto nel supporto didattico si sviluppa sul passo precedente. Seguire ciascuno dei passi per creare un'applicazione Java Platform, Standard Edition Versione 1.4 (o successive) che utilizza un ObjectGrid locale, in memoria.

Procedura

1. “Supporto didattico ObjectQuery - passo 1”
   - Come creare un ObjectGrid locale
   - Come definire uno schema per un singolo oggetto che utilizza l'accesso al campo
   - Come memorizzare l'oggetto
   - Come eseguire query sull'oggetto con ObjectQuery
2. “Supporto didattico ObjectQuery - passo 2” a pagina 196
   - Come creare un indice utilizzabile dalla query
3. “Supporto didattico ObjectQuery - passo 3” a pagina 197
   - Come creare uno schema con due entità correlate
   - Come memorizzare oggetti con un riferimento di chiave esterna tra di essi
   - Come eseguire query sugli oggetti utilizzando una query semplice con JOIN
4. “Supporto didattico ObjectQuery - passo 4” a pagina 199
   - Come creare uno schema con più entità correlate
   - Come utilizzare l'accesso al metodo o alla proprietà anziché l'accesso al campo

Supporto didattico ObjectQuery - passo 1

Con i passi di seguito riportati è possibile proseguire con lo sviluppo di un ObjectGrid locale in memoria che memorizza le informazioni sull'ordinamento per un archivio al dettaglio in linea utilizzando le API ObjectMap. Si definisce uno schema per la mappa e si esegue una query sulla mappa.

Procedura

1. Creare un ObjectGrid con uno schema di mappa.

   Creare un ObjectGrid con uno schema di mappa per la mappa, poi inserire un oggetto nella cache e successivamente richiamarlo utilizzando una query semplice.

   OrderBean.java

   ```
   public class OrderBean implements Serializable {
       String orderNumber;
       java.util.Date date;
       String customerName;
       String itemName;
       int quantity;
       double price;
   }
   ```

2. Definire la chiave primaria.

   Il codice precedentemente riportato mostra un oggetto OrderBean. Tale oggetto implementa l'interfaccia java.io.Serializable poiché tutti gli oggetti presenti nella cache devono essere (per impostazione predefinita) serializzabili.

   L'attributo orderNumber è la chiave primaria dell'oggetto. Il programma di esempio di seguito riportato può essere eseguito in modalità autonoma. Questo supporto didattico deve essere seguito in un progetto Eclipse Java in cui il file `objectgrid.jar` sia stato aggiunto al percorso di classe.

   Application.java

   ```
   package querytutorial.basic.step1;

   import java.util.Iterator;
   ```

```
import com.ibm.websphere.objectgrid.ObjectGrid;
import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
import com.ibm.websphere.objectgrid.ObjectMap;
import com.ibm.websphere.objectgrid.Session;
import com.ibm.websphere.objectgrid.config.QueryConfig;
import com.ibm.websphere.objectgrid.config.QueryMapping;
import com.ibm.websphere.objectgrid.query.ObjectQuery;

public class Application
{

    static public void main(String [] args) throws Exception
    {
        ObjectGrid og = ObjectGridManagerFactory.getObjectGridManager().createObjectGrid();
        og.defineMap("Order");

        // Define the schema
        QueryConfig queryCfg = new QueryConfig();
        queryCfg.addQueryMapping(new QueryMapping("Order", OrderBean.class.getName(),
     "orderNumber", QueryMapping.FIELD_ACCESS));
        og.setQueryConfig(queryCfg);

        Session s = og.getSession();
        ObjectMap orderMap = s.getMap("Order");

        s.begin();
        OrderBean o = new OrderBean();
        o.customerName = "John Smith";
        o.date = new java.util.Date(System.currentTimeMillis());
        o.itemName = "Widget";
        o.orderNumber = "1";
        o.price = 99.99;
        o.quantity = 1;
        orderMap.put(o.orderNumber, o);
        s.commit();

        s.begin();
        ObjectQuery query = s.createObjectQuery("SELECT o FROM Order o WHERE o.itemName='Widget'");
        Iterator result = query.getResultIterator();
        o = (OrderBean) result.next();
        System.out.println("Found order for customer: " + o.customerName);
        s.commit();
    }
}
```

Questa applicazione eXtreme Scale in primo luogo inizializza un ObjectGrid locale con un nome generato automaticamente. Dopodiché, l'applicazione crea un oggetto BackingMap e un oggetto QueryConfig che definisce il tipo Java associato alla mappa, il nome del campo che rappresenta la chiave primaria per la mappa e il modo in cui accedere ai dati presenti nell'oggetto. Si ottiene quindi una sessione per richiamare l'istanza ObjectMap e inserire un oggetto OrderBean nella mappa in una transazione.

Dopo aver eseguito il commit dei dati nella cache, è possibile utilizzare ObjectQuery per cercare l'oggetto OrderBean utilizzando qualunque campo persistente presente nella classe. I campi persistenti sono quelli che non hanno il modificatore transitorio. Poiché non sono stati definiti indici sull'oggetto BackingMap, ObjectQuery deve eseguire la scansione di ciascun oggetto nella mappa utilizzando la reflection di Java.

Operazioni successive

"Supporto didattico ObjectQuery - passo 2" illustra il modo in cui è possibile utilizzare un indice per ottimizzare la query.

Supporto didattico ObjectQuery - passo 2

Con i passi di seguito riportati, è possibile proseguire con la creazione di un ObjectGrid con una mappa e un indice, insieme ad uno schema per la mappa. Poi, è possibile inserire un oggetto nella cache e successivamente richiamarlo utilizzando una query semplice.

Prima di iniziare

Accertarsi di aver completato quanto indicato in "Supporto didattico ObjectQuery - passo 1" a pagina 195 prima di procedere con questo passo del supporto didattico.

Procedura

Schema e indice

Application.java

```
// Create an index
    HashIndex idx= new HashIndex();
    idx.setName("theItemName");
    idx.setAttributeName("itemName");
    idx.setRangeIndex(true);
    idx.setFieldAccessAttribute(true);
    orderBMap.addMapIndexPlugin(idx);
}
```

L'indice deve essere un'istanza com.ibm.websphere.objectgrid.plugins.index.HashIndex con le seguenti impostazioni:

- Il valore di Name è arbitrario, ma deve essere univoco per un dato oggetto BackingMap.
- AttributeName è il nome del campo o della proprietà di bean utilizzato dal motore di indicizzazione per eseguire l'introspezione della classe. In questo caso, esso è il nome del campo per il quale verrà creato un indice.
- RangeIndex deve essere sempre impostato su true.
- FieldAccessAttribute deve corrispondere al valore impostato nell'oggetto QueryMapping al momento della creazione dello schema della query. In questo caso, si accede all'oggetto Java utilizzando direttamente i campi.

Quando una query esegue tali filtri sul campo itemName, il motore di query utilizza automaticamente l'indice definito. L'utilizzo dell'indice consente l'esecuzione molto più veloce della query e non è necessaria la scansione di una mappa. Il passo successivo illustra il modo in cui è possibile utilizzare un indice per ottimizzare la query.
Passo successivo

Supporto didattico ObjectQuery - passo 3

Con i passi di seguito riportati, è possibile creare un ObjectGrid con due mappe e uno schema per le mappe con una relazione, poi inserire oggetti nella cache e successivamente richiamarli utilizzando una query semplice.

Prima di iniziare

Accertarsi di aver completato quanto indicato in “Supporto didattico ObjectQuery - passo 2” a pagina 196 prima di procedere con questo passo.

Informazioni su questa attività

In questo esempio esistono due mappe, ciascuna con singolo tipo Java associato ad essa. La mappa Order contiene oggetti OrderBean e la mappa Customer contiene oggetti CustomerBean.

Procedura

Definire le mappe con una relazione.

OrderBean.java

```
public class OrderBean implements Serializable {
    String orderNumber;
    java.util.Date date;
    String customerId;
    String itemName;
    int quantity;
    double price;
}
```

OrderBean non contiene più customerName. Esso contiene, invece, customerId, che è la chiave primaria per l'oggetto CustomerBean e per la mappa Customer.

CustomerBean.java

```
public class CustomerBean implements Serializable{
    private static final long serialVersionUID = 1L;
    String id;
    String firstName;
    String surname;
    String address;
    String phoneNumber;
}
```

Segue la relazione tra i due tipi o le due mappe:

Application.java

```
public class Application
{

    static public void main(String [] args)
        throws Exception
    {
        ObjectGrid og = ObjectGridManagerFactory.getObjectGridManager().createObjectGrid();
        og.defineMap("Order");
        og.defineMap("Customer");

        // Define the schema
        QueryConfig queryCfg = new QueryConfig();
        queryCfg.addQueryMapping(new QueryMapping(
            "Order", OrderBean.class.getName(), "orderNumber", QueryMapping.FIELD_ACCESS));
        queryCfg.addQueryMapping(new QueryMapping(
            "Customer", CustomerBean.class.getName(), "id", QueryMapping.FIELD_ACCESS));
        queryCfg.addQueryRelationship(new QueryRelationship(
             OrderBean.class.getName(), CustomerBean.class.getName(), "customerId", null));
        og.setQueryConfig(queryCfg);

        Session s = og.getSession();
        ObjectMap orderMap = s.getMap("Order");
        ObjectMap custMap = s.getMap("Customer");

        s.begin();
        CustomerBean cust = new CustomerBean();
        cust.address = "Main Street";
        cust.firstName = "John";
        cust.surname = "Smith";
        cust.id = "C001";
        cust.phoneNumber = "5555551212";
        custMap.insert(cust.id, cust);

        OrderBean o = new OrderBean();
        o.customerId = cust.id;
        o.date = new java.util.Date();
        o.itemName = "Widget";
        o.orderNumber = "1";
        o.price = 99.99;
        o.quantity = 1;
        orderMap.insert(o.orderNumber, o);
        s.commit();

        s.begin();
        ObjectQuery query = s.createObjectQuery(
            "SELECT c FROM Order o JOIN o.customerId as c WHERE o.itemName='Widget'");
        Iterator result = query.getResultIterator();
        cust = (CustomerBean) result.next();
        System.out.println("Found order for customer: " + cust.firstName + " " + cust.surname);
        s.commit();
    }
}
```

Segue l'equivalente XML nel descrittore di distribuzione ObjectGrid:

```
<?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="CompanyGrid">
      <backingMap name="Order"/>
      <backingMap name="Customer"/>

      <querySchema>
         <mapSchemas>
           <mapSchema
             mapName="Order"
             valueClass="com.mycompany.OrderBean"
             primaryKeyField="orderNumber"
             accessType="FIELD"/>
           <mapSchema
             mapName="Customer"
             valueClass="com.mycompany.CustomerBean"
             primaryKeyField="id"
             accessType="FIELD"/>
         </mapSchemas>
         <relationships>
           <relationship
             source="com.mycompany.OrderBean"
             target="com.mycompany.CustomerBean"
             relationField="customerId"/>
         </relationships>
      </querySchema>
    </objectGrid>
  </objectGrids>
</objectGridConfig>
```

Operazioni successive

"Supporto didattico ObjectQuery - passo 4" amplia il passo corrente includendo oggetti di accesso a campi e a proprietà e relazioni aggiuntive.

Supporto didattico ObjectQuery - passo 4

Il passo si seguito riportato illustra il modo in cui creare un ObjectGrid con quattro mappe ed uno schema per le mappe con più relazioni unidirezionali e bidirezionali. Poi, è possibile inserire oggetti nella cache e successivamente richiamarli utilizzando varie query.

Prima di iniziare

Accertarsi di aver completato quanto indicato in "Supporto didattico ObjectQuery - passo 3" a pagina 197 prima di proseguire con il passo corrente.

Procedura

Più relazioni di mappa

OrderBean.java

```
public class OrderBean implements Serializable {
    String orderNumber;
    java.util.Date date;
    String customerId;
```

```
    String itemName;
    int quantity;
    double price;
}
```

Come nel passo precedente, OrderBean non contiene più customerName. Esso contiene, invece, customerId, che è la chiave primaria per l'oggetto CustomerBean e per la mappa Customer.

CustomerBean.java

```
public class CustomerBean implements Serializable{
    private static final long serialVersionUID = 1L;
    String id;
    String firstName;
    String surname;
    String address;
    String phoneNumber;
}
```

Avendo creato le classi sopra riportate, è possibile eseguire l'applicazione di seguito indicata.

Application.java

```
public class Application
{

    static public void main(String [] args)
        throws Exception
    {
        ObjectGrid og = ObjectGridManagerFactory.getObjectGridManager().createObjectGrid();
        og.defineMap("Order");
        og.defineMap("Customer");

        // Define the schema
        QueryConfig queryCfg = new QueryConfig();
        queryCfg.addQueryMapping(new QueryMapping(
            "Order", OrderBean.class.getName(), "orderNumber", QueryMapping.FIELD_ACCESS));
        queryCfg.addQueryMapping(new QueryMapping(
            "Customer", CustomerBean.class.getName(), "id", QueryMapping.FIELD_ACCESS));
        queryCfg.addQueryRelationship(new QueryRelationship(
             OrderBean.class.getName(), CustomerBean.class.getName(), "customerId", null));
        og.setQueryConfig(queryCfg);

        Session s = og.getSession();
        ObjectMap orderMap = s.getMap("Order");
        ObjectMap custMap = s.getMap("Customer");

        s.begin();
        CustomerBean cust = new CustomerBean();
        cust.address = "Main Street";
        cust.firstName = "John";
        cust.surname = "Smith";
        cust.id = "C001";
        cust.phoneNumber = "5555551212";
        custMap.insert(cust.id, cust);

        OrderBean o = new OrderBean();
        o.customerId = cust.id;
        o.date = new java.util.Date();
        o.itemName = "Widget";
        o.orderNumber = "1";
        o.price = 99.99;
        o.quantity = 1;
        orderMap.insert(o.orderNumber, o);
        s.commit();

        s.begin();
        ObjectQuery query = s.createObjectQuery(
            "SELECT c FROM Order o JOIN o.customerId as c WHERE o.itemName='Widget'");
        Iterator result = query.getResultIterator();
        cust = (CustomerBean) result.next();
        System.out.println("Found order for customer: " + cust.firstName + " " + cust.surname);
        s.commit();
    }
}
```

L'utilizzo della configurazione XML di seguito indicata (nel descrittore di distribuzione ObjectGrid) equivale all'approccio porgrammatico sopra riportato.

```
<?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="og1">
      <backingMap name="Order"/>
      <backingMap name="Customer"/>

      <querySchema>
         <mapSchemas>
           <mapSchema
             mapName="Order"
             valueClass="com.mycompany.OrderBean"
             primaryKeyField="orderNumber"
             accessType="FIELD"/>
           <mapSchema
             mapName="Customer"
             valueClass="com.mycompany.CustomerBean"
             primaryKeyField="id"
             accessType="FIELD"/>
         </mapSchemas>
         <relationships>
           <relationship
             source="com.mycompany.OrderBean"
             target="com.mycompany.CustomerBean"
             relationField="customerId"/>
         </relationships>
      </querySchema>
    </objectGrid>
  </objectGrids>
</objectGridConfig>
```

Supporto didattico sulla sicurezza Java SE: panoramica

Con il seguente supporto didattico, è possibile creare un ambiente distribuito eXtreme Scale in un ambiente Java Platform, Standard Edition.

Prima di iniziare

È necessario avere familiarità con i concetti di base di una configurazione eXtreme Scale distribuita.

Informazioni su questa attività

In questo supporto didattico, il server di catalogo, il server contenitore e il client vengono eseguiti tutti in un ambiente Java SE. Ogni passo nel supporto didattico si basa su quello precedente. Seguire ogni passo per proteggere un eXtreme Scale distribuito e sviluppare una semplice applicazione Java SE e accedere a eXtreme Scale sicura.

Inizio supporto didattico

Procedura

1. "Supporto didattico sulla sicurezza - Passo 1 Java SE" a pagina 202
   - Avviare un server di catalogo non sicuro
   - Avviare un server contenitore non sicuro
   - Avviare un client per accedere ai dati

- Utilizzare xsadmin per visualizzare la dimensione della mappa
- Arrestare il server

2. "Java SE supporto didattico sulla sicurezza - Passo 2" a pagina 205
   - Utilizzo di CredentialGenerator
   - Utilizzo di Authenticator
   - Avviare un server di catalogo sicuro
   - Avviare un server contenitore sicuro
   - Avviare un client per accedere a ObjectGrid sicuro
   - Utilizzare xsadmin per visualizzare la dimensione della mappa
   - Arrestare il server sicuro
3. "Java SE Supporto didattico - Passo 3" a pagina 212
   - Utilizzare la politica di autorizzazione JAAS
4. "Java SE supporto didattico - Passo 4" a pagina 215
   - Creare un keystore e un truststore
   - Configurare le proprietà SSL per il server
   - Configurare le proprietà SSL per il client
   - Utilizzare xsadmin per visualizzare la dimensione della mappa
   - Arrestare il server sicuro

Supporto didattico sulla sicurezza - Passo 1 Java SE

Questa sezione descrive un *semplice esempio libero*. Funzioni aggiuntive sulla sicurezza vengono aggiunte progressivamente nei passi del supporto didattico per aumentare la quantità di sicurezza integrata disponibile.

Prima di iniziare

Nota: Tutti i file richiesti per questo passo del supporto didattico vengono forniti nella seguente sezione.

Procedura

Eseguire l'esempio
Avviare il servizio catalogo utilizzando i seguenti script. Per ulteriori informazioni sull'avvio del servizio catalogo, consultare per informazioni sull'avvio di un servizio catalogo in *Guida alla gestione*.

1. Navigare nella directory bin: `cd objectgridRoot/bin`
2. Avviare un server di catalogo denominato catalogServer:
   - UNIX Linux `startOgServer.sh catalogServer`
   - Windows `startOgServer.bat catalogServer`
3. Navigare nella directory bin `cd objectgridRoot/bin`
4. Quindi lanciare un server contenitore denominato c0 con il seguente script:
   - UNIX Linux `startOgServer.sh c0 -objectGridFile ../xml/SimpleApp.xml -deploymentPolicyFile ../xml/SimpleDP.xml -catalogServiceEndpoints localhost:2809`
   - Windows `startOgServer.bat c0 -objectGridFile ../xml/SimpleApp.xml -deploymentPolicyFile ../xml/SimpleDP.xml -catalogServiceEndpoints localhost:2809`

Esempio

Per ulteriori informazioni sull'avvio dei server contenitori, consultare per informazioni sull'avvio dei processi contenitori in *Guida alla gestione*.

Dopo aver avviato il server di catalogo e il server contenitore, lanciare il client come segue.

1. Navigate nella directory bin ancora una volta.
2. `java -classpath ../lib/objectgrid.jar;../applib/secsample.jar com.ibm.websphere.objectgrid.security.sample.guide.SimpleApp`

Il file `secsample.jar` contiene la classe SimpleApp.

L'output di questo programma è:

```
Il nome cliente per ID 0001 è fName lName
```

È possibile utilizzare inoltre xsadmin per mostrare le opzioni mapsizes della griglia "accounting".

- Navigare nella directory `objectgridRoot/bin`.
- Utilizzare il comando `xsadmin` con l'opzione -mapSizes nel modo che segue.
  - UNIX Linux `xsadmin.sh -g accounting -m mapSet1 -mapSizes`
  - Windows `xsadmin.bat -g accounting -m mapSet1 -mapSizes`

  Verrà visualizzato il seguente output.

  ```
  This administrative utility is provided as a sample only and is not to be
  considered a fully supported component of the WebSphere eXtreme Scale
  product.
  Connecting to Catalog service at localhost:1099
  *********** Displaying Results for Grid - accounting, MapSet - mapSet1
  ***********
  *** Listing Maps for c0 ***
  Map Name: customer Partition #: 0 Map Size: 1 Shard Type: Primary
  Server Total: 1
  Total Domain Count: 1
  ```

Arresto dei server

Server contenitori

Utilizzare i seguenti comandi per arrestare il server contenitore c0.

UNIX Linux `stopOgServer.sh c0 -catalogServiceEndPoints localhost:2809`

Windows `stopOgServer.bat c0 -catalogServiceEndPoints localhost:2809`

Verrà visualizzato il seguente messaggio.

```
CWOBJ2512I: Server ObjectGrid c0 arrestato.
```

Server di catalogo

È possibile arrestare il server di catalogo utilizzando il seguente comando.

UNIX Linux
```
stopOgServer.sh catalogServer -catalogServiceEndPoints
localhost:2809
```

Windows
```
stopOgServer.bat catalogServer -catalogServiceEndPoints
localhost:2809
```

Se si chiude il server di catalogo, verrà visualizzato il seguente messaggio.

```
CWOBJ2512I: Server ObjectGrid catalogServer arrestato.
```

File richiesti

Il seguente file è la classe Java per SimpleApp.

SimpleApp.java

```
// 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.
// Licensed Materials - Property of IBM
// 5724-J34 (C) COPYRIGHT International Business Machines Corp. 2007-2009
package com.ibm.websphere.objectgrid.security.sample.guide;

import com.ibm.websphere.objectgrid.ClientClusterContext;
import com.ibm.websphere.objectgrid.ObjectGrid;
import com.ibm.websphere.objectgrid.ObjectGridManager;
import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
import com.ibm.websphere.objectgrid.ObjectMap;
import com.ibm.websphere.objectgrid.Session;

public class SimpleApp {

    public static void main(String[] args) throws Exception {

        SimpleApp app = new SimpleApp();
        app.run(args);
    }

    /**
     * read and write the map
     * @throws Exception
     */
    protected void run(String[] args) throws Exception {
        ObjectGrid og = getObjectGrid(args);

        Session session = og.getSession();

        ObjectMap customerMap = session.getMap("customer");

        String customer = (String) customerMap.get("0001");

        if (customer == null) {
            customerMap.insert("0001", "fName lName");
        } else {
            customerMap.update("0001", "fName lName");
        }
        customer = (String) customerMap.get("0001");

        System.out.println("The customer name for ID 0001 is " + customer);
    }

    /**
     * Get the ObjectGrid
     * @return an ObjectGrid instance
     * @throws Exception
     */
    protected ObjectGrid getObjectGrid(String[] args) throws Exception {
        ObjectGridManager ogManager = ObjectGridManagerFactory.getObjectGridManager();
```

```
        // Create an ObjectGrid
        ClientClusterContext ccContext = ogManager.connect("localhost:2809", null, null);
        ObjectGrid og = ogManager.getObjectGrid(ccContext, "accounting");

        return og;

    }

}
```

Il metodo getObjectGrid in questa classe ottiene un ObjectGrid e il metodo run legge un record dalla mappa customer e aggiorna il valore.

Per eseguire questo esempio in un ambiente distribuito, un file XML descrittore ObjectGrid `SimpleApp.xml` e un file XML di distribuzione `SimpleDP.xml`, vengono creati. Questi file vengono evidenziati nel seguente esempio:

SimpleApp.xml

```
<?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="accounting">
            <backingMap name="customer" readOnly="false" copyKey="true"/>
        </objectGrid>
    </objectGrids>
</objectGridConfig>
```

Il seguente file XML configura l'ambiente di distribuzione.

SimpleDP.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
 xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">

 <objectgridDeployment objectgridName="accounting">
  <mapSet name="mapSet1" numberOfPartitions="1" minSyncReplicas="0" maxSyncReplicas="2" maxAsyncReplicas="1">
   <map ref="customer"/>
  </mapSet>
 </objectgridDeployment>
</deploymentPolicy>
```

Questa è una semplice configurazione di ObjectGrid con un'istanza ObjectGrid denominata "accounting" e una mappa denominata "customer" (all'interno di mapSet "mapSet1"). Il file `SimpleDP.xml` rappresenta una serie di mappe configurata con una partizione 1 e il numero minimo richiesto di repliche 0.

Passo successivo del supporto didattico

Java SE supporto didattico sulla sicurezza - Passo 2

Sulla base del passo precedente, il seguente argomento mostra in che modo implementare l'autenticazione del client in un ambiente eXtreme Scale distribuito.

Prima di iniziare

Assicurarsi di aver completato “Supporto didattico sulla sicurezza - Passo 1 Java SE” a pagina 202.

Informazioni su questa attività

Con l'autenticazione client abilitata, un client viene autenticato prima della connessione al server eXtreme Scale. Questa sezione dimostra in che modo

l'autenticazione client può essere effettuata in un ambiente server eXtreme Scale, incluso codice di esempio e script per dimostrazione.

Come qualsiasi altro meccanismo di autenticazione, l'autenticazione minima consiste dei seguenti passi:

1. L'amministratore modifica le configurazioni per rendere l'autenticazione un requisito.
2. Il client fornisce le credenziali al server.
3. Il server autentica le credenziali nel registro.

Procedura

1. **Credenziali client**

   Le credenziali di un client sono rappresentate da un'interfaccia com.ibm.websphere.objectgrid.security.plugins.Credential. Le credenziali di un client possono essere una coppia di nome utente e password, un ticket Kerberos, un certificato client o dati in qualsiasi formato su cui concordano il client e il server. Per ulteriori dettagli, fare riferimento alla documentazione API delle credenziali.

   Questa interfaccia definisce esplicitamente i metodi equals(Object) e hashCode(). Questi due metodi sono importanti in quanto gli oggetti Subject autenticati vengono memorizzati nella cache utilizzando l'oggetto Credential come la chiave lato server.

   eXtreme Scale fornisce inoltre un plug-in per generare una credenziale. Questo plug-in viene rappresentato dall'interfaccia com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator e viene utilizzato per generare le credenziali del client. È utile quando le credenziali sono soggette a scadenza. In questo caso, il metodo getCredential() viene chiamato per rinnovare le credenziali. Fare riferimento alla documentazione relativa a CredentialGenerator API per ulteriori dettagli.

   È possibile implementare queste due interfacce per eXtreme Scale client runtime per ottenere le credenziali del client.

   Di seguito vengono riportati due esempi di implementazioni di plug-in forniti da eXtreme Scale.

   ```
   com.ibm.websphere.objectgrid.security.plugins.builtins.UserPasswordCredential
   com.ibm.websphere.objectgrid.security.plugins.builtins.UserPasswordCredentialGenerator
   ```

   Per ulteriori informazioni su questi plug-in, consultare la sezione sulla programmazione dell'autenticazione client nella *Guida alla programmazione*.

2. **Programma di autenticazione server** Dopo che il client eXtreme Scale richiama l'oggetto Credential tramite l'oggetto CredentialGenerator, tale oggetto Credential del client viene inviato insieme alla richiesta client al server eXtreme Scale. Il server eXtreme Scale autentica l'oggetto Credential prima di elaborare la richiesta. Se l'oggetto Credential viene autenticato correttamente, un oggetto Subject viene restituito per rappresentare questo client.

   Questo oggetto Subject viene quindi memorizzato nella cache e scade dopo che il suo ciclo di vita raggiunge il valore di timeout della sessione. Il valore di timeout della sessione di login può essere impostato utilizzando la proprietà loginSessionExpirationTime nel file XML del cluster. Ad esempio, l'impostazione di `loginSessionExpirationTime="300"` fa scadere l'oggetto Subject in 300 secondi.L'oggetto Subject viene quindi utilizzato per autorizzare la richiesta, che viene mostrato successivamente.

   Un server eXtreme Scale utilizza il plug-in Authenticator per autenticare l'oggetto Credential. Fare riferimento alla documentazione API relativa a Authenticator per ulteriori dettagli.

Questo esempio utilizza un'implementazione eXtreme Scale incorporata: KeyStoreLoginAuthenticator a scopo di esempio e test (un keystore è un semplice registro utente e non dovrebbe essere utilizzato per la produzione). nella *Guida alla programmazione*.

KeyStoreLoginAuthenticator utilizza un KeyStoreLoginModule per autenticare l'utente con il keystore tramite il modulo di login JAAS "KeyStoreLogin". Il keystore può essere configurato come un'opzione nella classe KeyStoreLoginModule. Il seguente esempio illustra l'alias keyStoreLogin configurato nel file di configurazione JAAS og_jaas.config:

```
KeyStoreLogin{
com.ibm.websphere.objectgrid.security.plugins.builtins.KeyStoreLoginModule required
     keyStoreFile="../security/sampleKS.jks" debug = true;
};
```

I seguenti comandi creano un keystore sampleKS.jks nella directory %OBJECTGRID_HOME%/security con la password come sampleKS1. Inoltre, tre certificati utente che rappresentano l'utente amministratore, l'utente gestore e l'utente cassiere vengono creati con relative password.

a. Navigare nella directory root di eXtreme Scale.

   ```
   cd objectgridRoot
   ```

b. Creare una directory denominata "security".

   ```
   mkdir security
   ```

c. Navigare nella directory security appena creata.

   ```
   cd security
   ```

d. Utilizzare keytool (nella directory javaHOME/bin) per creare un utente "administator" con password "administrator1" nel keystore sampleKS.jks.

   ```
   keytool -genkey -v -keystore ./sampleKS.jks -storepass sampleKS1
   -alias administrator -keypass administrator1
   -dname CN=administrator,O=acme,OU=OGSample -validity 10000
   ```

e. Utilizzare keytool (nella directory javaHOME/bin) per creare un utente "manager" con password "manager1" nel keystore sampleKS.jks.

   ```
   keytool -genkey -v -keystore ./sampleKS.jks -storepass sampleKS1
   -alias manager -keypass manager1
   -dname CN=manager,O=acme,OU=OGSample -validity 10000
   ```

f. Utilizzare keytool (nella directory javaHOME/bin) per creare un utente "cashier" con password "cashier1" nel keystore sampleKS.jks.

   ```
   keytool -genkey -v -keystore ./sampleKS.jks -storepass sampleKS1
   -alias cashier -keypass cashier1 -dname CN=cashier,O=acme,OU=OGSample
   -validity 10000
   ```

La configurazione della sicurezza client viene effettuata nel file delle proprietà del client. Utilizzare il seguente comando per creare una copia nella directory %OBJECTGRID_HOME%/security:

a. Andare nella directory security.

   ```
   cd objectgridRoot/security
   ```

b. Copiare il file sampleClient.properties nel file client.properties file.

   ```
   cp ../properties/sampleClient.properties client.properties
   ```

Le seguenti proprietà vengono evidenziate nel file client.properties nella directory security.

a. **securityEnabled:** impostando securityEnabled su true (valore predefinito) si abilita la sicurezza del client, che include l'autenticazione.

b. **credentialAuthentication:** impostare credentialAuthentication su Supported (valore predefinito), che significa che il client supporta l'autenticazione credenziali.

c. **transportType:** impostare transportType su TCP/IP, che significa che nessun SSL verrà utilizzato.

d. **singleSignOnEnabled:** impostarlo su false (valore predefinito). L'accesso singolo non è consentito.

3. **Configurazione della sicurezza server**

   La configurazione della sicurezza server viene specificata nel file XML descrittore della sicurezza e nel file delle proprietà della sicurezza server.Il file XML descrittore della sicurezza descrive le proprietà della sicurezza comune a tutti i server (inclusi i server di catalogo e i server contenitori). Un esempio di proprietà è la configurazione del programma di autenticazione che rappresenta il registro utente e il meccanismo di autenticazione.

   Di seguito il file `security.xml` da utilizzare in questo esempio:

```
<?xml version="1.0" encoding="UTF-8"?>
<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" loginSessionExpirationTime="300" >

        <authenticator className ="com.ibm.websphere.objectgrid.security.plugins.builtins.
     KeyStoreLoginAuthenticator">
        </authenticator>
    </security>

</securityConfig>
```

   a. **securityEnabled:** impostare su true, per abilitare la sicurezza del server, inclusa l'autenticazione.

   b. **loginSessionExpirationTime:** impostare il valore su 300 (valore predefinito).

   c. **authenticator:** aggiungere la classe authenticator KeyStoreLoginAuthenticator al file XML del cluster come di seguito indicato:

```
<authenticator className ="com.ibm.websphere.objectgrid.security.plugins.builtins.
     KeyStoreLoginAuthenticator">
                </authenticator>
```

   d. **credentialAuthentication:** impostare l'attributo credentialAuthentication su Required in modo che il server richieda l'autenticazione

   Per spiegazioni aggiuntive più dettagliate sul file `security.xml`, consultare le informazioni sul file XML descrittore della sicurezza in the information about the security descriptor XML file in *Guida alla gestione*.

   Copiare il file delle proprietà del server nella directory security. In questo momento, non è necessario apportare alcuna modifica a questo file.

   a. Navigare nella directory security.

      ```
      cd objectgridRoot/security
      ```

   b. Copiare il file objectGrid `sampleServer.properties` di esempio dalla directory delle proprietà nel nuovo file `server.properties`.

      ```
      cp ../properties/containerServer.properties server.properties
      ```

   Apportare le seguenti modifiche nel file `server.properties`:

   a. **securityEnabled:** impostare l'attributo **securityEnabled** su true.

   b. **transportType:** impostare l'attributo **transportType** su TCP/IP, che significa che nessun SSL viene utilizzato.

   c. **secureTokenManagerType:** impostare l'attributo **secureTokenManagerType** su none per non configurare il gestore del token secure.

4. **Client sicuro** Connettere l'applicazione client al server in modo sicuro come mostrato nel seguente esempio:

```
// 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.
// Licensed Materials - Property of IBM
// 5724-J34 (C) COPYRIGHT International Business Machines Corp. 2007-2009
package com.ibm.websphere.objectgrid.security.sample.guide;

import com.ibm.websphere.objectgrid.ClientClusterContext;
import com.ibm.websphere.objectgrid.ObjectGrid;
import com.ibm.websphere.objectgrid.ObjectGridManager;
import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
import com.ibm.websphere.objectgrid.security.config.ClientSecurityConfiguration;
import com.ibm.websphere.objectgrid.security.config.ClientSecurityConfigurationFactory;
import com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator;
import com.ibm.websphere.objectgrid.security.plugins.builtins.UserPasswordCredentialGenerator;

public class SecureSimpleApp extends SimpleApp {

    public static void main(String[] args) throws Exception {

        SecureSimpleApp app = new SecureSimpleApp();
        app.run(args);
    }

    /**
     * Get the ObjectGrid
     * @return an ObjectGrid instance
     * @throws Exception
     */
    protected ObjectGrid getObjectGrid(String[] args) throws Exception {
        ObjectGridManager ogManager = ObjectGridManagerFactory.getObjectGridManager();
        ogManager.setTraceFileName("logs/client.log");
        ogManager.setTraceSpecification("ObjectGrid*=all=enabled:ORBRas=all=enabled");

        // Creates a ClientSecurityConfiguration object using the specified file
        ClientSecurityConfiguration clientSC = ClientSecurityConfigurationFactory
                .getClientSecurityConfiguration(args[0]);

        // Creates a CredentialGenerator using the passed-in user and password.
        CredentialGenerator credGen = new UserPasswordCredentialGenerator(args[1], args[2]);
        clientSC.setCredentialGenerator(credGen);

        // Create an ObjectGrid by connecting to the catalog server
        ClientClusterContext ccContext = ogManager.connect("localhost:2809", clientSC, null);
        ObjectGrid og = ogManager.getObjectGrid(ccContext, "accounting");

        return og;

    }

}
```

Ci sono tre differenti aspetti rispetto all'applicazione non sicura:

a. È stato creato un oggetto ClientSecurityConfiguration aggirando il file `client.properties` configurato.
b. È stato creato UserPasswordCredentialGenerator utilizzando ID utente e password approvati.
c. Connesso al server di catalogo per ottenere un ObjectGrid da ClientClusterContext aggirando un oggetto ClientSecurityConfiguration.

5. **Eseguire l'applicazione**

   Per eseguire l'applicazione, avviare il server di catalogo. Immettere le opzioni della riga comandi -clusterFile e -serverProps per far approvare le proprietà relative alla sicurezza:

   a. Navigare nella directory bin:

      ```
      cd objectgridRoot/bin
      ```

   b. Lanciare il server di catalogo:

      - UNIX Linux

        ```
        startOgServer.sh catalogServer -clusterSecurityFile ../security/security.xml
        -serverProps ../security/server.properties -jvmArgs
        -Djava.security.auth.login.config="../security/og_jaas.config"
        ```

- Windows

```
startOgServer.bat catalogServer -clusterSecurityFile ../security/security.xml
-serverProps ../security/server.properties -jvmArgs
-Djava.security.auth.login.config="../security/og_jaas.config"
```

Successivamente, lanciare un server contenitore sicuro utilizzando il seguente script:

a. Navigare di nuovo nella directory bin:

```
cd objectgridRoot/bin
```

b. Lanciare un server contenitore sicuro:

- Linux UNIX

```
startOgServer.sh c0 -objectgridFile ../xml/SimpleApp.xml
-deploymentPolicyFile ../xml/SimpleDP.xml
-catalogServiceEndpoints localhost:2809
-serverProps ../security/server.properties
-jvmArgs -Djava.security.auth.login.config="../security/og_jaas.config"
```

- Windows

```
startOgServer.bat c0 -objectgridFile ../xml/SimpleApp.xml
-deploymentPolicyFile ../xml/SimpleDP.xml
-catalogServiceEndpoints localhost:2809
-serverProps ../security/server.properties
-jvmArgs -Djava.security.auth.login.config="../security/og_jaas.config"
```

Il file delle proprietà del server viene aggirato immettendo -serverProps.

Dopo che il server è stato avviato, lanciare il client utilizzando il seguente comando:

a. `cd objectgridRoot/bin`

b.

```
java -classpath ../lib/objectgrid.jar;../applib/secsample.jar
  com.ibm.websphere.objectgrid.security.sample.guide.SecureSimpleApp
  ../security/client.properties manager manager1
```

Il file `secsample.jar` contiene la classe SimpleApp.

SecureSimpleApp utilizza tre parametri che vengono forniti nel seguente elenco:

a. Il file `../security/client.properties` è il file delle proprietà per la sicurezza del client.

b. `manager` is the user ID.

c. `manager1` is the password.

Dopo aver immesso la classe, si ottiene il seguente output:

Il nome cliente per `ID 0001 è fName lName`.

È possibile utilizzare inoltre xsadmin per mostrare le opzioni mapsizes della griglia "accounting".

- Navigare nella directory `objectgridRoot/bin`.
- Utilizzare il comando `xsadmin` con l'opzione -mapSizes nel modo che segue.
  - UNIX Linux `xsadmin.sh -g accounting -m mapSet1 -username manager -password manager1 -mapSizes`
  - Windows `xsadmin.bat -g accounting -m mapSet1 -username manager -password manager1 -mapSizes`

  Viene visualizzato il seguente output.

```
This administrative utility is provided as a sample only and is not to
be considered a fully supported component of the WebSphere eXtreme
Scale product.
Connecting to Catalog service at localhost:1099
```

```
********** Displaying Results for Grid - accounting, MapSet - mapSet1
**********
*** Listing Maps for c0 ***
Map Name: customer Partition #: 0 Map Size: 1 Shard Type: Primary
Server Total: 1
Total Domain Count: 1
```

Ora è possibile utilizzare il comando stopOgServer per arrestare il server contenitore o il processo del servizio catalogo. Tuttavia, è necessario fornire un file di configurazione della sicurezza. Il file di esempio delle proprietà del client definisce le due seguenti proprietà per generare le credenziali IDutente/password (manager/manager1).

```
credentialGeneratorClass=com.ibm.websphere.objectgrid.security.plugins.builtins.UserPasswordCredentialGenerator
credentialGeneratorProps=manager manager1
```

Arrestare il contenitore c0 con il seguente comando.

- UNIX Linux `stopOgServer.sh c0 -catalogServiceEndPoints localhost:2809 -clientSecurityFile ..\security\client.properties`
- Windows `stopOgServer.bat c0 -catalogServiceEndPoints localhost:2809 -clientSecurityFile ..\security\client.properties`

Se non si fornisce l'opzione -clientSecurityFile, verrà visualizzata un'eccezione con il seguente messaggio.

```
>> SERVER (id=39132c79, host=9.10.86.47) TRACE START:
>> org.omg.CORBA.NO_PERMISSION: Server requires credential
authentication but there is no security context from the client. This
usually happens when the client does not pass a credential the server.
vmcid: 0x0
minor code: 0
completed: No
```

È possibile chiudere il server di catalogo utilizzando il seguente comando. Tuttavia, se si desidera continuare a provare con il passo successivo del supporto didattico, è possibile lasciare il server di catalogo in esecuzione.

- UNIX Linux `stopOgServer.sh catalogServer -catalogServiceEndPoints localhost:2809 -clientSecurityFile ..\security\client.properties`
- Windows `stopOgServer.bat catalogServer -catalogServiceEndPoints localhost:2809 -clientSecurityFile ..\security\client.properties`

Se si chiude il server di catalogo, verrà visualizzato il seguente output.

```
CWOBJ2512I: ObjectGrid server catalogServer stopped
```

Ora, il sistema è stato correttamente protetto in modo parziale, abilitando l'autenticazione. Il server è stato configurato per il plug in del registro utente, il client è stato configurato per fornire le credenziali del client ed è stato modificato il file delle proprietà del client e il file XML del cluster per abilitare l'autenticazione.

Se si fornisce una password non valida, viene visualizzata un'eccezione che dichiara che il nome utente o la password non sono corretti.

Per ulteriori dettagli sull'autenticazione del client, consultare le informazioni sull'autenticazione client dell'applicazione in *Guida alla gestione*.

Passo successivo del suppporto didattico

Java SE Supporto didattico - Passo 3

Dopo aver autenticato un client, come nel passo precedente, è possibile fornire i privilegi sulla sicurezza tramite i meccanismi di autorizzazione eXtreme Scale.

Prima di iniziare

Assicurarsi di aver completato “Java SE supporto didattico sulla sicurezza - Passo 2” a pagina 205 prima di procedere con quest'attività.

Informazioni su questa attività

Il passo precedente di questo supporto didattico mostra come abilitare l'autenticazione in una griglia eXtreme Scale. Di conseguenza, nessun client non autenticato può connettersi al proprio server e sottoporre richieste al proprio sistema. Tuttavia, ogni client autenticato ha la stessa autorizzazione o privilegi sul server, come lettura, scrittura o cancellazione dati memorizzati nelle mappe ObjectGrid. I client possono inoltre immettere qualsiasi tipo di query. Questa sezione mostra come utilizzare l'autorizzazione eXtreme Scale per dare ai vari utenti autenticati i diversi privilegi.

Analogamente a molti altri sistemi, eXtreme Scale adotta un meccanismo che si basa sulle autorizzazioni. WebSphere eXtreme Scale ha differenti categorie di autorizzazioni rappresentate da differenti classi di autorizzazioni. Questa sezione consiglia MapPermission. Per una completa categoria di autorizzazioni, consultare il the client authorization reference in the *Guida alla programmazione*..

In WebSphere eXtreme Scale, la classe com.ibm.websphere.objectgrid.security.MapPermission rappresenta le autorizzazioni alle risorse eXtreme Scale, specificamente i metodi delle interfacce ObjectMap o JavaMap. WebSphere eXtreme Scale definisce le seguenti stringhe di autorizzazione per accedere ai metodi di ObjectMap e JavaMap:

- read: garantisce l'autorizzazione a leggere i dati dalla mappa.
- write: garantisce l'autorizzazione ad aggiornare i dati nella mappa.
- insert: garantisce l'autorizzazione ad inserire i dati nella mappa.
- remove: garantisce l'autorizzazione ad eliminare i dati dalla mappa.
- invalidate: garantisce l'autorizzazione a invalidare i dati dalla mappa.
- all: garantisce tutte le autorizzazioni a leggere, scrivere, inserire rimuovere e invalidare.

L'autorizzazione si verifica quando un client chiama un metodo di ObjectMap o JavaMap. Il runtime di eXtreme Scale controlla le diverse autorizzazioni della mappa per i diversi metodi. Se le autorizzazioni richieste non sono garantite nel client, il risultato è AccessControlException.

Questo supporto didattico mostra come utilizzare l'autorizzazione JAAS (Authentication and Authorization Service) Java per garantire gli accessi alla mappa delle autorizzazioni per i diversi utenti.

Procedura

1. **Abilitare l'autorizzazione eXtreme Scale**. Per abilitare l'autorizzazione su ObjectGrid, è necessario impostare l'attributo securityEnabled su `true` per quel particolare ObjectGrid nel file XML. Abilitare la sicurezza su ObjectGrid significa che si sta abilitando l'autorizzazione. Utilizzare i seguenti comandi per creare un nuovo file XML ObjectGrid con la sicurezza abilitata.

a. Navigare nella directory bin.

```
cd objectgridRoot/bin
```

b. Copiare il file SimpleApp.xml nel file SecureSimpleApp.xml.

```
cp SimpleApp.xml SecureSimpleApp.xml
```

c. Aprire il file SecureSimpleApp.xml e aggiungere securityEnabled="true" a livello ObjectGrid come mostra il seguente XML:

```
<?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="accounting" securityEnabled="true">
            <backingMap name="customer" readOnly="false" copyKey="true"/>
        </objectGrid>
    </objectGrids>
</objectGridConfig>
```

2. **Definire la politica di autorizzazione.** Nella sezione di autenticazione pre-client, sono stati creati tre utenti nel keystore: cashier, manager e administrator. In questo esempio, l'utente "cashier" ha solo autorizzazioni in lettura a tutte le mappe e l'utente "manager" ha tutte le autorizzazioni. In questo esempio viene utilizzata l'autorizzazione JAAS. L'autorizzazione JAAS utilizza il file della politica di autorizzazione per garantire le autorizzazioni ai principal. Il seguente file viene definito nella directory security:

```
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";
};
```

Nota:

- codebase "http://www.ibm.com/com/ibm/ws/objectgrid/security/PrivilegedAction" è un URL appositamente riservato per ObjectGrid. Tutte le autorizzazioni ObjectGrid garantite ai principal devono utilizzare questa base di codice speciale.
- La prima istruzione di garanzia concede autorizzazione alla mappa in "read" nel principal "CN=cashier,O=acme,OU=OGSample", in modo che l'utente "cashier" abbia solo l'autorizzazione in "read" a tutte le mappe nell'accounting ObjectGrid.
- La seconda istruzione di garanzia concede l'autorizzazione "all" alla mappa nel principal "CN=manager,O=acme,OU=OGSample", in modo che l'utente "manager" abbia tutte le autorizzazioni alle mappe nell'accounting ObjectGrid.

Ora è possibile lanciare un server con una politica di autorizzazione. Il file della politica di autorizzazione JAAS può essere impostato utilizzando lo standard -D property: -Djava.security.auth.policy=../security/ogAuth.policy

3. **Eseguire applicazione.**

Dopo aver creato i file di cui sopra, è possibile eseguire l'applicazione.

Utilizzare i seguenti comandi per avviare il server di catalogo. Per ulteriori informazioni sull'avvio del servizio catalogo, consultare per informazioni sull'avvio di un servizio catalogo in *Guida alla gestione*.

a. Navigare nella directory bin: cd objectgridRoot/bin

b. Avviare il server di catalogo.

- UNIX Linux startOgServer.sh catalogServer -clusterSecurityFile ../security/security.xml -serverProps

```
../security/server.properties -jvmArgs
-Djava.security.auth.login.config="../security/og_jaas.config"
```

- Windows `startOgServer.bat catalogServer -clusterSecurityFile ../security/security.xml -serverProps ../security/server.properties -jvmArgs -Djava.security.auth.login.config="../security/og_jaas.config"`

I file `security.xml` e `server.properties` sono stati creati nel passo precedente di questo supporto didattico.

T

c. È possibile avviare un server contenitore sicuro utilizzando il seguente script. Eseguire il seguente script dalla directory bin:

- UNIX Linux `# startOgServer.sh c0 -objectGridFile ../xml/SecureSimpleApp.xml -deploymentPolicyFile ../xml/SimpleDP.xml -catalogServiceEndpoints localhost:2809 -serverProps ../security/server.properties -jvmArgs -Djava.security.auth.login.config="../security/og_jaas.config" -Djava.security.auth.policy="../security/og_auth.policy"`
- Windows `startOgServer.bat c0 -objectGridFile ../xml/SecureSimpleApp.xml -deploymentPolicyFile ../xml/SimpleDP.xml -catalogServiceEndpoints localhost:2809 -serverProps ../security/server.properties -jvmArgs -Djava.security.auth.login.config="../security/og_jaas.config" -Djava.security.auth.policy="../security/og_auth.policy"`

Notare le seguenti differenze rispetto al precedente comando di avvio del server contenitore:

- Utilizzare il file `SecureSimpleApp.xml` invece del file `SimpleApp.xml`.
- Aggiungere un altro argomento `-Djava.security.auth.policy` per impostare il file della politica di autorizzazione JAAS nel processo del server contenitore.

Utilizzare lo stesso comando come nel passo precedente del supporto didattico:

a. Navigare nella directory bin.

b. `java -classpath ../lib/objectgrid.jar;../applib/secsample.jar com.ibm.websphere.objectgrid.security.sample.guide.SecureSimpleApp ../security/client.properties manager manager1`

Poiché l'utente "manager" ha tutte le autorizzazioni alle mappe nell'accounting ObjectGrid, l'applicazione viene eseguita correttamente.

Adesso, invece di utilizzare l'utente "manager", utilizzare l'utente "cashier" per lanciare l'applicazione client.

c. Navigare nella directory bin.

d. `java -classpath ../lib/objectgrid.jar;../applib/secsample.jar com.ibm.ws.objectgrid.security.sample.guide.SecureSimpleApp ../security/client.properties cashier cashier1`

Il risultato è la seguente eccezione:

```
Exception in thread "P=387313:O=0:CT" com.ibm.websphere.objectgrid.TransactionException:
rolling back transaction, see caused by exception
 at com.ibm.ws.objectgrid.SessionImpl.rollbackPMapChanges(SessionImpl.java:1422)
  at com.ibm.ws.objectgrid.SessionImpl.commit(SessionImpl.java:1149)
  at com.ibm.ws.objectgrid.SessionImpl.mapPostInvoke(SessionImpl.java:2260)
  at com.ibm.ws.objectgrid.ObjectMapImpl.update(ObjectMapImpl.java:1062)
  at com.ibm.ws.objectgrid.security.sample.guide.SimpleApp.run(SimpleApp.java:42)
 at com.ibm.ws.objectgrid.security.sample.guide.SecureSimpleApp.main(SecureSimpleApp.java:27)
Caused by: com.ibm.websphere.objectgrid.ClientServerTransactionCallbackException:
   Client Services - received exception from remote server:
     com.ibm.websphere.objectgrid.TransactionException: transaction rolled back,
```

```
   see caused by Throwable
        at com.ibm.ws.objectgrid.client.RemoteTransactionCallbackImpl.processReadWriteResponse(
            RemoteTransactionCallbackImpl.java:1399)
        at com.ibm.ws.objectgrid.client.RemoteTransactionCallbackImpl.processReadWriteRequestAndResponse(
            RemoteTransactionCallbackImpl.java:2333)
        at com.ibm.ws.objectgrid.client.RemoteTransactionCallbackImpl.commit(RemoteTransactionCallbackImpl.java:557)
        at com.ibm.ws.objectgrid.SessionImpl.commit(SessionImpl.java:1079)
        ... 4 more
Caused by: com.ibm.websphere.objectgrid.TransactionException: transaction rolled back, see caused by Throwable
        at com.ibm.ws.objectgrid.ServerCoreEventProcessor.processLogSequence(ServerCoreEventProcessor.java:1133)
        at com.ibm.ws.objectgrid.ServerCoreEventProcessor.processReadWriteTransactionRequest
     (ServerCoreEventProcessor.java:910)
        at com.ibm.ws.objectgrid.ServerCoreEventProcessor.processClientServerRequest(ServerCoreEventProcessor.java:1285)

        at com.ibm.ws.objectgrid.ShardImpl.processMessage(ShardImpl.java:515)
        at com.ibm.ws.objectgrid.partition.IDLShardPOA._invoke(IDLShardPOA.java:154)
        at com.ibm.CORBA.poa.POAServerDelegate.dispatchToServant(POAServerDelegate.java:396)
        at com.ibm.CORBA.poa.POAServerDelegate.internalDispatch(POAServerDelegate.java:331)
        at com.ibm.CORBA.poa.POAServerDelegate.dispatch(POAServerDelegate.java:253)
        at com.ibm.rmi.iiop.ORB.process(ORB.java:503)
        at com.ibm.CORBA.iiop.ORB.process(ORB.java:1553)
        at com.ibm.rmi.iiop.Connection.respondTo(Connection.java:2680)
        at com.ibm.rmi.iiop.Connection.doWork(Connection.java:2554)
        at com.ibm.rmi.iiop.WorkUnitImpl.doWork(WorkUnitImpl.java:62)
        at com.ibm.rmi.iiop.WorkerThread.run(ThreadPoolImpl.java:202)
        at java.lang.Thread.run(Thread.java:803)
Caused by: java.security.AccessControlException: Access denied (
   com.ibm.websphere.objectgrid.security.MapPermission accounting.customer write)
        at java.security.AccessControlContext.checkPermission(AccessControlContext.java:155)
        at com.ibm.ws.objectgrid.security.MapPermissionCheckAction.run(MapPermissionCheckAction.java:141)
        at java.security.AccessController.doPrivileged(AccessController.java:275)
        at javax.security.auth.Subject.doAsPrivileged(Subject.java:727)
        at com.ibm.ws.objectgrid.security.MapAuthorizer$1.run(MapAuthorizer.java:76)
        at java.security.AccessController.doPrivileged(AccessController.java:242)
        at com.ibm.ws.objectgrid.security.MapAuthorizer.check(MapAuthorizer.java:66)
        at com.ibm.ws.objectgrid.security.SecuredObjectMapImpl.checkMapAuthorization(SecuredObjectMapImpl.java:429)
        at com.ibm.ws.objectgrid.security.SecuredObjectMapImpl.update(SecuredObjectMapImpl.java:490)
        at com.ibm.ws.objectgrid.SessionImpl.processLogSequence(SessionImpl.java:1913)
        at com.ibm.ws.objectgrid.SessionImpl.processLogSequence(SessionImpl.java:1805)
        at com.ibm.ws.objectgrid.ServerCoreEventProcessor.processLogSequence(ServerCoreEventProcessor.java:1011)
        ... 14 more
```

Questa eccezione si verifica in quanto l'utente "cashier" non ha l'autorizzazione di scrittura, quindi non può aggiornare il cliente mappa.

Adesso il sistema supporta l'autorizzazione. È possibile definire le politiche di autorizzazione per concedere autorizzazioni differenti a utenti diversi. Per ulteriori informazioni sull'autorizzazione, consultare le informazioni relative all'autorizzazione client dell'applicazione in *Guida alla programmazione*.

Operazioni successive

Completare il passo successivo del supporto didattico. Consultare “Java SE supporto didattico - Passo 4”.

Java SE supporto didattico - Passo 4

Il seguente passo spiega come abilitare uno strato della sicurezza per la comunicazione tra gli endpoint del proprio ambiente.

Prima di iniziare

Assicurarsi di aver completato “Java SE Supporto didattico - Passo 3” a pagina 212 prima di procedere con quest'attività.

Informazioni su questa attività

La topologia di eXtreme Scale supporta sia TLS/SSL (Transport Layer Security/Secure Sockets Layer) per la comunicazione tra gli endpoint ObjectGrid (client, server contenitori e server di catalogo). Questo passo del supporto didattico

si basa sulla procedura precedente per abilitare la sicurezza del trasporto.

Procedura

1. **Creare le chiavi TLS/SSL e i keystore**

   Al fine di abilitare la sicurezza dei trasporti, è necessario creare un keystore e truststore. Questo esercizio crea solo una coppia di key e trust-store. Questi store vengono utilizzati per i client ObjectGrid, i server contenitori e i server di catalogo e vengono creati con JDK keytool.

   - *Creare una chiave privata nel keystore*

     ```
     keytool -genkey -alias ogsample -keystore key.jks -storetype JKS
     -keyalg rsa -dname "CN=ogsample, OU=Your Organizational Unit, O=Your
     Organization, L=Your City, S=Your State, C=Your Country" -storepass
     ogpass -keypass ogpass -validity 3650
     ```

     Utilizzando questo comando, un keystore key.jks viene creata una chiave una chiave "ogsample" ivi memorizzata. Tale keystore key.jks verrà utilizzato come truststore SSL.

   - *Esportare il certificato pubblico*

     ```
     keytool -export -alias ogsample -keystore key.jks -file temp.key
     -storepass ogpass
     ```

     Utilizzando questo comando, il certificato pubblico della chiave "ogsample" viene estratto e memorizzato nel file temp.key.

   - *Importare il certificato pubblico del client nel truststore*

     ```
     keytool -import -noprompt -alias ogsamplepublic -keystore trust.jks
     -file temp.key -storepass ogpass
     ```

     Utilizzando questo comando, il certificato pubblico è stato aggiunto al keystore trust.jks. Trust.jks viene utilizzato come truststore SSL.

2. **Configurazione del file delle proprietà di ObjectGrid**

   In questo passo, è necessario configurare il file delle proprietà ObjectGrid per abilitare la sicurezza dei trasporti.

   Primo, copiare i file key.jks e trust.jks nella directory objectgridRoot/security.

   Impostare le seguenti proprietà nel file client.properties e server.properties.

   ```
   transportType=SSL-Required

   alias=ogsample
   contextProvider=IBMJSSE2
   protocol=SSL
   keyStoreType=JKS
   keyStore=../security/key.jks
   keyStorePassword=ogpass
   trustStoreType=JKS
   trustStore=../security/trust.jks
   trustStorePassword=ogpass
   ```

   transportType: il valore di transportType è impostato su "SSL-Required", che significa che il trasporto richiede SSL. In questo modo tutti gli endpoint ObjectGrid (i client, i server di catalogo e i server contenitore) devono avere una configurazione SSL impostata e tutta la comunicazione di trasporto verrà criptata.

   Le altre proprietà vengono utilizzate per impostare le configurazioni SSL. Consultare le informazioni relative ai protocolli TLS (Transport Layer Security) e SSL (Secure Sockets Layer) in *Guida alla gestione* per maggiori dettagli. Assicurarsi di seguire le istruzioni in questa sezione per aggiornare il file orb.properties.

   Assicurarsi di seguire questa pagina per aggiornare il file `orb.properties`.

Nel file `server.properties` , è necessario aggiungere una proprietà clientAuthentication aggiuntiva e impostarla su false. Dalla parte del server, non è necessario eseguire il trust del client.

```
clientAuthentication=false
```

3. **Eseguire l'applicazione**

   I comandi sono uguali ai comandi della sezione “Java SE Supporto didattico - Passo 3” a pagina 212.

   Utilizzare i seguenti comandi per avviare il server di catalogo.

   a. Navigare nella directory bin: `cd objectgridRoot/bin`

   b. Avviare il server di catalogo:

   - Linux UNIX

     ```
     startOgServer.sh catalogServer -clusterSecurityFile ../security/security.xml
     -serverProps ../security/server.properties -JMXServicePort 11001
     -jvmArgs -Djava.security.auth.login.config="../security/og_jaas.config"
     ```

   - Windows

     ```
     startOgServer.bat catalogServer -clusterSecurityFile ../security/security.xml
     -serverProps ../security/server.properties -JMXServicePort 11001 -jvmArgs
     -Djava.security.auth.login.config="../security/og_jaas.config"
     ```

     I file `security.xml` e `server.properties` sono stati creati nella pagina “Java SE supporto didattico sulla sicurezza - Passo 2” a pagina 205.

     Utilizzare l'opzione -JMXServicePort per specificare esplicitamente la porta JMX per il server. Questa opzione è obbligatoria per utilizzare il comando xsadmin.

     Eseguire un server contenitore ObjectGrid sicuro:

   c. Navigare di nuovo nella directory bin: `cd objectgridRoot/bin`

   d.

   - Linux UNIX

     ```
     startOgServer.sh c0 -objectGridFile ../xml/SecureSimpleApp.xml
     -deploymentPolicyFile ../xml/SimpleDP.xml -catalogServiceEndpoints
     localhost:2809 -serverProps ../security/server.properties
     -JMXServicePort 11002 -jvmArgs
     -Djava.security.auth.login.config="../security/og_jaas.config"
     -Djava.security.auth.policy="../security/og_auth.policy"
     ```

   - Windows

     ```
     startOgServer.bat c0 -objectGridFile ../xml/SecureSimpleApp.xml
     -deploymentPolicyFile ../xml/SimpleDP.xml -catalogServiceEndpoints localhost:2809
     -serverProps ../security/server.properties -JMXServicePort 11002
     -jvmArgs -Djava.security.auth.login.config="../security/og_jaas.config"
     -Djava.security.auth.policy="../security/og_auth.policy"
     ```

   Notare le seguenti differenze rispetto al precedente comando di avvio del server contenitore:

   - Utilizzare SecureSimpleApp.xml anziché SimpleApp.xml
   - Aggiungere un altro -Djava.security.auth.policy per impostare il file della politica di distribuzione JAAS nel processo del server contenitore.

   Eseguire il seguente comando per l'autenticazione del client:

   a. `cd objectgridRoot/bin`

   b.

   ```
   javaHome/java -classpath ../lib/objectgrid.jar;../applib/secsample.jar
   com.ibm.websphere.objectgrid.security.sample.guide.SecureSimpleApp
   ../security/client.properties manager manager1
   ```

   Poichè l'utente "manager" ha l'autorizzazione a tutte le mappe nell'accounting ObjectGrid, l'applicazione funziona correttamente.

   È possibile utilizzare inoltre xsadmin per mostrare le opzioni mapsizes della griglia "accounting".

   - Navigare nella directory `objectgridRoot/bin`.

- Utilizzare il comando xsadmin con l'opzione -mapSizes nel modo che segue.
  - UNIX Linux

    ```
    xsadmin.sh -g accounting -m mapSet1 -mapsizes -p 11001 -ssl
    -trustpath ..\security\trust.jks -trustpass ogpass -trusttype jks
    -username manager -password manager1
    ```

  - Windows

    ```
    xsadmin.bat -g accounting -m mapSet1 -mapsizes -p 11001 -ssl
    -trustpath ..\security\trust.jks -trustpass ogpass -trusttype jks
    -username manager -password manager1
    ```

    Notare che si specifica la porta JMX del servizio catalogo utilizzando -p 11001 qui.

    Viene visualizzato il seguente output.

    ```
    This administrative utility is provided as a sample only and is not to
    be considered a fully supported component of the WebSphere eXtreme Scale product.
    Connecting to Catalog service at localhost:1099
    ********** Displaying Results for Grid - accounting, MapSet - mapSet1 **********
    *** Listing Maps for c0 ***
    Map Name: customer Partition #: 0 Map Size: 1 Shard Type: Primary
    Server Total: 1
    Total Domain Count: 1
    ```

Eseguire l'applicazione con un keystore non corretto

Se il truststore non contiene il certificato pubblico della chiave privata nel keystore, si otterrà un'eccezione che reclama che non è possibile effettuare il trust della chiave.

Per visualizzarla, creare un'altro keystore key2.jks.

```
keytool -genkey -alias ogsample -keystore key2.jks -storetype JKS
-keyalg rsa -dname "CN=ogsample, OU=Your Organizational Unit, O=Your
Organization, L=Your City, S=Your State, C=Your Country" -storepass
ogpass -keypass ogpass -validity 3650
```

Quindi modificare server.properties in modo che il keyStore punti a questo nuovo keystore key2.jks:

```
keyStore=../security/key2.jks
```

Eseguire il seguente comando per avviare il server di catalogo:

a. Navigare nella directory bin: `cd objectgridRoot/bin`

b. Avviare il server di catalogo:

   Linux UNIX

   ```
   startOgServer.sh c0 -objectGridFile ../xml/SecureSimpleApp.xml
   -deploymentPolicyFile ../xml/SimpleDP.xml -catalogServiceEndpoints localhost:2809
   -serverProps ../security/server.properties -jvmArgs
   -Djava.security.auth.login.config="../security/og_jaas.config"
   -Djava.security.auth.policy="../security/og_auth.policy"
   ```

   Windows

   ```
   startOgServer.bat c0 -objectGridFile ../xml/SecureSimpleApp.xml
   -deploymentPolicyFile ../xml/SimpleDP.xml -catalogServiceEndpoints localhost:2809
    -serverProps ../security/server.properties -jvmArgs
   -Djava.security.auth.login.config="../security/og_jaas.config"
   -Djava.security.auth.policy="../security/og_auth.policy"
   ```

   Viene visualizzata la seguente eccezione:

   ```
   Caused by: com.ibm.websphere.objectgrid.ObjectGridRPCException:
       com.ibm.websphere.objectgrid.ObjectGridRuntimeException:
           SSL connection fails and plain socket cannot be used.
   ```

   Infine, modificare di nuovo il file server.properties affinchè utilizzi di nuovo il file key.jks.

Esempio e supporto didattico del servizio dati REST

Questo argomento descrive il modo in cui iniziare velocemente ad utilizzare il servizio dati REST di WebSphere eXtreme Scale. Le istruzioni vengono fornite per WebSphere Application Server versione 7.0,WebSphere Application Server Community Edition e per Apache Tomcat.

Informazioni su questa attività

L'esempio incluso dispone del codice sorgente e codici binari compilati per eseguire una griglia eXtreme Scale partizionata. Questo esempio illustra il modo in cui creare una griglia semplice, modellare i dati utilizzando le entità eXtreme Scale e fornisce due applicazioni client di riga comandi che consentono di aggiungere entità ed eseguire query su di esse utilizzando Java o C# (vedere Figura 1).

Il client Java di esempio utilizza l'API EntityManager di eXtreme Scale Java per la persistenza dei dati e l'esecuzione di query nella griglia. Questo client può essere eseguito in Eclipse o mediante uno script di riga comandi. Notare che il client Java di esempio non illustra il servizio dati REST, ma consente l'aggiornamento dei dati nella griglia, per cui i dati possono essere letti mediante un browser web o altri client. Il client Java di esempio e il browser web, come mostrato nella Figura 1, illustrano i client HTTP che utilizzano il servizio dati REST e i client eXtreme Scale Java che utilizzano la stessa griglia eXtreme Scale e i dati ivi contenuti.

Il client C# Microsoft WCF Data Services di esempio comunica con la griglia eXtreme Scale attraverso il servizio dati REST mediante .NET Framework. Il client WCF Data Services può essere utilizzato sia per aggiornare la griglia sia per eseguire query su di essa.

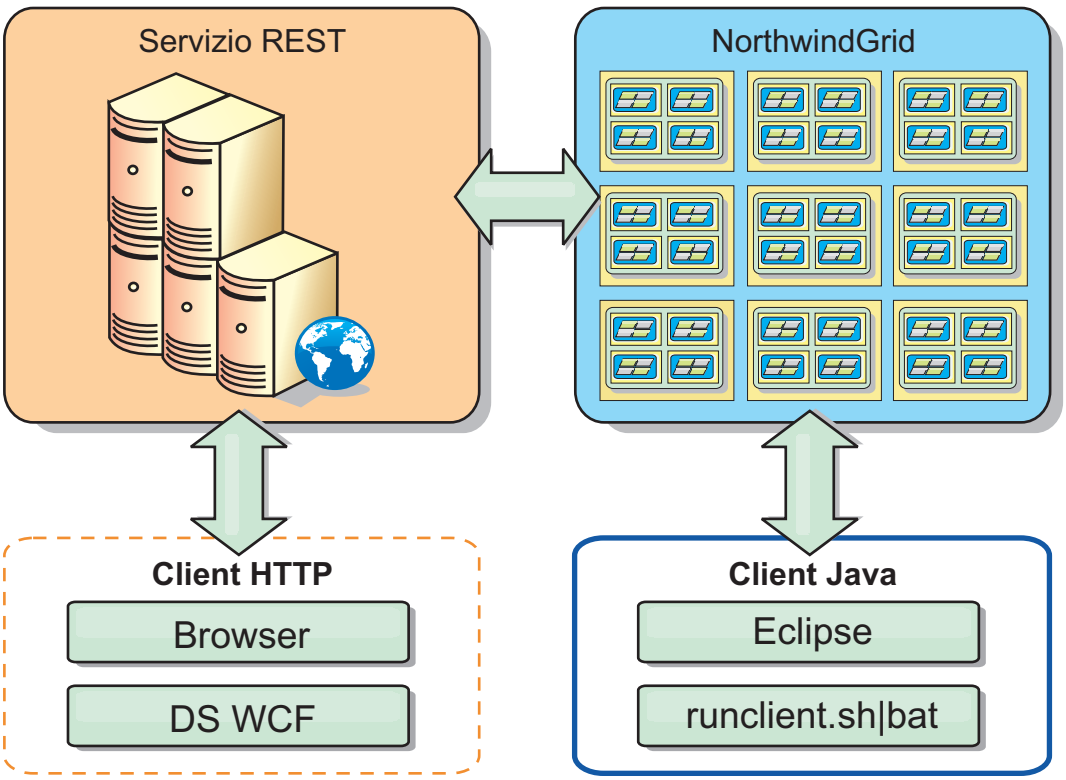


Figura 44. Introduzione alla topologia di esempio

Procedura

1. Configurare e avviare la griglia eXtreme Scale. Consultare "Abilitazione del servizio dati REST" a pagina 221.
2. Configurare e avviare il servizio dati REST in un server web. Consultare "Configurazione dei server delle applicazioni per il servizio dati REST" a pagina 230.
3. Eseguire un client per interagire con il servizio dati REST. Sono disponibili due opzioni:
   a. Eseguire il client Java di esempio per inserire dati nella griglia utilizzando l'API EntityManager ed eseguire query sui dati nella griglia utilizzando un browser web e il servizio dati REST di eXtreme Scale. Consultare "Utilizzo di un client Java con i servizi dati REST" a pagina 239.
   b. Eseguire il client C# WCF Data Services di esempio. Consultare "Visual Studio 2008 WCF client con servizio dati REST" a pagina 241.

Convenzioni sulle directory

Questa sezione descrive molti esempi e sintassi di riga comandi che fanno riferimento a directory speciali, come ad esempio *wxs_install_root* e *wxs_home*. Tali directory sono definite come di seguito riportato.

root_installazione_wxs
La directory *root_installazione_wxs* è la directory root in cui vengono installati i file del prodotto WebSphere eXtreme Scale. Questa può essere la directory in cui viene estratto il file zip di prova oppure la directory in cui viene installato il prodotto eXtreme Scale.

- Esempio per l'estrazione del file zip di prova:
  `/opt/IBM/WebSphere/eXtremeScale`
- Esempio per l'installazione di eXtreme Scale in una directory autonoma:
  `/opt/IBM/eXtremeScale`
- Esempio per l'integrazione di eXtreme Scale con WebSphere Application Server:
  `/opt/IBM/WebSphere/AppServer`

home_wxs
La directory *wxs_home* è la directory root delle librerie WebSphere eXtreme Scale, degli esempi e dei componenti. Questa directory è uguale alla directory *root_installazione_wxs* quando viene estratto il file zip di prova. Per le installazioni di tipo autonomo, questa è la sottodirectory ObjectGrid nella directory *root_installazione_wxs*. Per le installazioni che vengono integrate con WebSphere Application Server, questa directory è `optionalLibraries/ObjectGrid` nella directory *root_installazione_wxs*.

- Esempio per l'estrazione del file zip di prova:
  `/opt/IBM/WebSphere/eXtremeScale`
- Esempio per l'installazione di eXtreme Scale in una directory autonoma:
  `/opt/IBM/eXtremeScale/ObjectGrid`
- Esempio per l'integrazione di eXtreme Scale con WebSphere Application Server:
  `/opt/IBM/WebSphere/AppServer/optionalLibraries/ObjectGrid`

root_was
La directory *root_was* è la directory root dell'installazione di WebSphere Application Server:

/opt/IBM/WebSphere/AppServer

home_serviziorest
La directory *restservice_home* è la directory in cui sono ubicati gli esempi e le librerie del servizio dati REST di eXtreme Scale. Questa directory si chiama *restservice* ed è una directory secondaria di *wxs_home*.

- Esempio per distribuzioni autonome:
  `/opt/IBM/WebSphere/eXtremeScale/ObjectGrid/restservice`
- Esempio per distribuzioni integrate con WebSphere Application Server:
  `/opt/IBM/WebSphere/AppServer/optionalLibraries/ObjectGrid/restservice`

root_tomcat
La directory *root_tomcat* è la directory root dell'installazione di Apache Tomcat.

`/opt/tomcat5.5`

root_wasce
La directory *root_wasce* è la directory root dell'installazione di WebSphere Application Server Community Edition.

`/opt/IBM/WebSphere/AppServerCE`

home_java
java_home è la directory root di un'installazioneJRE (Java Runtime Environment).

`/opt/IBM/WebSphere/eXtremeScale/java`

Abilitazione del servizio dati REST

Il servizio dati REST può rappresentare i metadati dell'entità WebSphere eXtreme Scale per rappresentare ciascuna entità come una EntitySet.

Avvio di un griglia di esempio eXtreme Scale

In generale, prima di avviare un servizio dati REST, avviare la griglia eXtreme Scale. I passi di seguito riportati avviano un singolo processo di servizio catalogo eXtreme Scale e due processi di server contenitori.

WebSphere eXtreme Scale può essere installato utilizzando tre differenti metodi:

- Installazione della versione di prova
- Distribuzione autonoma
- Distribuzione integrata di WebSphere Application Server

Modello di dati scalabile in eXtreme Scale

L'esempio Microsoft Northwind utilizza la tabella Order Detail per stabilire un'associazione molti-a-molti tra Order e Product.

Le specifiche ORMS (Object to Relational Mapping) come ADO.NET Entity Framework e JPA (Java Persistence API) possono associare le tabelle e le relazioni utilizzando le entità. Tuttavia, questa architettura non è scalabile. Per una valida esecuzione, tutto deve trovarsi sulla stessa macchina o su un dispendioso cluster di macchine.

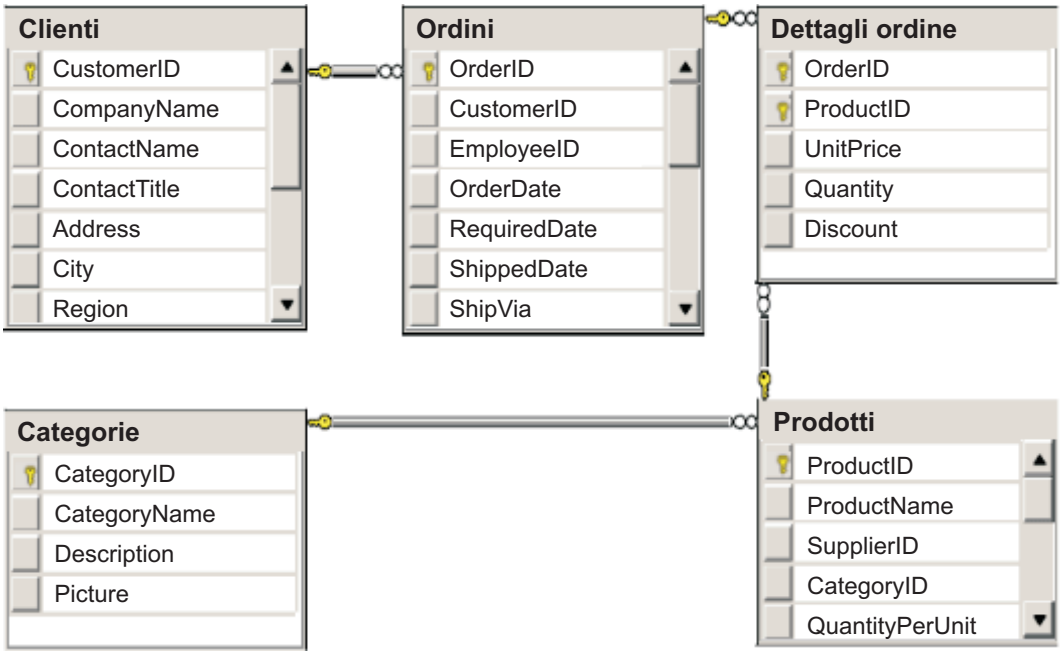


Figura 45. Diagramma dello schema di esempio di Microsoft SQL Server Northwind

Per creare una versione scalabile dell'esempio, le entità devono essere modellate in modo che ciascuna entità o gruppo di entità correlate possano essere partizionate in base ad una singola chiave. Creando le partizioni su una singola chiave, le richieste possono essere distribuite su più server indipendenti. Per ottenere questa configurazione, le entità devono essere divise in due strutture ad albero: la struttura ad albero Cliente e ordine e la struttura ad albero Prodotto e Categoria In questo modello ciascuna struttura ad albero può essere partizionata indipendentemente e perciò può crescere a differenti velocità incrementando la scalabilità.

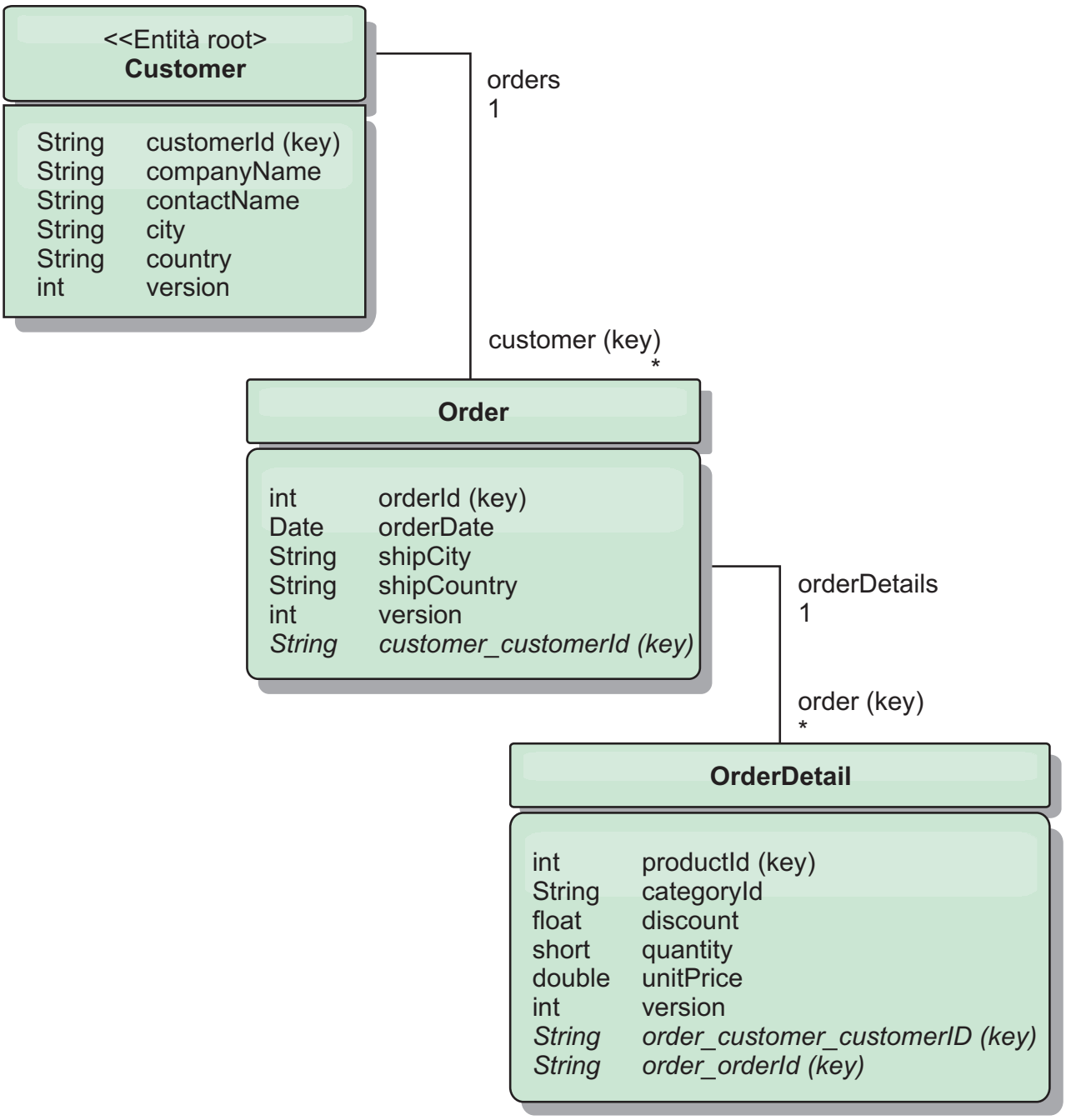


Figura 46. Diagramma dello schema dell'entità Customer e Order

Ad esempio, sia Order che Product hanno come chiavi numeri univoci, interi separati. Infatti, le tabelle Order e Product sono realmente indipendenti l'una dall'altra. Ad esempio, considerare l'effetto della dimensione di un catalogo (numero di prodotti venduti) con il numero totale di ordini. Intuitivamente potrebbe sembrare che avere molti prodotti implichi anche avere molti ordini, ma non è necessariamente così. Se questo fosse vero, si potrebbe facilmente aumentare le vendite semplicemente aggiungendo più prodotti al catalogo. Gli ordini e i prodotti hanno le proprie tabelle indipendenti. È possibile estendere ulteriormente questo concetto in modo che i prodotti e gli ordini abbiano ognuno le proprie griglie di dati separate. Con le griglie indipendenti, è possibile controllare il numero di partizioni e di server, in aggiunta alla dimensione di ciascuna griglia,

separatamente, in modo che l'applicazione possa sfruttare la scalabilità. Se si raddoppia la dimensione del catalogo, è necessario raddoppiare i prodotti della griglia, ma l'ordine potrebbe non essere modificato. Al contrario è vero per un aumento di ordini, o un previsto aumento di ordini.

Nello schema, un cliente (Customer) ha zero o più ordini e un ordine (Order) ha elementi di riga (OrderDetail), ognuno con un prodotto specifico. Un prodotto viene identificato da un ID (la chiave del prodotto) in ciascun OrderDetail. Una singola griglia memorizza i clienti e gli ordini e OrderDetails con il cliente come entità root della griglia. È possibile richiamare i clienti mediante l'ID, ma è necessario richiamare gli ordini che iniziando dall'ID del cliente. Quindi l'ID cliente viene aggiunto all'ordine come parte della sua chiave. Allo stesso modo, l'ID cliente e l'ID ordine fanno parte dell'id OrderDetail.

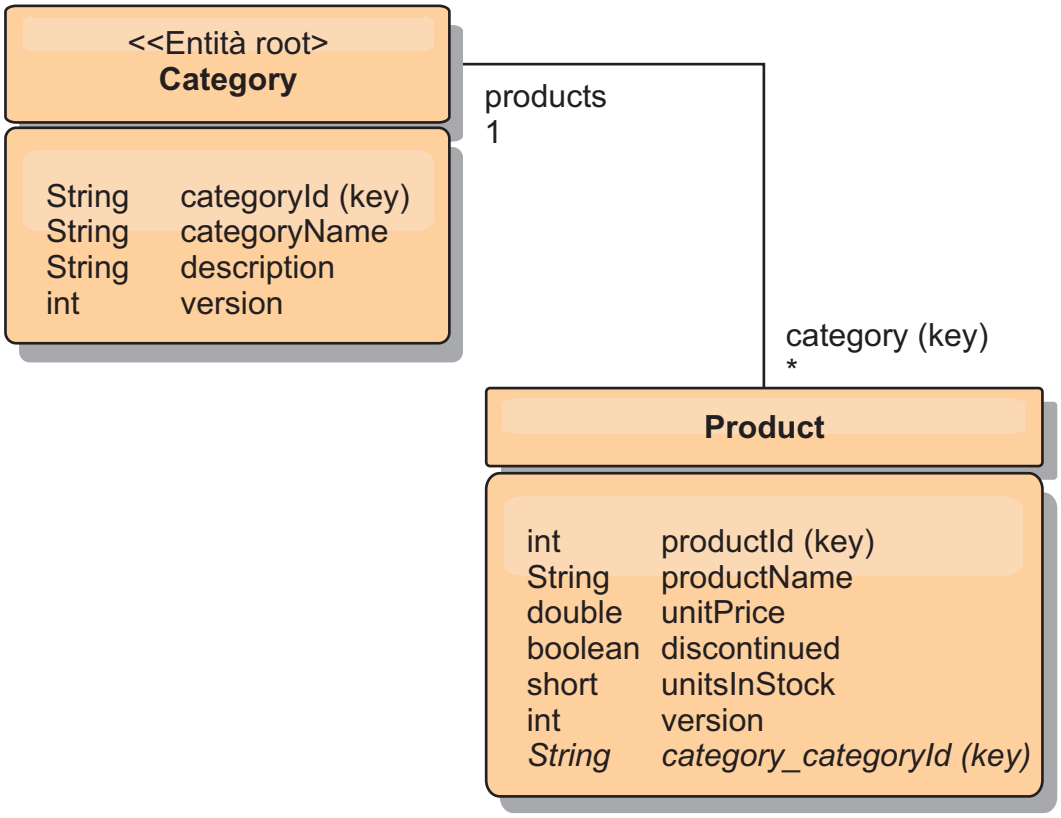


Figura 47. Diagramma dello schema dell'entità Category e Product

Nello schema Category e Product, Category è la root dello schema. Utilizzando questo schema i clienti possono eseguire la query sui prodotti per categoria. Consultare “Richiamo e aggiornamento dei dati con REST” per ulteriori dettagli sulle associazioni della chiave e la loro importanza.

Richiamo e aggiornamento dei dati con REST

Il protocollo OData richiede che tutte le entità possano essere indirizzate nella loro forma canonica. Questo significa che ciascuna entità deve includere la chiave dell'entità root con partizioni (la root dello schema).

Quello che segue è un esempio di come utilizzare l'associazione di un'entità root in cui indirizzare un elemento child:

```
/Customer('ACME')/order(100)
```

In WCF Data Services, l'entità child deve essere indirizzabile direttamente, ossia la chiave nella root dello schema deve far parte della chiave dell'elemento child: /Order(customer_customerId='ACME', orderId=100). Questo si ottiene creando un'associazione all'entità root in cui anche l'associazione uno-a-uno o molti-a-uno all'entità root viene identificata come una chiave. Quando le entità sono incluse come parte della chiave, gli attributi dell'entità parent vengono esposti come proprietà della chiave.

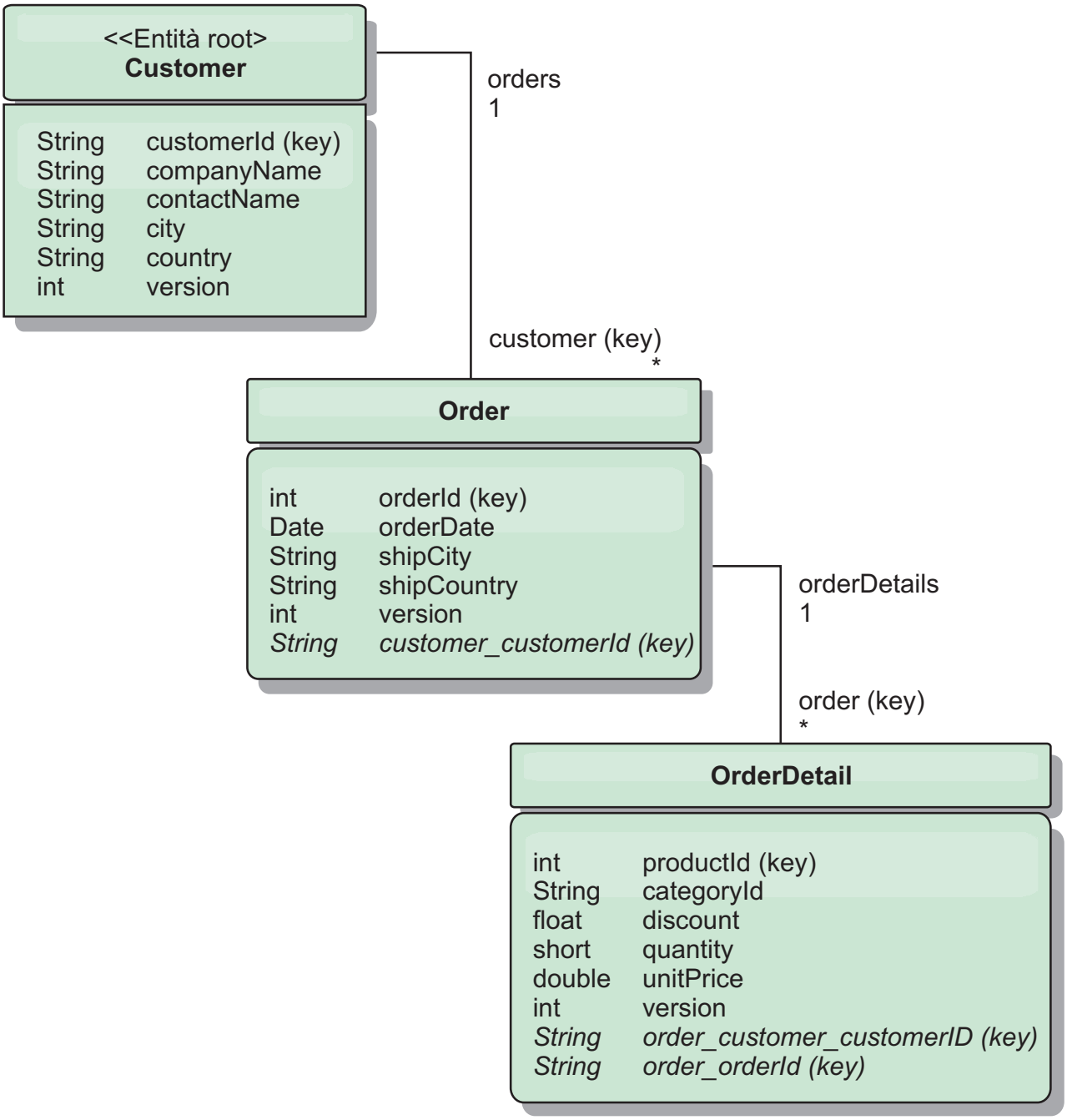


Figura 48. Diagramma dello schema entità Customer e Order

Il diagramma dello schema entità Customer/Order illustra in che modo ciascuna entità viene suddivisa in partizioni utilizzando Customer. L'entità Order include Customer come parte della relativa chiave ed è pertanto accessibile direttamente. Il

servizio dati REST espone tutte le associazioni di chiavi come proprietà individuali: Order ha customer_customerId e OrderDetail ha order_customer_customerId e order_orderId.

Utilizzando l'API EntityManager, è possibile trovare Order utilizzando Customer e l'id ordine:

```
transaction.begin();
// Look-up the Order using the Customer.  We only include the Id
// in the Customer class when building the OrderId key instance.
Order order = (Order) em.find(Order.class,
    new OrderId(100, new Customer('ACME')));
...
transaction.commit();
```

Quando si utilizza il servizio dati REST, Order può essere richiamato con uno degli URL:

- `/Order(orderId=100, customer_customerId='ACME')`
- `/Customer('ACME')/orders?$filter=orderId eq 100`

La chiave del cliente viene indirizzata utilizzando il nome attributo dell'entità Customer, un carattere di sottolineatura e il nome attributo dell'id cliente: customer_customerId.

Un'entità può includere anche un'entità non root come parte della relativa chiave se tutti i predecessori dell'entità non root hanno associazioni di chiavi alla root. In questo esempio, OrderDetail ha un'associazione di chiave a Order e Order ha un'associazione di chiave all'entità Customer root. Utilizzo dell'API EntityManager:

```
transaction.begin();
// Construct an OrderDetailId key instance.  It includes
// The Order and Customer with only the keys set.
Customer customerACME = new Customer("ACME");
Order order100 = new Order(100, customerACME);
OrderDetailId orderDetailKey =
    new OrderDetailId(order100, "COMP");
OrderDetail orderDetail = (OrderDetail)
    em.find(OrderDetail.class, orderDetailKey);
...
```

Il servizio dati REST consente l'indirizzamento di OrderDetail direttamente:

```
/OrderDetail(productId=500, order_customer_customerId='ACME', order_orderId =100)
```

L'associazione dall'entità OrderDetail all'entità Product è stata interrotta per consentire la suddivisione in partizioni dell'inventario Orders and Product in modo indipendente. L'entità OrderDetail memorizza la categoria e l'id prodotto invece di una relazione difficile. Dividendo i due schemi di entità, viene acceduta una sola partizione alla volta.

Lo schema Category e Product, illustrato nel diagramma mostra che l'entità root è Category ed ogni Product ha un'associazione ad un'entità Category. L'entità Category viene inclusa nell'identità di Product. Il servizio dati REST espone una proprietà della chiave: category_categoryId che consente di indirizzare direttamente Product.

Poiché Category è l'entità root, in un ambiente con partizioni, Category deve essere noto per poter trovare Product. Utilizzando l'API EntityManager, la transazione deve essere bloccata sull'entità Category prima di trovare Product.

Utilizzo dell'API EntityManager:

```
transaction.begin();
// Create the Category root entity with only the key.  This
// allows us to construct a ProductId without needing to find
// The Category first.  The transaction is now pinned to the
// partition where Category "COMP" is stored.
Category cat = new Category("COMP");
Product product = (Product) em.find(Product.class,
    new ProductId(500, cat));
...
```

Il servizio dati REST consente di indirizzare Product direttamente:

```
/Product(productId=500, category_categoryId='COMP')
```

Avvio di una griglia autonoma per i servizi dati REST

Effettuare i passi di seguito riportati per avviare la griglia di esempio del servizio REST di WebSphere eXtreme Scale per una distribuzione eXtreme Scale autonoma.

Prima di iniziare

Installare la versione di prova di WebSphere eXtreme Scale o il prodotto completo:

- Installare la versione autonoma del prodotto WebSphere eXtreme Scale 7.1 e applicare le eventuali fix successive.
- Scaricare ed estrarre la versione di prova di WebSphere eXtreme Scale Versione 7.1, che include il servizio dati REST di WebSphere eXtreme Scale.

Informazioni su questa attività

Avviare la griglia di esempio WebSphere eXtreme Scale.

Procedura

1. Avviare il processo del servizio catalogo. Aprire una finestra di riga comandi o di terminale e impostare la variabile di ambiente JAVA_HOME:
   - Linux UNIX `export JAVA_HOME=java_home`
   - Windows `set JAVA_HOME=java_home`
2. `cd restservice_home/gettingstarted`
3. Avviare il processo del servizio catalogo. Per avviare il servizio *senza* la sicurezza eXtreme Scale, utilizzare i seguenti comandi.
   - Linux UNIX `./runcat.sh`
   - Windows `runcat.bat`

   Per avviare il servizio con la sicurezza eXtreme Scale, utilizzare i seguenti comandi.
   - Linux UNIX `./runcat_secure.sh`
   - Windows `runcat_secure.bat`
4. Avviare due processi del server contenitore. Aprire un'altra finestra di riga comandi o di terminale e impostare la variabile di ambiente JAVA_HOME:
   - Linux UNIX `export JAVA_HOME=java_home`
   - Windows `set JAVA_HOME=java_home`
5. `cd restservice_home/gettingstarted`
6. Avviare un processo del server contenitore:

   Per avviare il server senza la sicurezza eXtreme Scale, utilizzare i seguenti comandi:

- Linux UNIX `./runcontainer.sh container0`
- Windows `runcontainer.bat container0`

Per avviare il server con la sicurezza eXtreme Scale, utilizzare i seguenti comandi.

- Linux UNIX `./runcontainer_secure.sh container0`
- Windows `runcontainer_secure.bat container0`

7. Aprire un'altra finestra di riga comandi o di terminale e impostare la variabile di ambiente JAVA_HOME:
   - Linux UNIX `export JAVA_HOME=`*`java_home`*
   - Windows `set JAVA_HOME=`*`java_home`*
8. `cd restservice_home/gettingstarted`
9. Avviare un secondo processo del server contenitore.

   Per avviare il server senza la sicurezza eXtreme Scale, utilizzare i seguenti comandi.

   - Linux UNIX `./runcontainer.sh container1`
   - Windows `runcontainer.bat container1`

   Per avviare il server con la sicurezza eXtreme Scale, utilizzare i seguenti comandi.

   - Linux UNIX `./runcontainer_secure.sh container1`
   - Windows `runcontainer_secure.bat container1`

Risultati

Prima di procedere con i passi successivi, attendere che i contenitori eXtreme Scale siano pronti. I server contenitore sono pronti quando nella finestra di terminale viene visualizzato il seguente messaggio:

`CWOBJ1001I: ObjectGrid Server `*`container_name`*` is ready to process requests. (CWOBJ1001I: ObjectGrid Server nome_contenitore è pronto per elaborare le richieste.)`

Dove *nome contenitore* è il nome del contenitore che è stato avviato.

Avvio di una griglia per i servizi dati REST in WebSphere Application Server

Effettuare questi passi per avviare una griglia di dati autonoma di esempio del servizio REST di WebSphere eXtreme Scale per una distribuzione WebSphere eXtreme Scale integrata con WebSphere Application Server. Nonostante WebSphere eXtreme Scale sia integrato con WebSphere Application Server, questi passi avviano un contenitore e un processo del servizio catalogo WebSphere eXtreme Scale.

Prima di iniziare

Installare il prodotto WebSphere eXtreme Scale Versione 7.1 nella directory di installazione di WebSphere Application Server Versione 7.0.0.5 o versioni successive (con la sicurezza disabilitata), convertire almeno un profilo server delle applicazioni.

Informazioni su questa attività

Avviare la griglia di esempio di WebSphere eXtreme Scale.

Procedura

1. Avviare il processo del servizio catalogo. Aprire una finestra di riga comandi o di terminale e impostare la variabile di ambiente JAVA_HOME:
   - Linux UNIX `export JAVA_HOME=java_home`
   - Windows `set JAVA_HOME=java_home`

   `cd restservice_home/gettingstarted`
2. Avviare il processo del servizio catalogo.

   Per avviare il server senza la sicurezza eXtreme Scale, utilizzare i seguenti comandi.
   - Linux UNIX `./runcat.sh`
   - Windows `runcat.bat`

   Per avviare il server con la sicurezza eXtreme Scale, utilizzare i seguenti comandi.
   - Linux UNIX `./runcat_secure.sh`
   - Windows `runcat_secure.bat`
3. Avviare due processi del server contenitore. Aprire un'altra finestra di riga comandi o di terminale e impostare la variabile di ambiente JAVA_HOME:
   - Linux UNIX `export JAVA_HOME=java_home`
   - Windows `set JAVA_HOME=java_home`
4. Avviare un processo del server contenitore.

   Per avviare il server senza la sicurezza eXtreme Scale, utilizzare i seguenti comandi.
   a. Aprire una finestra della riga comandi.
   b. `cd restservice_home/gettingstarted`
   c. Per avviare il server *senza* la sicurezza eXtreme Scale utilizzare i seguenti comandi.
      - Linux UNIX `./runcontainer.sh container0`
      - Windows `runcontainer.bat container0`
   d. Per avviare il server con la sicurezza eXtreme Scale, utilizzare i seguenti comandi.
      - Linux UNIX `./runcontainer_secure.sh container0`
      - Windows `runcontainer_secure.bat container0`
5. Avviare un secondo processo del server contenitore.
   a. Aprire una finestra della riga comandi.
   b. `cd restservice_home/gettingstarted`
   c. Per avviare il server *senza* la sicurezza eXtreme Scale utilizzare i seguenti comandi.
      - Linux UNIX `./runcontainer.sh container1`
      - Windows `runcontainer.bat container1`

d. Per avviare il server *con* la sicurezza eXtreme Scale, utilizzare i seguenti comandi.
   - Linux UNIX `./runcontainer_secure.sh container1`
   - Windows `runcontainer_secure.bat container1`

Risultati

Prima di procedere con i passi successivi, attendere che i server contenitori siano pronti. I server contenitore sono pronti quando viene visualizzato il seguente messaggio:

`CWOBJ1001I: ObjectGrid Server container_name is ready to process requests. (CWOBJ1001I: ObjectGrid Server nome_contenitore è pronto per elaborare le richieste.)`

Dove *nome_contenitore* è il nome del contenitore avviato nel passo precedente.

Configurazione dei server delle applicazioni per il servizio dati REST

Avvio dei servizi dati REST in WebSphere Application Server Versione 7.0

Questo argomento descrive come configurare ed avviare il servizio dati REST di eXtreme Scale utilizzando WebSphere Application Server Versione 7.0.

Prima di iniziare

Verificare che la griglia eXtreme Scale di esempio sia stata avviata. Consultare "Abilitazione del servizio dati REST" a pagina 221 per i dettagli su come avviare la griglia.

Procedura

1. Scaricare ed installare WebSphere Application Server Versione 7.0 per Developers.

   Limitazione: Non abilitare la sicurezza.
2. Scaricare ed installare WebSphere Application Server Versione 7.0 Fix Pack 5 o successive.
3. Aggiungere il JAR runtime del client WebSphere eXtreme Scale, il file `wsogclient.jar` e il file JAR o la directory di configurazione del servizio dati REST al classpath del server delle applicazioni:
   a. Aprire la console di gestione WebSphere Application Server.
   b. Navigare su **Ambiente** → **Librerie condivise**
   c. Fare clic su **Nuovo**
   d. Aggiungere le seguenti voci nei campi:
      1) Name: extremescale_client_v71
      2) Classpath: `wxs_home/lib/wsogclient.jar`
   e. Fare clic su **OK**
   f. Fare clic su **Nuovo**
   g. Aggiungere le seguenti voci nei campi appropriati:
      1) Name: `extremescale_gettingstarted_config`

2) Classpath:
   - `restservice_home/gettingstarted/restclient/bin`
   - `restservice_home/gettingstarted/common/bin`

Attenzione: Aggiungere ciascun percorso su una riga separata.

h. Fare clic su **OK**

i. Salvare le modifiche nella configurazione principale

4. Installare il file EAR del servizio dati REST, `wxsrestservice.ear`, in WebSphere Application Server utilizzando la console di gestione:
   a. Aprire la console di gestione WebSphere Application Server
   b. Navigare su **Applicazioni** → **Nuova applicazione**
   c. Andare al file `restservice_home/lib/wxsrestservice.ear`, selezionarlo e fare clic su **Avanti**
   d. Selezionare le opzioni avanzate di installazione e fare clic su **Avanti**
   e. Nel pannello delle avvertenze sulla sicurezza dell'applicazione, fare clic su **Continua**
   f. Scegliere le opzioni di installazione predefinite e fare clic su **Avanti**
   g. Scegliere un server da associare all'applicazione e fare clic su **Avanti**
   h. Nella pagina di ricaricamento JSP, utilizzare le impostazione predefinita e fare clic su **Avanti**
   i. Nella pagina delle librerie condivise, associare il modulo `wxsrestservice.war` alle seguenti librerie condivise definite:
      - extremescale_client_v71
      - extremescale_ gettingstarted _config
   j. Nella pagina Mappa relazioni delle librerie condivise, utilizzare le impostazione predefinita e fare clic su **Avanti**
   k. Nella pagina Mappa host virtuali, utilizzare le impostazione predefinita e fare clic su **Avanti**
   l. Nella pagina delle root di contesto mappa, impostare la root di contesto su: `wxsrestservice`.
   m. Nel pannello Riepilogo, fare clic su **Fine** per completare l'installazione.
   n. Salvare le modifiche nella configurazione principale
5. Avviare il server delle applicazioni e l'applicazione del servizio dati REST di eXtreme Scale, wxsrestservice. Dopo aver avviato l'applicazione, riesaminare il file `SystemOut.log` per il server delle applicazioni e verificare che venga visualizzato il seguente messaggio: `CWOBJ4000I: The WebSphere eXtreme Scale REST data service has been started. - (CWOBJ4000I: Il servizio dati REST di WebSphere eXtreme Scale è stato avviato.)`
6. Verificare che il servizio dati REST stia funzionando
   a. Aprire un browser e navigare su: `http://localhost:9080/wxsrestservice/restservice/NorthwindGrid`. Viene visualizzato il documento di servizio NorthwindGrid.
   b. Navigare fino a: `http://localhost:9080/wxsrestservice/restservice/NorthwindGrid/$metadata`. Viene visualizzato il documento EDMX (Entity Model Data Extensions).
7. Per arrestare i processi della griglia, utilizzare `CTRL+C` nella relativa finestra comandi.

Avvio servizi dati REST con WebSphere eXtreme Scale integrato in WebSphere Application Server 7.0

Questa sezione descrive come configurare e avviare il servizio dati REST di eXtreme Scale utilizzando WebSphere Application Server versione 7.0 che è stato integrato ed ampliato con WebSphere eXtreme Scale.

Prima di iniziare

Verificare che la griglia di eXtreme Scale autonoma e di esempio sia stata avviata. Consultare "Abilitazione del servizio dati REST" a pagina 221 per i dettagli su come avviare la griglia.

Informazioni su questa attività

Per iniziare ad utilizzare il servizio dati REST di WebSphere eXtreme Scale utilizzando WebSphere Application Server, completare la seguente procedura:

Procedura

1. Aggiungere il JAR di configurazione di esempio del servizio dati REST di WebSphere eXtreme Scale al percorso di classe:
   a. Aprire la console di gestione di WebSphere
   b. Navigare su Ambiente -> Librerie condivise
   c. Fare clic su Nuovo
   d. Aggiungere le seguenti voci nei campi appropriati:
      1) Nome: extremescale_gettingstarted_config
      2) Percorso di classe
         - `restservice_home/gettingstarted/restclient/bin`
         - `restservice_home/gettingstarted/common/bin`

         Attenzione: Ogni percorso deve apparire su una riga diversa.
   e. Fare clic su **OK**
   f. Salvare le modifiche sulla configurazione principale
2. Installare il file EAR del servizio dati REST, wxsrestservice.ear, in WebSphere Application Server utilizzando la console di gestione di WebSphere:
   a. Aprire la console di gestione di WebSphere
   b. Navigare su Applications -> New Application
   c. Selezionare il file `restservice_home/lib/wxsrestservice.ear` nel file system. Selezionare il file e fare clic su **Successivo**.
   d. Selezionare le opzioni di installazione dettagliate e fare clic su **Avanti**.
   e. Dalle schermate di avvertenza sulla sicurezza, fare clic su **Continua**.
   f. Selezionare le opzioni di installazione predefinite e fare clic su **Avanti**.
   g. Selezionare un server da associare cui associare il modulo wxsrestservice.war e fare clic su **Avanti**.
   h. Sulla pagina di ricaricamento JSP, utilizzare le impostazioni predefinite e fare clic su **Avanti**.
   i. Nella pagina delle librerie condivise, associare il modulo "wxsrestservice.war" alle seguenti librerie condivise definite durante il passo 1: `extremescale_ gettingstarted _config`
   j. Nella pagina Mappa relazioni della librerie condivise, utilizzare le impostazioni predefinite e fare clic su **Avanti**.

k. Nella pagina degli host virtuali della mappa, utilizzare le impostazioni predefinite e fare clic su **Avanti**.

l. Nella pagina delle root di contesto mappa, impostare la root di contesto su: wxsrestservice.

m. Sulla schermata di riepilogo, fare clic su **Fine** per completare l'installazione.

n. Salvare le modifiche nella configurazione principale.

3. Se la griglia di eXtreme Scale è stata avviata con la sicurezza eXtreme Scale abilitata, impostare la seguente proprietà nel file `restservice_home/gettingstarted/restclient/bin/wxsRestService.properties` file.

```
ogClientPropertyFile=restservice_home/gettingstarted/security/security.ogclient.properties
```

4. Avviare il server delle applicazioni e l'applicazione "wxsrestservice" del servizio dati REST di eXtreme Scale.

   Dopo aver avviato l'applicazione, riesaminare SystemOut.log per il server delle applicazioni e verificare che venga visualizzato il seguente messaggio: `CWOBJ4000I: Il servizio dati REST di WebSphere eXtreme Scale è stato avviato.`

5. Verificare che il servizio dati REST sta funzionando:

   a. Aprire un browser e navigare su:

      http://localhost:9080/wxsrestservice/restservice/NorthwindGrid

      Il documento del servizio per NorthwindGrid viene visualizzato.

   b. Navigare su:

      http://localhost:9080/wxsrestservice/restservice/NorthwindGrid/$metadata

      Il documento EDMX (Entity Model Data Extensions) viene visualizzato

6. Per arrestare i processi della griglia, utilizzare CTRL+C nella rispettiva finestra comandi per arrestare il processo.

Avvio del servizio dati REST in WebSphere Application Server Community Edition

Questo argomento descrive come configurare ed avviare il servizio dati REST di eXtreme Scale utilizzando WebSphere Application Server Community Edition.

Prima di iniziare

Verificare che la griglia dati di esempio sia avviata. Consultare “Abilitazione del servizio dati REST” a pagina 221 per dettagli su come avviare la griglia.

Procedura

1. Scaricare ed installare WebSphere Application Server Community Edition Versione 2.1.1.3 o successiva in `wasce_root`, come: `/opt/IBM/wasce`
2. Avviare il server WebSphere Application Server Community Edition eseguendo il seguente comando:
   - Linux UNIX `wasce_root/bin/startup.sh`
   - Windows `wasce_root/bin/startup.bat`
3. Se la griglia eXtreme Scale è stata avviata con la sicurezza eXtreme Scale abilitata, impostare le seguenti proprietà nel file `restservice_home/gettingstarted/restclient/bin/wxsRestService.properties`.

```
ogClientPropertyFile=restservice_home/gettingstarted/security/security.ogclient.properties
loginType=none
```

4. Installare il servizio dati REST di eXtreme Scale e l'esempio fornito nel server WebSphere Application Server Community Edition:
   a. Aggiungere il JAR runtime del client ObjectGrid al repository di WebSphere Application Server Community Edition:
      1) Aprire la console di gestione WebSphere Application Server Community Edition ed eseguire l'accesso.

         Suggerimento: L'URL predefinito è: `http://localhost:8080/console`. L'ID utente predefinito è `system` e la password è `manager`.
      2) Fare clic su **Repository**, nella cartella Servizi.
      3) Nella sezione **Aggiunta di un archivio al repository**, inserire le seguenti informazioni nelle caselle di testo di immissione.

Tabella 16. Archivio a repository

casella Text (Testo)	**Valore**
File	wxs_home/lib/ogclient.jar
Group (Gruppo)	com.ibm.websphere.xs
Artifact (Risorsa)	ogclient
Version (Versione)	7.0
Type (Tipo)	jar

      4) Fare clic sul pulsante di installazione.

         Suggerimento: Consultare le seguenti note tecniche per i dettagli sui differenti metodi di configurazione delle dipendenze di classi e librerie: Specifying external dependencies to applications running on WebSphere Application Server Community Edition.
   b. Distribuire il modulo del servizio dati REST, cioè il file `wxsrestservice.war`, nel server WebSphere Application Server Community Edition.
      1) Modificare il file di esempio XML di distribuzione `restservice_home/gettingstarted/wasce/geronimo-web.xml` in modo che includa le dipendenze di percorso alle directory classpath dell'esempio iniziale:

         Modificare i percorsi classesDirs per i due GBeans del client di avvio.
         - Il percorso "classesDirs" per il GBean GettingStarted_Client_SharedLib deve essere impostato su: `restservice_home/gettingstarted/restclient/bin`
         - Il percorso "classesDirs" per il GBean GettingStarted_Common_SharedLib deve essere impostato su: `restservice_home/gettingstarted/common/bin`
      2) Aprire la console di gestione WebSphere Application Server Community Edition ed eseguire l'accesso.

         Suggerimento: L'URL predefinito è: `http://localhost:8080/console`. L'ID utente predefinito è `system` e la password è `manager`.
      3) Fare clic su **Distribuisci nuovo**.
      4) Nella pagina **Installa nuova applicazione**, immettere i seguenti valori nelle caselle di testo:

Tabella 17. Valori di installazione

casella Text (Testo)	**Valore**
Archive (Archivio)	`restservice_home/lib/wxsrestservice.war`
Plan (Piano)	`restservice_home/gettingstarted/wasce/geronimo-web.xml`

5) Fare clic sul pulsante di installazione.

   La pagina della console dovrebbe indicare che l'applicazione è stata correttamente installata e avviata.

6) Esaminare il log di output di sistema o la console di WebSphere Application Server Community Edition per verificare che il servizio dati REST sia stato avviato correttamente, verificando che sia presente il seguente messaggio:

   ```
   CWOBJ4000I: The WebSphere eXtreme Scale REST data service has
   been started (Il servizio dati REST di WebSphere eXtreme Scale è
   stato avviato).
   ```

5. Verificare che il servizio dati REST stia funzionando:
   a. Aprire il seguente link con un browser: `http://localhost:8080/wxsrestservice/restservice/NorthwindGrid`. Viene visualizzato il documento di servizio NorthwindGrid.
   b. Aprire il seguente link con un browser: `http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/$metadata`. Viene visualizzato il documento EDMX (Entity Model Data Extensions).
6. Per arrestare i processi della griglia, utilizzare `CTRL+C` nella rispettiva finestra comandi per arrestare il processo.
7. Per arrestare WebSphere Application Server Community Edition, utilizzare il seguente comando:
   - UNIX Linux `wasce_root/bin/shutdown.sh`
   - Windows `wasce_root\bin\shutdown.bat`

   Suggerimento: L'ID utente predefinito è `system` e la password è `manager`.Se si utilizza una porta personalizzata, usare l'opzione `-port`.

Avvio dei servizi dati REST su Apache Tomcat

Questo argomento descrive come configurare ed avviare il servizio dati REST di eXtreme Scale utilizzando Apache Tomcat, versione 5.5 o successive.

Prima di iniziare

Verificare che la griglia eXtreme Scale di esempio sia stata avviata. Consultare “Abilitazione del servizio dati REST” a pagina 221 per i dettagli su come avviare la griglia.

Procedura

1. Scaricare ed installare Apache Tomcat Versione 5.5 ocessiva su tomcat_root. Ad esempio: `/opt/tomcat`
2. Installare il servizio dati REST di eXtreme Scale e l'esempio fornito sul server Tomcat nel modo seguente:

a. Se si sta utilizzando Sun JRE o JDK, installare IBM ORB in Tomcat:
   - Per Tomcat versione 5.5

     Copiare tutti i file JAR da:

     `wxs_home/lib/endorsed`

     a

     `tomcat_root/common/endorsed`
   - Per Tomcat versione 6.0
     1) Creare una directory "approvata":
        - UNIX Linux `mkdir tomcat_root/endorsed`
        - Windows `md tomcat_root/endorsed`
     2) Copiare tutti i file JAR da:

        `wxs_home/lib/endorsed`

        a

        `tomcat_root/endorsed`

b. Distribuire il modulo del servizio dati REST: wxsrestservice.war sul server Tomcat.

   Copiare il file wxsrestservice.war da:

   `restservice_home/lib`

   a:

   `tomcat_root/webapps`

c. Aggiungere il JAR runtime del client ObjectGrid e l'applicazione JAR al classpath condiviso in Tomcat:
   1) Modificare il file `tomcat_root/conf/catalina.properties`
   2) Aggiungere i seguenti nomi di percorso alla fine della proprietà shared.loader nel formato di un elenco delimitato da virgole:
      - wxs_home/lib/ogclient.jar
      - restservice_home/gettingstarted/restclient/bin
      - restservice_home/gettingstarted/common/bin

      Importante: Il separatore del percorso deve essere una **barra** (/).

3. Se la griglia eXtreme Scale è stata avviata con la sicurezza eXtreme Scale abilitata, impostare le seguenti proprietà nel file `restservice_home/gettingstarted/restclient/bin/wxsRestService.properties`.

```
ogClientPropertyFile=restservice_home/gettingstarted/security/security.ogclient.properties
loginType=none
```

4. Avviare il server Tomcat con il servizio dati REST:
   - Se si utilizza Tomcat 5.5 su UNIX® o su Windows®, o Tomcat 6.0 su UNIX:
     a. `cd tomcat_root/bin`
     b. Avviare il server:
        - UNIX Linux `./catalina.sh run`
        - Windows `catalina.bat run`
     c. La console visualizza i file di registrazione di Apache Tomcat. Una volta avviato correttamente il servizio dati REST, viene visualizzato il seguente messaggio nella console di gestione:

        ```
        CWOBJ4000I: The WebSphere eXtreme Scale REST data service has been
        started (Il servizio dati REST di WebSphere eXtreme Scale è stato
        avviato).
        ```

- Se si utilizza Tomcat 6.0 su Windows:
  a. `cd tomcat_root/bin`
  b. Avviare lo strumento di configurazione Apache Tomcat 6 con il seguente comando: `tomcat6w.exe`
  c. Fare clic sul pulsante Start nella finestra delle proprietà di Apache Tomcat 6 per avviare il server Tomcat.
  d. Rivedere i seguenti file di registrazione per verificare che il server Tomcat sia stato avviato correttamente.
     – `tomcat_root/bin/catalina.log`

       Visualizza lo stato del motore del server di Tomcat
     – `tomcat_root/bin/stdout.log`

       Visualizza il file di registrazione dell'output di sistema.
  e. Una volta avviato correttamente il servizio dati REST, viene visualizzato il seguente messaggio nel file di registrazione dell'output di sistema: `CWOBJ4000I: (Il servizio dati REST di WebSphere eXtreme Scale è stato avviato).`

5. Verificare che il servizio dati REST stia funzionando:
   a. Aprire un browser Web e navigare su:

      http://localhost:8080/wxsrestservice/restservice/NorthwindGrid

      Viene visualizzato il documento di servizio NorthwindGrid.
   b. Navigare su:

      http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/$metadata

      Viene visualizzato il documento EDMX (Entity Model Data Extensions).
6. Per arrestare i processi della griglia, utilizzare CRTL+C nella relativa finestra comandi.
7. Per arrestare Tomcat, utilizzare CTRL + C nella finestra in cui è stato avviato.

Utilizzo di un browser con i servizi dati REST

Il servizio dati REST di eXtreme Scale crea feed ATOM per impostazione predefinita quando si utilizza un browser Web. Il formato feed ATOM potrebbe non essere compatibile con browser più vecchi o potrebbe essere interpretato così da non poter visualizzare i dati come XML. Nelle seguenti sezioni vengono forniti dettagli sulle modalità di configurazione di Internet Explorer Versione 8 e Firefox Versione 3 per visualizzare i feed ATOM e XML nel browser.

Informazioni su questa attività

Il servizio dati REST di eXtreme Scale crea feed ATOM per impostazione predefinita quando si utilizza un browser Web. Il formato feed ATOM potrebbe non essere compatibile con browser più vecchi o potrebbe essere interpretato così da non poter visualizzare i dati come XML. Per i browser più vecchi verrà richiesto di salvare i file su disco. Una volta scaricati i file, utilizzare il lettore XML preferito per visualizzare i file. L'XML generato non viene formattato per essere visualizzato, pertanto verrà tutto visualizzato su un'unica riga. La maggior parte dei programmi di lettura XML, come, ad esempio, Eclipse, supportano la riformattazione di XML in un formato leggibile.

Per browser più recenti, come Microsoft Internet Explorer Versione 8 e Firefox Versione 3, i file XML ATOM possono essere visualizzati localmente nel browser. Nelle seguenti sezioni vengono forniti dettagli sulle modalità di configurazione di

Internet Explorer Versione 8 e Firefox Versione 3 per visualizzare i feed ATOM e XML nel browser.

Procedura

Configurazione di Internet Explorer Versione 8

- Per abilitare Internet Explorer alla lettura di feed ATOM generati dal servizio dati REST utilizzare la seguente procedura:
  1. Fare clic in **Strumenti** → **Opzioni Internet**
  2. Selezionare la scheda **Contenuto**
  3. Fare clic sul pulsante **Impostazioni** nella sezione **Feed e Web Slice**
  4. Annullare la selezione della casella: "Attiva visualizzazione di lettura feed"
  5. Fare clic su **OK** per ritornare al browser.
  6. Riavviare Internet Explorer.

Configurazione di Firefox Versione 3

- Firefox non visualizza automaticamente le pagine con il tipo di contenuto: application/atom+xml. La prima volta in cui si visualizza una pagina, in Firefox viene richiesto di salvare il file. Per visualizzare la pagina, aprire il file con Firefox nel modo seguente:
  1. Dalla finestra di dialogo per la selezione delle applicazioni, selezionare il pulsante di scelta "Apri con" e fare clic sul pulsante **Sfoglia**.
  2. Passare alla directory di installazione Firefox. Ad esempio: `C:\Program Files\Mozilla Firefox`
  3. Selezionare `firefox.exe` e premere il pulsante **OK**.
  4. Selezionare la casella di spunta simile alla seguente “Eseguire automaticamente per file come questo...”.
  5. Fare clic sul pulsante **OK**.
  6. Firefox visualizza successivamente la pagina XML ATOM in una nuova scheda o finestra del browser.
- Firefox visualizza automaticamente i feed ATOM in un formato leggibile. Tuttavia, i feed creati dal servizio dati REST includono gli XML. Firefox non può visualizzare XML a meno che non venga disabilitato il renderer dei feed. Diversamente da Internet Explorer, in Firefox il plug-in di rendering dei feed ATOM deve essere modificato esplicitamente. Per configurare Firefox in modo da leggere i feed ATOM come file XML, completare la seguente procedura:
  1. Aprire il seguente file in un editor di testo: `<firefoxInstallRoot>\components\FeedConverter.js`. Nel percorso `<firefoxInstallRoot>` è la directory principale dove è installato Firefox.

     Per i sistemi operativi Windows la directory predefinita è: `C:\Program Files\Mozilla Firefox`.
  2. Ricercare il frammento simile al seguente:

     ```
     // show the feed page if it wasn't sniffed and we have a document,
     // or we have a document, title, and link or id
     if (result.doc && (!this._sniffed ||
         (result.doc.title && (result.doc.link || result.doc.id)))) {
     ```
  3. Impostare come commento le due righe che iniziano con `if` e `result` inserendo due `//` (due barre) all'inizio di queste righe.
  4. Aggiungere la seguente istruzione al frammento: `if(0) {`.
  5. Il testo risultante deve essere simile al seguente:

```
// show the feed page if it wasn't sniffed and we have a document,
// or we have a document, title, and link or id
//if (result.doc && (!this._sniffed ||
//    (result.doc.title && (result.doc.link || result.doc.id)))) {
if(0) {
```

6. Salvare il file.
7. Riavviare Firefox
8. Ora Firefox è in grado di visualizzare automaticamente tutti i feed nel browser.

- Verificare l'impostazione provando degli URL.

Esempio

In questa sezione vengono descritti alcuni URL di esempio che possono essere utilizzati per visualizzare i dati aggiunti mediante l'esempio iniziale fornito con il servizio dati REST di eXtreme Scale. Prima di utilizzare i seguenti URL, aggiungere le serie di dati predefinite alla griglia di esempio di eXtreme Scale utilizzando il client Java di esempio o il client Visual Studio WCF Data Services di esempio.

Nei seguenti esempi si presuppone che la porta sia 8080, ma può variare. Per dettagli sulla modalità di configurazione del servizio dati su server delle applicazioni differenti, consultare la sezione.

- Visualizzare un cliente singolo con l'ID "ACME":

  ```
  http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/Customer('ACME')
  ```

- Visualizzare tutti gli ordini per il cliente "ACME":

  ```
  http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/Customer('ACME')/orders
  ```

- Visualizzare il cliente "ACME" e gli ordini:

  ```
  http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/Customer('ACME')?$expand=orders
  ```

- Visualizzare l'ordine 1000 per il cliente "ACME":

  ```
  http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/Order(orderId=1000,customer_customerId='ACME')
  ```

- Visualizzare l'ordine 1000 per il cliente "ACME" ed i Customer associati:

  ```
  http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/
  Order(orderId=1000,customer_customerId='ACME')?$expand=customer
  ```

- Visualizzare l'ordine 1000 per il cliente "ACME" ed i Customer e gli OrderDetails associati:

  ```
  http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/
  Order(orderId=1000,customer_customerId='ACME')?$expand=customer,orderDetails
  ```

- Visualizzare tutti gli ordini per il cliente "ACME" per il mese di ottobre 2009 (GMT):

  ```
  http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/
  Customer(customerId='ACME')/orders?$filter=orderDate
  ge datetime'2009-10-01T00:00:00'
  and orderDate lt datetime'2009-11-01T00:00:00'
  ```

- Visualizzare tutti i primi 3 ordini e orderDetails per il cliente "ACME" per il mese di ottobre 2009 (GMT):

  ```
  http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/
  Customer(customerId='ACME')/orders?$filter=orderDate
  ge datetime'2009-10-01T00:00:00'
  and orderDate lt datetime'2009-11-01T00:00:00'
  &$orderby=orderDate&$top=3&$expand=orderDetails
  ```

Utilizzo di un client Java con i servizi dati REST

L'applicazione Java client utilizza l'API eXtreme Scale EntityManager per inserire dati nella griglia.

Informazioni su questa attività

Le sezioni precedenti descrivevano come creare una griglia eXtreme Scale e come configurare ed avviare il servizio dati REST di eXtreme Scale. L'applicazione Java client utilizza l'API eXtreme Scale EntityManager per inserire dati nella griglia. Non mostra come utilizzare le interfacce REST. Lo scopo di questo client è quello di dimostrare il modo in cui l'API EntityManager viene utilizzata per interagire con la griglia eXtreme Scale e consentire la modifica dei dati nella griglia. Per visualizzare i dati nella griglia utilizzando il servizio dati REST, usare un browser Web o utilizzare l'applicazione client Visual Studio 2008.

Procedura

Per aggiungere velocemente contenuto alla griglia eXtreme Scale, eseguire i comandi di seguito riportati:

1. Aprire una riga comandi o una finestra di terminale ed impostare la variabile di ambiente JAVA_HOME:
   - Linux UNIX `export JAVA_HOME=java_home`
   - Windows `set JAVA_HOME=java_home`
2. `cd restservice_home/gettingstarted`
3. Inserire nella griglia alcuni dati. I dati inseriti verranno richiamati tramite un browser Web ed il servizio dati REST.

   Se la griglia è stata avviata *senza* la sicurezza eXtreme Scale, utilizzare i seguenti comandi.
   - UNIX Linux `./runclient.sh load default`
   - Windows `runclient.bat load default`

   Se la griglia è stata avviata *con* la sicurezza eXtreme Scale, utilizzare i seguenti comandi.
   - UNIX Linux `./runclient_secure.sh load default`
   - Windows `runclient_secure.bat load default`

   Per un client Java, utilizzare la seguente sintassi di comando:
   - UNIX Linux `runclient.sh command`
   - Windows `runclient.bat command`

   Sono disponibili i seguenti comandi:
   - `load default`

     Carica nella griglia una serie predefinita di entità Customer, Category e Product e crea una serie casuale di ordini per ogni cliente.
   - `load category categoryId categoryName firstProductId num_products`

     Crea nella griglia una categoria del prodotto ed un numero fisso di entità Product. Il parametro firstProductId identifica il numero id del primo prodotto e ad ogni prodotto successivo viene assegnato l'id successivo fino a che non viene creato il numero specificato di prodotti.
   - `load customer companyCode contactNamecompanyName numOrders firstOrderIdshipCity maxItems discountPct`

     Carica un nuovo cliente nella griglia e crea una serie fissa di entità Order per qualunque prodotto casuale caricato al momento nella griglia. Il numero di ordini è determinato dall'impostazione del parametro <numOrders>. Ogni ordine avrà un numero casuale di entità OrderDetail fino a <maxItems>

- `display customer companyCode`
  Visualizza un'entità Customer, l'Order associato e l'entità OrderDetail.
- `display category categoryId`
  Visualizza un'entità Category del prodotto e le entità Product associate.

Risultati

- `runclient.bat load default`
- `runclient.bat load customer IBM "John Doe" "IBM Corporation" 5 5000 Rochester 5 0.05`
- `runclient.bat load category 5 "Household Items" 100 5`
- `runclient.bat display customer IBM`
- `runclient.bat display category 5`

Esecuzione e build della griglia di esempio e di Java client con Eclipse

L'esempio iniziale del servizio dati REST può essere aggiornato e migliorato utilizzando Eclipse. Per i dettagli su come impostare il proprio ambiente Eclipse, consultare il documento di testo: `restservice_home/gettingstarted/ECLIPSE_README.txt`.

Una volta importato il progetto WXSRestGettingStarted in Eclipse ed eseguito correttamente il build, l'esempio verrà automaticamente ricompilato e i file script verranno utilizzati per avviare il server contenitore e il client rileverà automaticamente i file di classe ed i file XML. Il servizio dati REST rileverà automaticamente anche eventuali modifiche poiché il server Web è configurato per leggere automaticamente le directory del build Eclipse.

Importante: Quando si modificano i file di configurazione o di origine, deve essere riavviato sia il server contenitore di eXtreme Scale sia l'applicazione del servizio dati REST. Il server contenitore di eXtreme Scale deve essere avviato prima dell'applicazione Web del servizio dati REST.

Visual Studio 2008 WCF client con servizio dati REST

L'esempio di inizio del servizio dati REST di eXtreme Scale include un client di WCF Data Services che può interagire con il servizio dati REST di eXtreme Scale. L'esempio viene scritto come un'applicazione da riga comandi in C#.

Requisiti software

Il client C# WCF Data Services di esempio richiede quanto segue:

- Sistema operativo
  - Microsoft Windows XP
  - Microsoft Windows Server 2003
  - Microsoft Windows Server 2008
  - Microsoft Windows Vista
- Microsoft Visual Studio 2008 con Service Pack 1

  Suggerimento: Consultare il precedente collegamento per conoscere i requisiti hardware e software aggiuntivi.
- Microsoft .NET Framework 3.5 Service Pack 1

- Supporto Microsoft: è disponibile un aggiornamento per .NET Framework 3.5 Service Pack 1

Creazione ed esecuzione del client di avvio

Il client di esempio di WCF Data Services include un progetto e una soluzione Visual Studio 2008 e il codice sorgente per l'esecuzione dell'esempio. L'esempio deve essere caricato in Visual Studio 2008 e compilato in un programma eseguibile Windows prima che possa essere eseguito. Per costruire ed eseguire l'esempio, consultare il documento di testo: `restservice_home/gettingstarted/VS2008_README.txt`.

Sintassi del comando client C# WCF Data Services

Windows `WXSRESTGettingStarted.exe <service URL> <command>`

<service URL> è l'URL del servizio dati REST di eXtreme Scale configurato nella sezione.

Sono disponibili i seguenti comandi:

- `load default`

  Carica una serie predefinita di entità Customer, Category e Product in una griglia e crea una serie casuale di ordini per ciascun cliente.
- `load category <categoryId> <categoryName> <firstProductId> <numProducts>`

  Crea una categoria del prodotto ed un numero fisso di entità Product nella griglia. Il parametro firstProductId identifica il numero id del primo prodotto e ad ogni prodotto successivo viene assegnato l'id successivo fino a che non viene creato il numero specificato di prodotti.
- `load customer <companyCode> <contactName> <companyName> <numOrders> <firstOrderId> <shipCity> <maxItems> <discountPct>`

  Carica un nuovo cliente nella griglia e crea una serie fissa di entità Order per qualunque prodotto casuale caricato al momento nella griglia. Il numero di ordini è determinato impostando il parametro <numOrders>. Ogni ordine avrà un numero casuale di entità OrderDetail fino a <maxItems>
- `display customer <companyCode>`

  Visualizza un'entità Customer, l'Order associato e l'entità OrderDetail.
- `display category <categoryId>`

  Visualizza un'entità Category del prodotto e le entità Product associate.
- `unload`

  Rimuove tutte le entità che erano state caricate utilizzando il comando "default load".

Gli esempi seguenti illustrano vari comandi.

- `WXSRestGettingStarted.exe http://localhost:8080/wxsrestservice/restservice/NorthwindGrid load default`
- `WXSRestGettingStarted.exe http://localhost:8080/wxsrestservice/restservice/NorthwindGrid load customer`
- `IBM "John Doe" "IBM Corporation" 5 5000 Rochester 5 0.05`
- `WXSRestGettingStarted.exe http://localhost:8080/wxsrestservice/restservice/NorthwindGrid load category 5 "Household Items" 100 5`

- `WXSRestGettingStarted.exe http://localhost:8080/wxsrestservice/restservice/NorthwindGrid display customer IBM`
- `WXSRestGettingStarted.exe http://localhost:8080/wxsrestservice/restservice/NorthwindGrid display category 5`

Informazioni particolari

Riferimenti in questa documentazione a prodotti, programmi o servizi IBM non implica che IBM intenda renderli disponibili in tutti i paesi in cui IBM opera. Qualsiasi riferimento ad un prodotto, programma o servizio IBM non implica o non intende dichiarare che possa essere utilizzato solo quel prodotto, programma o servizio IBM. In sostituzione a quelli forniti da IBM possono essere utilizzati prodotti, programmi o servizi funzionalmente equivalenti che non comportino la violazione dei diritti di proprietà intellettuale IBM. È responsabilità dell'utente valutare e verificare il funzionamento con altri prodotti, ad eccezione di quelli progettati espressamente da IBM.

IBM può avere brevetti o domande di brevetto in corso relativi a quanto trattato nel presente documento. Il rilascio di questo documento non fornisce alcuna licenza a questi brevetti. È possibile inviare per iscritto richieste di licenze a:

IBM Director of Commercial Relations
IBM Europe
Schoenaicher Str. 220
D-7030 Boeblingen Deutschland

Coloro che detengono la licenza su questo programma e desiderano avere informazioni su di esso allo scopo di consentire (i) uno scambio di informazioni tra programmi indipendenti ed altri (compreso questo) e (ii) l'uso reciproco di tali informazioni, dovrebbero rivolgersi a:

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

Tali informazioni possono essere disponibili, in base ad appropriate clausole e condizioni, includendo in alcuni casi, il pagamento di un corrispettivo.

Marchi

I seguenti termini sono marchi di IBM Corporation negli Stati Uniti e/o in altri paesi:

- AIX
- CICS
- Cloudscape
- DB2
- Domino
- IBM
- Lotus
- RACF
- Redbooks
- Tivoli
- WebSphere
- z/OS

Java e tutti i marchi basati su Java sono marchi di Sun Microsystems, Inc. negli Stati Uniti e/o in altri paesi.

LINUX è un marchio di Linus Torvalds negli Stati Uniti e/o in altri paesi.

Microsoft, Windows, Windows NT®, e il logo Windows sono marchi di Microsoft Corporation negli Stati Uniti e/o in altri paesi.

UNIX è un marchio registrato di The Open Group negli Stati Uniti e in altri paesi.

Nomi di altri prodotti, società e servizi possono essere marchi di altre società.

Indice analitico

A

API di statistiche 157
architettura 9, 10, 12, 13, 14
autonoma 230
avvio dei server 230

B

bilanciamento del carico 45, 109, 134, 154
blocco
  ottimistico 163
  pessimistico 163
  strategie per 163

C

cache 1, 5, 8, 9
  locale 15
cache coerente 31
completa 32
contenitore 118
contenitori
  posizionamento per contenitore 93
convalida basata sugli eventi 51

D

database 31, 33
  sincronizzazione 50
  tecniche di sincronizzazione del database 50
disponibilità
  connettività 107
  errore
    contenitore 107
    servizio catalogo 107
  replica
    lato client 45, 109, 134, 154
dominio del servizio catalogo 118

E

Extreme Transaction Processing 1, 5, 8

F

failover
  configurazione 115
  heartbeat e 115
  impostazioni consigliate 115
frammenti
  allocazione 132
  primario 132
  replica 132
frammento 90
  ciclo di vita 134
  errore 134
frammento *(Continua)*
  ripristino 134
funzionamento 5
funzioni obsolete 4

G

gestore di sessione 7
gestore entità 186, 187
  aggiornamento elementi 192, 193
  creazione di una classe di entità 186
  eseguire una query 193
  relazione di entità 187
  supporto didattico 186, 187
  utilizzo di un indice per aggiornare e rimuovere elementi 193
gestore entitàEntityManager
  creazione di uno schema entità Order 189
Gestore sessioni HTTP 67
griglia 90

I

in-line cache 33
indice
  prestazioni 52
  qualità dati 52
integrazione 31
integrazione con altri server 7

J

JPA (Java Persistence API)
  plug-in della cache
    introduzione 63
  topologia della cache
    incorporata 63
    remota 63
    suddivisa in partizioni incorporate 63

L

limitata 32

M

mappa di backup
  strategia di blocco 160
memorizzazione nella cache 32, 33
modifiche di distribuzione
  utilizzo di Java message service 153, 166

N

nuove funzioni 4

P

panoramica eXtreme Scale 1, 7, 8
panoramicain modo programmatico
  utilizzo di un programma di caricamento 40
partizionamento
  con le entità 91
  introduzione 91
partizione 90
partizioni
  posizionamento fisso 93
  transazioni 97, 167
pianificazione
  distribuzione dell'applicazione 7, 8
PMI i
  MBean 157
PMI (Performance Monitoring Infrastructure) 157
posizionamento
  strategie per 93
precaricamento della mappa 45, 109, 134, 154
prestazioni 45, 109, 134, 154
programma di caricamento
  Panoramica su JPA (Java Persistence API) 61

Q

query oggetto
  chiave primaria 195
  indice 196
  schema mappa 195
  supporto didattico 194, 195, 196
query oggettopiù relazioni
  supporto didattico 199
query oggettorelazioni mappa
  supporto didattico 197
quorum
  comportamento del contenitore 121
  xsadmin 121

R

replica
  costo di memoria 129
  programmi di caricamento e 129
  tipi di frammenti 129
repliche
  lettura da 133

S

scalabilità
  con unità o pod 103
  introduzione 89
scenari di memorizzazione nella cache
  read-through 34
  write-through 34

serializzazione
  blocco 58
  prestazione 58
sessioni 67
sicurezza
  autenticazione 175
  autorizzazione 175
  trasporto sicuro 175
side cache 33
Spring
  ambito del frammento 183
  bean di estensione 183
  creazione package 183
  framework 183
  supporto spazio dei nomi 183
  transazioni native 183
  webflow 183
strategia di posizionamento 90
supporto 36
supporto didattico 186, 187
  comunicazione sicura tra
    endpoint 215
  esempio libero 202
supporto didattico sicurezza
  autenticazione client 205
  autorizzazione 212
supporto didattico sicurezzaSSL/TLS
  autorizzazione client 201
  esempio libero 201
  programma di autenticazione
    client 201
supporto per la memorizzazione nella
  cache 36
supporto per la memorizzazione nella
  cacheprogramma di
  caricamentotransazione del programma
  di caricamento 36

T

topologia 9, 10, 12, 13, 14
transazioni
  con le sessioni 157
  panoramica 157, 159
  partizione singola 97, 167
  sulla griglia 97, 167
  vantaggi 157
transazioni su partizione 97, 167

V

vantaggi 36

W

write-behind 36

Stampato in Italia