IBM WebSphere eXtreme Scale Versão 7.1.1
Versão 7 Release 1

Visão Geral do Produto

21 de Novembro de 2011

Esta edição aplica-se à versão 7, release 1, modificação 1 do WebSphere eXtreme Scale e a todos os releases e modificações subsequentes até que seja indicado de outra forma em novas edições.

Índice

Figuras

Tabelas

Sobre o *Visão Geral do Produto*

O conjunto da documentação do WebSphere eXtreme Scale inclui três volumes que fornecem as informações necessárias para utilizar, programar e administrar o produto WebSphere eXtreme Scale.

Biblioteca do WebSphere eXtreme Scale

A biblioteca do WebSphere eXtreme Scale contém os seguintes livros:

- O *Visão Geral do Produto* contém um visualização de alto nível dos conceitos do WebSphere eXtreme Scale, incluindo cenários de caso de uso e tutoriais.
- O *Guia de Instalação* descreve como instalar topologias comuns do WebSphere eXtreme Scale.
- O *Guia de Administração* contém as informações necessárias para os administradores de sistema, incluindo como planejar implementações do aplicativo, planejar capacidade, instalar e configurar o produto, iniciar e parar servidores, monitorar o ambiente e proteger o ambiente.
- O *Guia de Programação* contém informações para desenvolvedores de aplicativos sobre como desenvolver aplicativos para o WebSphere eXtreme Scale utilizando as informações da API incluídas.

Para fazer download dos manuais, vá para a Página da Biblioteca do WebSphere eXtreme Scale.

Também é possível acessar as mesmas informações nesta biblioteca no Centro de Informações do WebSphere eXtreme Scale Versão 7.1.1.

Usando Manuais Off-line

Todos os manuais na biblioteca do WebSphere eXtreme Scale contêm links para o centro de informações com a seguinte URL raiz: `http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r1m1`. Esses links levam diretamente para as informações relacionadas. No entanto, se estiver trabalhando off-line e encontrar um desses links, será possível procurar pelo título do link nos outros manuais na biblioteca. A documentação da API, o glossário e a referência de mensagens não estão disponíveis em manuais PDF.

Quem Deve Utilizar este Manual

Este manual é destinado a qualquer pessoa que esteja interessada em aprender sobre o WebSphere eXtreme Scale.

Obtendo Atualizações para este Manual

É possível obter as atualizações para esse manual ao fazer download da versão mais recente da Página da Biblioteca do WebSphere eXtreme Scale.

Como Enviar Seus Comentários

Entre em contato com a equipe de documentação. Você localizou o que precisava? O conteúdo era exato e completo? Envie seus comentários sobre esta documentação por e-mail para wasdoc@us.ibm.com.

Capítulo 1. Visão Geral do Produto

WebSphere eXtreme Scale é uma grade de dados na memória, elástica e escalável. A grade de dados dinamicamente armazena em cache, particiona, replica e gerencia dados do aplicativo e a lógica de negócios em diversos servidores. O WebSphere eXtreme Scale executa grandes volumes de processamento de transações com alta eficiência e escalabilidade linear. Com WebSphere eXtreme Scale, você também pode obter qualidades de serviço como integridade transacional, alta disponibilidade e tempos de respostas previsíveis.

WebSphere eXtreme ScaleVisão Geral

WebSphere eXtreme Scale é uma grade de dados na memória, elástica e escalável. A grade de dados dinamicamente armazena em cache, particiona, replica e gerencia dados do aplicativo e a lógica de negócios em diversos servidores. O WebSphere eXtreme Scale executa grandes volumes de processamento de transações com alta eficiência e escalabilidade linear. Com WebSphere eXtreme Scale, você também pode obter qualidades de serviço como integridade transacional, alta disponibilidade e tempos de respostas previsíveis.

O WebSphere eXtreme Scale pode ser usado de diferentes formas. É possível usar o produto como um cache muito poderoso, como um espaço de processamento de banco de dados em memória para gerenciar o estado do aplicativo ou para construir aplicativos do Extreme Transaction Processing (XTP). Essas capacidades do XTP incluem uma infraestrutura de aplicativo para suportar seus aplicativos de negócios críticos mais exigentes.

Escalabilidade Elástica

A escalabilidade elástica é possível por meio do uso do armazenamento em cache de objeto distribuído. Com escalabilidade elástica, a grade de dados monitora e gerencia a si mesma. A grade de dados pode incluir ou remover servidores da topologia, que aumenta ou diminui a memória, o rendimento de rede e a capacidade de processamento conforme necessário. Quando um processo de scale-out é iniciado, a capacidade é incluída na grade de dados enquanto ela está em execução sem requerer um reinício. De modo inverso, um processo de scale-in imediatamente remove a capacidade. A grade de dados também é auto-recuperável recuperando-se automaticamente de falhas.

WebSphere eXtreme Scale Versus um Banco de Dados em Memória

O WebSphere eXtreme Scale não pode ser considerado um banco de dados em memória real. Um banco de dados em memória é muito simples para manipular algumas das complexidades que o WebSphere eXtreme Scale pode gerenciar. Se um banco de dados em memória tiver um servidor que falha, ele não poderá reparar o problema. Uma falha poderá ser desastrosa se o seu ambiente inteiro estiver em um servidor.

Para resolver o problema desse tipo de falha, o eXtreme Scale divide o determinado conjunto de dados em partições, que são equivalentes aos esquemas em árvore restritos. Os esquemas em árvore restritos descrevem o relacionamento entre entidades. Quando estiver usando partições, os relacionamentos da entidade

devem modelar uma estrutura de dados em árvore. Nesta estrutura, o cabeça da árvore é a entidade raiz e é a única entidade particionada. Todos os outros filhos da entidade raiz são armazenados na mesma partição que a entidade raiz. Cada partição existe como uma cópia primária ou shard. Uma partição também contém shards de réplica para fazer backup de dados. Um banco de dados em memória não pode fornecer esta função porque ele não é estruturado e dinâmico dessa maneira. Com um banco de dados em memória, você deve implementar as operações que o WebSphere eXtreme Scale faz automaticamente. É possível executar operações SQL em bancos de dados em memória, aumentando a velocidade de processamento comparado com os bancos de dados que não são de memória. O WebSphere eXtreme Scale possui sua própria linguagem de consulta em vez de suporte SQL. Esta linguagem de consulta é mais elástica, permite o particionamento de dados e fornece recuperação de falha confiável.

WebSphere eXtreme Scale com Bancos de Dados

Com o recurso de cache write-behind, o WebSphere eXtreme Scale pode servir de cache front-end para um banco de dados. Com o uso desse cache front-end, o rendimento aumenta enquanto reduz a contenção e o carregamento do banco de dados. WebSphere eXtreme Scale fornece ampliação e adição previsíveis a um custo de processamento previsível.

A imagem a seguir mostra que em um ambiente de cache distribuído e coerente, os clientes eXtreme Scale enviam e recebem dados da grade de dados. A grade de dados pode ser automaticamente sincronizada com um armazenamento de dados de backend. O cache é coerente porque todos os clientes veem os mesmos dados no cache. Cada parte dos dados é armazenada exatamente em um servidor gravável no cache. Ter uma cópia de cada parte dos dados evita cópias de registros excessivas que podem conter diferentes versões dos dados. Um cache coerente contém mais dados à medida em que mais servidores são incluídos na grade de dados e escala linearmente conforme a grade de dados cresce em tamanho. Os dados também podem ser opcionalmente replicados para se obter tolerância a falhas opcional.

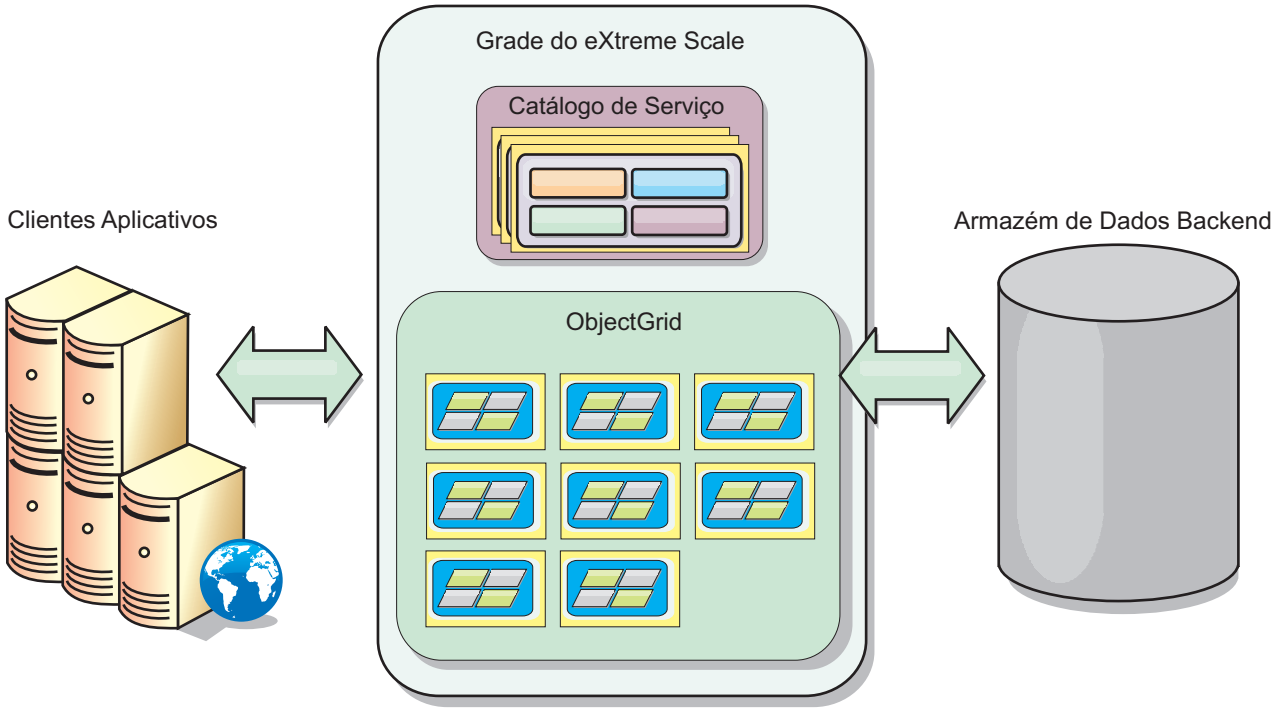


Figura 1. Topologia de Alto Nível

O WebSphere eXtreme Scale possui servidores, chamados de *servidores de contêiner*, que oferecem sua grade de dados em memória. Esses servidores podem executar dentro do WebSphere Application Server ou em um Java Standard Edition (J2SE) Java virtual machines simples. Mais de um servidor de contêiner pode ser executado em um único servidor físico. Como resultado, a grade de dados em memória pode ser grande. A grade de dados não é limitada pela memória ou pelo espaço de endereço do aplicativo ou do servidor de aplicativos e também não tem um impacto sobre eles. A memória pode ser a soma da memória de centenas ou milhares de Java virtual machines executados em vários servidores físicos diferentes.

Assim como no espaço de processamento do banco de dados em memória, o WebSphere eXtreme Scale pode ser suportado por disco, banco de dados ou ambos.

Enquanto o eXtreme Scale fornece diversas APIs Java, muitos casos de uso não necessitam de programação do usuário, apenas a configuração e implementação de sua infraestrutura do WebSphere.

Visão Geral da Grade de Dados

A interface de programação mais simples do eXtreme Scale é a interface ObjectMap, que é uma interface de mapa simples que inclui: um método map.put(key,value) para colocar um valor no cache e um método map.get(key) posteriormente para recuperar o valor.

O paradigma fundamental da grade de dados é um par de chave-valor, no qual a grade de dados armazena valores (objetos Java), com uma chave associada (outro objeto Java). A chave é utilizada posteriormente para recuperar o valor. No eXtreme Scale, um mapa consiste em entradas desses pares de chave-valor.

O WebSphere eXtreme Scale oferece várias configurações da grade de dados, de um cache local único e simples até um cache grande distribuído, usando várias Java virtual machines ou servidores.

Além de armazenar objetos Java simples, é possível armazenar objetos com relacionamentos. É possível utilizar uma linguagem de consulta como SQL, com instruções `SELECT ... FROM ... WHERE` para recuperar esses objetos. Por exemplo, um objeto de ordem pode ter um objeto de cliente e vários objetos de item associados a ele. WebSphere eXtreme Scale suporta relacionamentos um-para-um, um-para-muitos, muitos-para-um e muitos-para-muitos.

WebSphere eXtreme Scale também suporta uma interface de programação EntityManager para armazenar entidades no cache. Essa interface de programação é semelhante às entidades do Java Enterprise Edition. Os relacionamentos de entidades podem ser automaticamente descobertos a partir de um arquivo XML do descritor de entidade ou de anotações nas classes Java . É possível recuperar uma entidade do cache pela chave primária usando o método find na interface EntityManager. As entidades podem ser persistidas para a grade de dados ou removidas dela dentro de um limite de transação.

Considere um exemplo distribuído em que a chave é um nome alfabético simples. O cache pode ser dividido em quatro partições por chave: a partição 1 para chaves que começam com A-E, partição 2 para chaves que começam com F-L, e assim por diante. Para disponibilidade, uma partição possui um shard primário e um shard de réplica. As mudanças nos dados do cache são feitas no shard primário e replicadas para o shard de réplica. Configure um número de servidores que contêm os dados da grade de dados e o eXtreme Scale distribui os shards nos dados sobre essas instâncias do servidor. Para disponibilidade, os shards de réplica são colocados em servidores físicos separados dos shards primários.

O WebSphere eXtreme Scale usa um serviço de catálogo para localizar o shard primário para cada chave. Ele manipula a movimentação dos shards entre servidores do eXtreme Scale quando os servidores físicos falham e se recuperam posteriormente. Por exemplo, se o servidor contendo um shard de réplica falhar, o eXtreme Scale alocará um novo shard de réplica. Se um servidor que contém um shard primário falhar, o shard de réplica será promovido como shard primário. Como antes, um novo shard de réplica é construído.

O Que Há de Novo na Versão 7.1.1

O WebSphere eXtreme Scale inclui muitos novos recursos na Versão 7.1.1. Use este tópico para aprender sobre as atualizações do produto mais recentes.

Plug-ins DataSerializer

Quando clientes e servidores trocam informações ou quando servidores replicam dados de um servidor para um outro, os dados devem ser convertidos, ou serializados, para que possam ser transmitidos pela rede. Em liberações anteriores, usava-se a serialização Java ou o plug-in ObjectTransformer padrão para serializar os dados. Nesta liberação, os plug-ins DataSerializer podem ser usados para descrever de modo eficiente seu formato de serialização, ou uma matriz de bytes, para o WebSphere eXtreme Scale, de modo que o produto possa interagir com a matriz de bytes sem requerer um formato de objeto específico. Saiba mais...

Estrutura OSGi

Usando a estrutura OSGi, é possível expor seus plug-ins como serviços OSGi para que eles possam ser usados pelo tempo de execução do eXtreme Scale. Além disso, os servidores e clientes do eXtreme Scale podem ser iniciados em um contêiner OSGi, para poder incluir e atualizar dinamicamente plug-ins do eXtreme Scale no ambiente de tempo de execução. Saiba mais...

Aprimoramento de Desempenho do Provedor de Cache Dinâmico

O processamento de invalidação dentro do provedor de cache dinâmico do WebSphere eXtreme Scale foi melhorado. Solicitações de invalidações são processadas de modo assíncrono e em lote quando o parâmetro **wait** do método invalidate(key, wait) é definido para um valor de `false`. Esse aprimoramento melhora significativamente o desempenho. Saiba mais...

Mudança no Comportamento de Posicionamento Padrão

Em liberações anteriores, quando um novo servidor de contêiner era iniciado na grade de dados, o posicionamento dos shards nesse servidor de contêiner iniciava imediatamente. Este posicionamento imediato resultava em uma alta utilização do processador nos servidores que continham os novos servidores de contêiner. O comportamento padrão foi alterado para configurar um atraso de 15000 ms, ou 15 segundos, antes de o posicionamento ocorrer. O intervalo de posicionamento pode ser alterado com a propriedade do servidor **placementDeferralInterval**. Saiba mais...

Topologia de Intradomínio para as Configurações de Plug-in de Cache do Java Persistence API (JPA) Nível 2 (L2)

Ao configurar uma topologia de intradomínio em seu cache JPA L2, um shard primário é colocado em cada servidor de contêiner na configuração. Cada shard primário possui o conteúdo inteiro da partição. Ao usar essa configuração, é possível aumentar o desempenho porque os clientes podem acessar dados localmente e qualquer um dos shards primários pode ser gravados na grade de dados. Saiba mais...

Utilitário `xscmd`

O utilitário **xscmd** é a nova versão suportada do utilitário **xsadmin**. O utilitário **xsadmin** era incluído como uma amostra não suportada em liberações anteriores. Saiba mais...

Ferramenta para Gerar Relatórios de Análise de Log

Com a nova ferramenta **xsloganalyzer**, é possível gerar relatórios a partir dos arquivos de log que ajudam a analisar o desempenho de seu ambiente e a resolver problemas. Saiba mais...

IBM eXtremeIO e IBM eXtremeMemory

Ao configurar o eXtremeMemory, objetos podem ser armazenados na memória nativa em vez de serem armazenados no heap Java. Configurar o eXtremeMemory ativa o eXtremeIO como um novo mecanismo de transporte. Movendo objetos para fora do heap Java, é possível evitar pausas de coleta de lixo, levando a um desempenho mais constante e a tempos de resposta atribuíveis. Saiba mais...

Suporte ao WebSphere Application Server Versão 8

O WebSphere eXtreme Scale Versão 7.1.1 agora pode ser instalado no WebSphere Application Server e WebSphere Application Server Network Deployment Versão 8. Saiba mais...

Notas sobre o Release

São fornecidos links para o website de suporte do produto, para documentação do produto e para atualizações de última hora, limitações e problemas conhecidos para o produto.

- "Acessando Últimas Atualizações, Limitações e Problemas Conhecidos"
- "Acessando Requisitos de Software e de Sistema"
- "Acessando a Documentação do Produto"
- "Acessando o Website de Suporte do Produto"
- "Entrando em Contato com o Suporte ao Software IBM"

Acessando Últimas Atualizações, Limitações e Problemas Conhecidos

As notas sobre o release estão disponíveis como notas técnicas no site de suporte do produto. Para ver uma lista de todas as notas técnicas do WebSphere eXtreme Scale, vá para a Página da Web de Suporte. Clicar nos links fornecidos aqui resultará em uma pesquisa, na página da web de Suporte, das notas relevantes sobre o release, que serão retornadas como uma lista.

- 7.1.1+ Para visualizar uma lista das notas sobre a liberação para a Versão 7.1.1, acesse a Página da Web de Suporte.
- Para visualizar uma lista das notas sobre o release para a Versão 7.1, vá para a Página da Web de Suporte.
- Para visualizar uma lista de notas sobre o release para a Versão 7.0, vá para a Página da Web de Suporte.
- Para visualizar uma lista das notas sobre o release para a Versão 6.1, acesse a página wiki das Notas sobre o Release.

Acessando Requisitos de Software e de Sistema

Os requisitos de hardware e software são documentados nas páginas a seguir:

- Requisitos do Sistema Detalhados

Acessando a Documentação do Produto

Para obter o conjunto de informações inteiro, acesse apágina da Biblioteca.

Acessando o Website de Suporte do Produto

Para procurar as notas técnicas, os downloads, as correções e outras informações relacionadas ao suporte mais recentes, acesse o Support Portal.

Entrando em Contato com o Suporte ao Software IBM®

Se você encontrar um problema com o produto, primeiro, tente as seguintes ações:

- Siga as etapas descritas na documentação do produto

- Procure a documentação relacionada na ajuda on-line
- Consulte as mensagens de erro na referência de mensagens

Se você não conseguir resolver o problema com nenhum dos métodos anteriores, entre em contato com o Suporte Técnico IBM.

Requisitos de Hardware e Software

Pesquisar uma visão geral de hardware e de requisitos do sistema operacional. Embora não seja necessário usar um nível específico de hardware ou sistema operacional para o WebSphere eXtreme Scale, as opções de hardware e software formalmente suportadas estão disponíveis na página Requisitos do Sistema do site de suporte do produto. Se existir um conflito entre o centro de informações e a página Requisitos do Sistema, as informações no website terão precedência. As informações de pré-requisito no centro de informações são fornecidas apenas como uma conveniência.

Consulte a Página de Requisitos do Sistema para obter o conjunto de oficial de requisitos de hardware e software.

Não é necessário instalar e implementar o eXtreme Scale em um nível específico do sistema operacional. Cada instalação do Java Platform, Standard Edition (Java SE) e do Java Platform, Enterprise Edition (Java EE) requer níveis ou correções de sistema operacional diferentes.

É possível instalar e implementar o produto nos ambientes do Java EE e do Java SE. Também é possível incluir em pacote configurável o componente do cliente com os aplicativos Java EE diretamente sem integrar com o WebSphere Application Server. O WebSphere eXtreme Scale suporta o Java SE 5 e posterior e o WebSphere Application Server Versão 6.1 e posterior.

Requisitos de Hardware

O WebSphere eXtreme Scale não requer um nível específico de hardware. Os requisitos de hardware dependem do hardware suportado para a instalação do Java Platform, Standard Edition que é utilizado para executar o WebSphere eXtreme Scale. Se você estiver usando o eXtreme Scale com oWebSphere Application Server ou outra implementação do Java Platform, Enterprise Edition, os requisitos de hardware dessas plataformas são suficientes para oWebSphere eXtreme Scale.

Requisitos do Sistema Operacional

- **Sem o console da web**

  O eXtreme Scale não requer um nível de sistema operacional específico. Cada implementação de Java SE e Java EE necessita de diferentes níveis de sistema operacional ou correções para problemas que são descobertos durante o teste da implementação de Java. Os níveis que as implementações necessitam são suficientes para oeXtreme Scale.
- **Com o console da web**

  Os seguintes requisitos se aplicarão a cada sistema operacional se usar o console:
  - Linux: JVM de 32 bits ou 64 bits
  - Linux PPC: Somente JVM de 32 bits
  - Windows: Somente JVM de 32 bits
  - AIX: Somente JVM de 32 bits

Requisitos do Navegador da Web

O console da web suporta os seguintes navegadores da Web:

- Mozilla Firefox, versão 3.5.x e posterior
- Mozilla Firefox, versão 3.6.x e posterior
- Microsoft Internet Explorer, versão 7 ou 8

Requisitos do WebSphere Application Server

- WebSphere Application Server Versão 6.1.0.39 ou posterior
- WebSphere Application Server Versão 7.0.0.19 ou posterior
- WebSphere Application Server Versão 8.0.0.1 ou posterior

Consulte as Correções recomendadas para o WebSphere Application Server para obter informações adicionais.

Outros Requisitos do Servidor de Aplicativos

Outras implementações de Java EE podem usar o tempo de execução do eXtreme Scale como uma instância local ou como um cliente para servidores eXtreme Scale. Para implementar o Java SE, você deve usar a Versão 5 ou posterior.

Convenções de Diretório

As seguintes convenções de diretório devem ser usadas em todo o documento para referenciar diretórios especiais, como *wxs_install_root* e *wxs_home*. Acesse esses diretórios durante vários cenários diferentes, inclusive durante a instalação e durante o uso das ferramentas de linha de comandos.

wxs_install_root

O diretório *wxs_install_root* é o diretório-raiz no qual os arquivos do produto WebSphere eXtreme Scale são instalados. O diretório *wxs_install_root* pode ser o diretório no qual o archive de teste é extraído ou o diretório no qual o produto WebSphere eXtreme Scale é instalado.

- Exemplo ao extrair o teste:

  Exemplo: `/opt/IBM/WebSphere/eXtremeScale`

- Exemplo quando o WebSphere eXtreme Scale é instalado em um diretório independente:

  Exemplo: `/opt/IBM/eXtremeScale`

- Exemplo quando o WebSphere eXtreme Scale é integrado com o WebSphere Application Server:

  Exemplo: `/opt/IBM/WebSphere/AppServer`

wxs_home

O diretório *wxs_home* é o diretório raiz das bibliotecas, amostras e componentes do produto WebSphere eXtreme Scale. Esse diretório é o mesmo que o diretório *wxs_install_root* quando o teste é extraído. Para instalações independentes, o diretório *wxs_home* é o subdiretório `ObjectGrid` no diretório *wxs_install_root* . Para instalações que estão integradas ao WebSphere Application Server, esse diretório é o diretório `optionalLibraries/ObjectGrid` dentro do diretório *wxs_install_root*.

- Exemplo ao extrair o teste:

  Exemplo: `/opt/IBM/WebSphere/eXtremeScale`

- Exemplo quando o WebSphere eXtreme Scale é instalado em um diretório independente:

Exemplo: `/opt/IBM/eXtremeScale/ObjectGrid`

- Exemplo quando o WebSphere eXtreme Scale é integrado com o WebSphere Application Server:

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

was_root
O diretório *was_root* é o diretório-raiz de uma instalação do WebSphere Application Server:

Exemplo: `/opt/IBM/WebSphere/AppServer`

restservice_home
O diretório *restservice_home* é o diretório no qual as bibliotecas e amostras do serviço de dados REST do WebSphere eXtreme Scale estão localizadas. Este diretório é denominado `restservice` e é um subdiretório sob o diretório *wxs_home* .

- Exemplo para implementações independentes:

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

- Exemplo para implementações integradas do WebSphere Application Server:

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

tomcat_root
O *tomcat_root* é o diretório-raiz da instalação do Apache Tomcat.

Exemplo: `/opt/tomcat5.5`

wasce_root
O *wasce_root* é o diretório-raiz da instalação do WebSphere Application Server Community Edition.

Exemplo: `/opt/IBM/WebSphere/AppServerCE`

java_home
java_home é o diretório-raiz de uma instalação de Java Runtime Environment (JRE).

Exemplo: `/opt/IBM/WebSphere/eXtremeScale/java`

samples_home
O *samples_home* é o diretório no qual os arquivos de amostra usados para os tutoriais são extraídos.

Exemplo: `/wxs-samples/`

dvd_root
O diretório *dvd_root* é o diretório-raiz do DVD que contém o produto.

Exemplo: `dvd_root/docs/`

equinox_root
O diretório *equinox_root* é o diretório raiz da instalação da estrutura do Eclipse Equinox OSGi.

Exemplo:`/opt/equinox`

user_home
O diretório *user_home* é o local no qual arquivos de usuário são armazenados, tais como perfis de segurança.

Windows `c:\Documents and Settings\user_name`

UNIX `/home/user_name`

Visão Geral Técnica do WebSphere eXtreme Scale

WebSphere eXtreme Scale é uma grade de dados na memória, elástica e escalável. Ele dinamicamente armazena em cache, particiona, replica e gerencia dados do aplicativo e lógica de negócios em múltiplos servidores.

Como o WebSphere eXtreme Scale não é um banco de dados em memória, você deve considerar requisitos de configuração específicos. A primeira etapa para implementar uma grade de dados é iniciar um grupo principal e o serviço de catálogo, que atua como coordenador para todas as outras Java virtual machines que estiverem participando na grade de dados, e gerenciar as informações de configuração. Os processos do WebSphere eXtreme Scale são iniciados com chamadas de script de comando simples a partir da linha de comandos.

A próxima etapa é iniciar os processos do servidor WebSphere eXtreme Scale da grade de dados para armazenar e recuperar dados. Conforme os servidores são iniciados, eles se registram automaticamente no grupo principal e no serviço de catálogo, para que eles possam cooperar no fornecimento de serviços da grade de dados. Uma quantidade maior de servidores aumenta tanto a capacidade quanto a confiabilidade da grade de dados.

Uma grade de dados local é uma grade de instância única e simples na qual todos os dados estão em uma grade. Para usar eficientemente o WebSphere eXtreme Scale como um espaço de processamento de banco de dados na memória, é possível configurar e implementar uma grade de dados distribuída. Os dados na grade distribuída são espalhados por vários servidores do eXtreme Scale que a contêm, sendo que cada servidor contém somente parte dos dados, chamada de partição.

Um parâmetro de configuração principal da grade de dados distribuída é o número de partições na grade. Os dados da grade são particionados nesta quantidades de subconjuntos, cada um dos quais chamados de partição. O serviço de catálogo localiza a partição para um determinado datum com base em sua chave. O número de partições afeta diretamente a capacidade e a escalabilidade da grade de dados. Um servidor pode conter uma ou mais partições de grade de dados. Dessa forma, o espaço de memória do servidor limita o tamanho de uma partição. Por outro lado, aumentar o número de partições aumenta a capacidade da grade de dados. A capacidade máxima de uma grade de dados é o número de partições vezes o tamanho da memória utilizável de cada servidor. Um servidor pode ser uma JVM, porém é possível definir seu servidor eXtreme Scale de acordo com seu ambiente de implementação.

Os dados de uma partição são armazenados em um shard. Para disponibilidade, uma grade de dados pode ser configurada com réplicas, que podem ser síncronas ou assíncronas. Alterações nos dados da grade são feitas no shard primário, e replicadas nos shards de réplica. Portanto, a memória total que é usada ou necessária por uma grade de dados é o tamanho da grade vezes (1 (para o primário) + a quantidade de réplicas).

O WebSphere eXtreme Scale distribui shards de uma grade de dados sobre o número de servidores que compõe a grade. Estes servidores podem estar nos mesmos, ou em diferentes, servidores físicos. Para disponibilidade, os shards de réplica são colocados em servidores físicos separados dos shards primários.

O WebSphere eXtreme Scale monitora o status de seus servidores e movimenta os shards durante falha e recuperação de shard ou de servidor físico. Por exemplo, se

o servidor que contém um shard de réplica falhar, o WebSphere eXtreme Scale alocará um novo shard de réplica e replicará os dados do primário na nova réplica. Se um servidor que contém um shard primário falhar, o shard de réplica será promovido para ser o shard primário e um novo shard de réplica será construído. Se um servidor adicional for iniciado para a grade de dados, os shards serão equilibrados entre todos os servidores. Esse re-equilíbrio é chamado de scale-out. Da mesma forma, para o scale-in, é possível parar um dos servidores para reduzir os recursos que são usados por uma grade de dados. Como resultado, os shards estão equilibrados entre os servidores restantes.

Visão Geral do Armazenamento em Cache

O WebSphere eXtreme Scale pode operar como um espaço de processamento de banco de dados de memória, que pode ser utilizado para fornecer armazenamento em cache sequencial para um backend de banco de dados ou atuar como um cache secundário. O armazenamento em cache sequencial utiliza o eXtreme Scale como o meio principal de interação com os dados. Quando o eXtreme Scale é usado como um cache secundário, o backend é usado junto com a grade de dados. Esta seção descreve vários conceitos e cenários de cache e discute as topologias disponíveis para implementar uma grade de dados.

Arquitetura de Armazenamento em Cache: Mapas, Contêineres, Clientes e Catálogos

Com WebSphere eXtreme Scale, sua arquitetura pode utilizar armazenamento em cache de dados em memória local ou armazenamento em cache de dados de cliente/servidor distribuídos.

O WebSphere eXtreme Scale requer infraestrutura adicional mínima para operar. A infraestrutura consiste em scripts para instalar, iniciar e parar um aplicativo Java Platform, Enterprise Edition em um servidor. Os dados armazenados em cache são armazenados no servidor eXtreme Scale e os clientes conectam-se remotamente ao servidor.

Caches distribuídos oferecem desempenho, disponibilidade e escalabilidade melhorados e podem ser configurados usando topologias dinâmicas, nas quais os servidores são equilibrados automaticamente. Também é possível incluir servidores adicionais sem restaurar seus servidores eXtreme Scale existentes. É possível criar implementações simples ou grandes a nível de terabytes, em que milhares de servidores são necessários.

Serviço de Catálogo

O serviço de catálogo hospeda lógica que deve estar inativa durante um estado estável e tem pouca influência na escalabilidade. O serviço de catálogo é construído para atender a centenas de contêineres que se tornam disponíveis simultaneamente e executa serviços para gerenciar os contêineres.

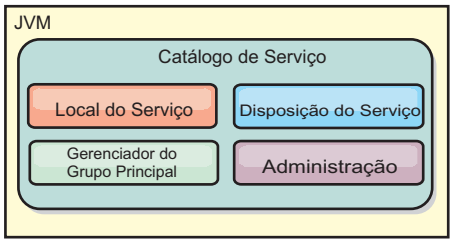


Figura 2. Serviço de Catálogo

As responsabilidades de catálogo consistem nos seguintes serviços:

Serviço de local
O serviço de local fornece localidade para clientes que estão procurando aplicativos de hosting de contêineres e para contêineres que estão procurando registrar aplicativos hospedados com o serviço de disposição. O serviço de local é executado em todos os membros da grade para efetuar o scale out desta função.

Serviço de disposição
O serviço de disposição é o sistema nervoso central para a grade e é responsável por alocar shards individuais para seu contêiner de host. O serviço de posicionamento é executado como Um de N serviços eleitos no cluster. Como a política Um de N é usada, há sempre exatamente uma instância do serviço de posicionamento em execução. Se tal instância deve ser interrompida, outro processo assume. Todos os estados do serviço de catálogo são replicados em todos os servidores hospedando o serviço de catálogo para redundância.

Gerenciador de grupo principal
O gerenciador de grupo principal gerencia agrupamento peer para monitoramento de funcionamento, organiza contêineres em grupos menores de servidores e automaticamente federa os grupos de servidores. Quando um contêiner entra em contato com o serviço de catálogo pela primeira vez, ele aguarda para ser designado a um grupo novo ou existente de várias Java virtual machines (JVM). Cada grupo das Java virtual machines monitora a disponibilidade de cada um dos membros por meio da pulsação. Um destes membros do grupo retransmite as informações de disponibilidade ao serviço de catálogo para permitir reação a falhas por meio de realocação e encaminhamento de rotas.

Administração
Os quatro estágios de administração do seu ambiente do WebSphere eXtreme Scale são planejamento, implementação, gerenciamento e monitoramento.

Para disponibilidade, configure um domínio do serviço de catálogo. Um domínio do serviço de catálogo consiste em várias Java virtual machines, incluindo uma JVM principal e várias Java virtual machines de backup.

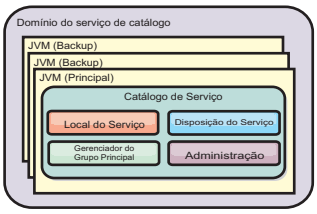


Figura 3. Domínio do Serviço de Catálogo

Servidores de Contêiner, Partições e Shards

O servidor de contêiner armazena dados do aplicativo para a grade de dados. Estes dados geralmente são divididos em partes, chamadas de partições, que são hospedadas em vários servidores de contêiner. Cada servidor de contêiner, por sua vez, hospeda um subconjunto de dados completos. Uma JVM pode hospedar um ou mais servidores de contêiner e cada servidor de contêiner pode hospedar diversos shards.

Lembre-se: Planeje o tamanho do heap para os servidores de contêiner, que hospedam todos os dados. Configure as configurações de heap apropriadamente.

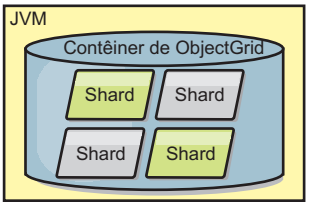


Figura 4. Servidor de Contêineres

Partições hospedam um subconjunto de dados na grade. O WebSphere eXtreme Scale coloca automaticamente diversas partições em um único servidor de contêiner e propaga as partições à medida que mais servidores de contêiner se tornam disponíveis.

Importante: Escolha o número de partições cuidadosamente antes da implementação final já que o número de partições não pode ser alterado dinamicamente. Um mecanismo hash é utilizado para localizar partições na rede e o eXtreme Scale não pode re-hash o conjunto de dados inteiro após ele ser implementado. Como uma regra geral, é possível superestimar o número de partições

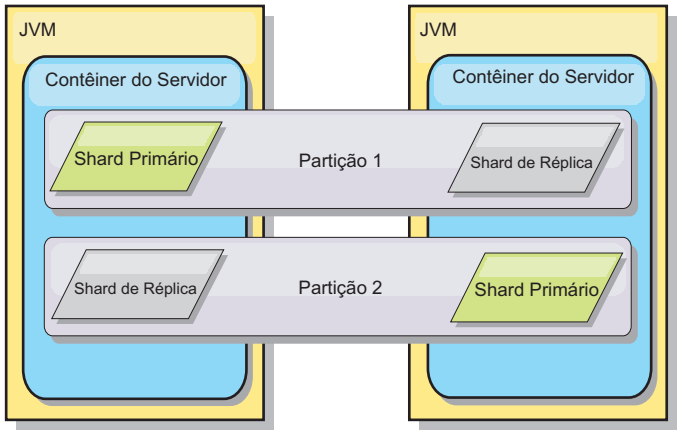


Figura 5. Partição

Os shards são instâncias de partições e possuem uma das duas funções: primário ou de réplica. O fragmento primário e suas réplicas constituem a manifestação física da partição. Cada partição tem vários shards que hospedam, cada um deles, todos os dados contidos nessa partição. Um shard é o primário e os outros são réplicas, que são cópias redundantes dos dados no primeiro shard. Um fragmento primário é a única instância da partição que permite que as transações sejam gravadas no cache. Um fragmento de réplica é uma instância "espelhada" da partição. Ele recebe atualizações de forma síncrona e assíncrona do fragmento primário. O fragmento de réplica permite apenas a leitura das transações do cache. As réplicas nunca são hospedadas no mesmo servidor de contêiner que os primários e normalmente não são hospedados na mesma máquina que os primários.

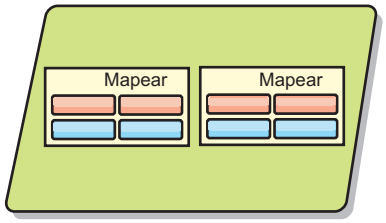


Figura 6. Shard

Para aumentar a disponibilidade dos dados, ou aumentar garantias de persistência, replique os dados. Entretanto, a replicação inclui custo na transação e troca desempenho em retorno para a disponibilidade. Com o eXtreme Scale, é possível controlar o custo já que ambas as replicações síncrona e assíncrona são suportadas, bem como modelos de replicação híbrida utilizando os modos de replicação síncrona e assíncrona. O shard da réplica síncrona recebe atualizações como parte da transação do shard primário para garantir consistência de dados. Uma réplica síncrona pode duplicar o tempo de resposta porque a transação precisa executar commit na réplica primária e na réplica síncrona antes da transação ser concluída. Um shard de réplica assíncrono recebe atualizações após o commit da transação para limitar o impacto no desempenho, mas introduz a possibilidade de perda de dados enquanto que a réplica assíncrona pode ser diversas transações atrás do shard primário.

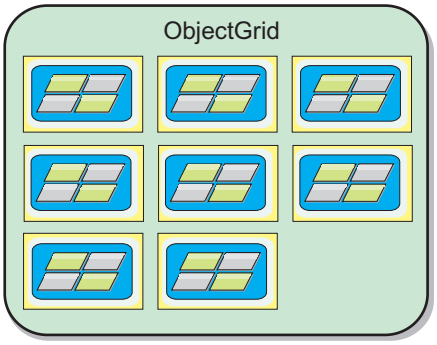


Figura 7. ObjectGrid

Mapas

Um mapa é um contêiner para pares chave/valor, que permite que um aplicativo armazene um valor indexado por uma chave. Os mapas suportam índices que podem ser incluídos nos atributos de índice na chave ou no valor. Esses índices são automaticamente usados pelo tempo de execução de consulta para determinar a maneira mais eficiente de executar uma consulta.

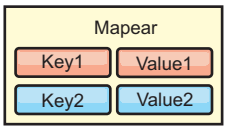


Figura 8. Mapa

Um conjunto de mapas é uma coleta de mapas com um algoritmo de particionamento comum. Os dados nos mapas são replicados com base na política definida no conjunto de mapas. Um conjunto de mapas é utilizado apenas para topologias distribuídas e não é necessário para topologias locais.

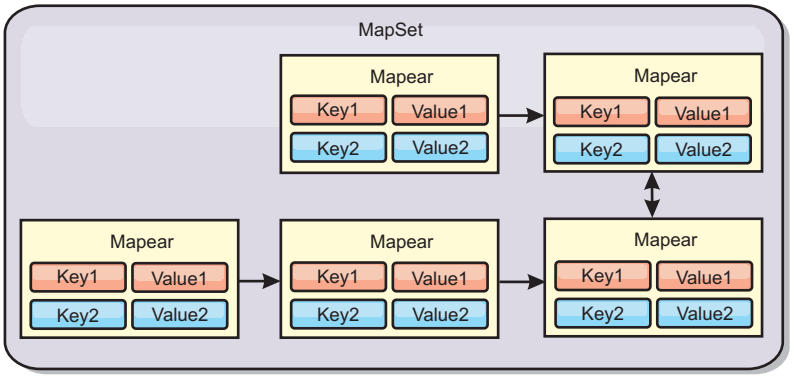


Figura 9. Conjuntos de Mapas

Um conjunto de mapas pode ter um esquema associado a ele. Um esquema são os metadados que descrevem os relacionamentos entre cada mapa ao utilizar tipos ou entidades de Objeto homogêneos.

O WebSphere eXtreme Scale pode armazenar objetos Java serializáveis em cada um dos mapas usando a API ObjectMap. Um esquema pode ser definido nos mapas para identificar o relacionamento entre os objetos nos mapas, em que cada mapa suspende objetos de um único tipo. A definição de um esquema para mapas é necessária para consultar o conteúdo dos objetos de mapa. O WebSphere eXtreme Scale pode ter vários esquemas de mapa definidos.

O WebSphere eXtreme Scale também pode armazenar entidades utilizando a API do EntityManager. Cada entidade está associada a um mapa. O esquema para um conjunto de mapas de entidade é automaticamente descoberto usando um arquivo XML do descritor de entidade ou classes Java anotadas. Cada entidade tem um conjunto de atributos-chave e um conjunto de atributos não-chave. Uma entidade também pode ter relacionamentos com outras entidades. O WebSphere eXtreme Scale suporta relacionamentos um para um, um para muitos, muitos para um e muitos para muitos. Cada entidade é mapeada fisicamente para um único mapa no conjunto de mapas. As entidades permitem que os aplicativos tenham facilmente gráficos de objeto complexos que expandem vários Mapas. Uma topologia distribuída pode ter vários esquemas de entidade.

Para obter informações adicionais, consulte Objetos de Armazenamento em Cache e seus Relacionamentos (API EntityManager).

Clientes

Os clientes se conectam a um serviço de catálogo, recuperam uma descrição da topologia do servidor e se comunicam diretamente com cada servidor, conforme necessário. Quando a topologia do servidor é alterada porque novos servidores foram incluídos ou porque servidores existentes falharam, o serviço de catálogo dinâmico roteia o cliente para o servidor apropriado que está hospedando os dados. Os clientes devem examinar as chaves dos dados do aplicativo para determinar para qual partição o pedido deve ser roteado. Os Clientes podem ler dados de várias partições em uma única transação. Entretanto, os clientes podem atualizar apenas uma única partição em uma transação. Após o cliente atualizar algumas entradas, a transação do cliente deve utilizar tal partição para atualizações.

As possíveis combinações de implementação são incluídas na lista a seguir:

- Um serviço de catálogo existe em sua própria grade de Java Virtual Machines. Um único serviço de catálogo pode ser utilizado para gerenciar vários clientes ou servidores do eXtreme Scale.
- Um contêiner pode ser iniciado em uma JVM por si ou pode ser carregado em uma JVM arbitrária com os outros contêineres para instâncias diferentes do ObjectGrid.
- Um cliente pode existir em algum JVM e comunicar-se com uma ou mais instâncias do ObjectGrid. Também pode existir um cliente na mesma JVM que um contêiner.

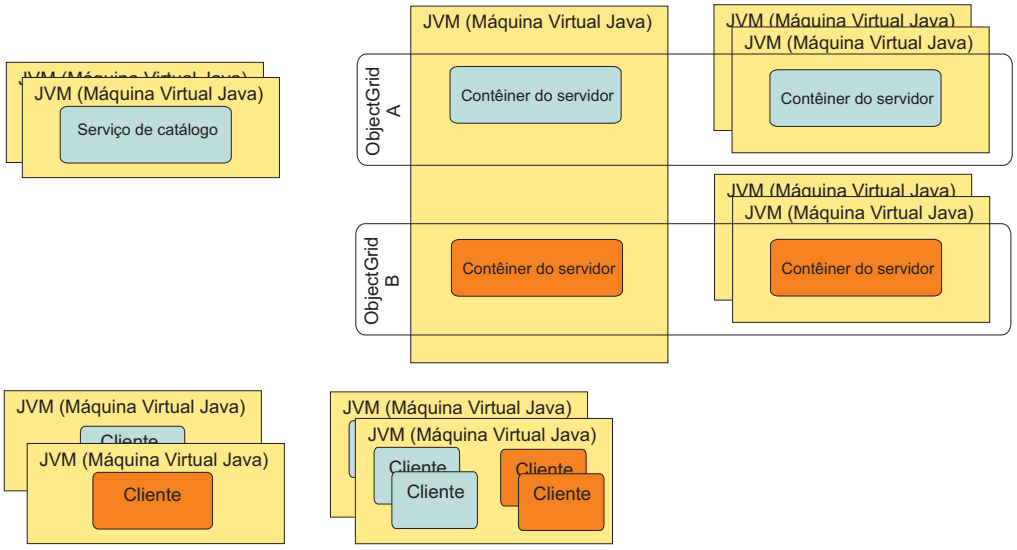


Figura 10. Possíveis Topologias

Zonas

As zonas fornecem um controle sobre o posicionamento de shard. As zonas são agrupamentos lógicos definidos pelo usuário de servidores físicos. A seguir há exemplos de diferentes tipos de zonas: servidores blade diferentes, chassi de servidores blade, pisos de um prédio, edifícios ou diferentes locais geográficas em um ambiente de diversos datacenters. Outro caso de uso está em um ambiente virtualizado onde muitas instâncias do servidor, cada uma com um endereço IP exclusivo, são executadas no mesmo servidor físico.

Zonas Definidas Entre os Datacenters

O exemplo e o caso de uso clássicos para zonas é quando você possui dois ou mais datacenters dispersos geograficamente. Os datacenters dispersos propagam sua grade de dados para diferentes locais para recuperação de falha do datacenter. Por exemplo, você pode querer garantir ter um conjunto completo de shards de réplica assíncronos para sua grade de dados em um datacenter remoto. Com essa estratégia, é possível recuperar-se da falha do datacenter local de modo transparente, sem perda de dados. Os datacenters em si possuem redes de alta velocidade e de baixa latência. No entanto, a comunicação entre um datacenter e outro tem latência mais alta. As réplicas síncronas são usadas em cada datacenter onde a baixa latência minimiza o impacto da replicação nos tempos de resposta. Usar a replicação assíncrona reduz o impacto no tempo de resposta. A distância geográfica fornece disponibilidade no caso de falha de um datacenter local.

No exemplo a seguir, os shards primários para a zona Chicago possui réplicas na zona London. Os shards primários para a zona London possuem réplicas na zona Chicago.

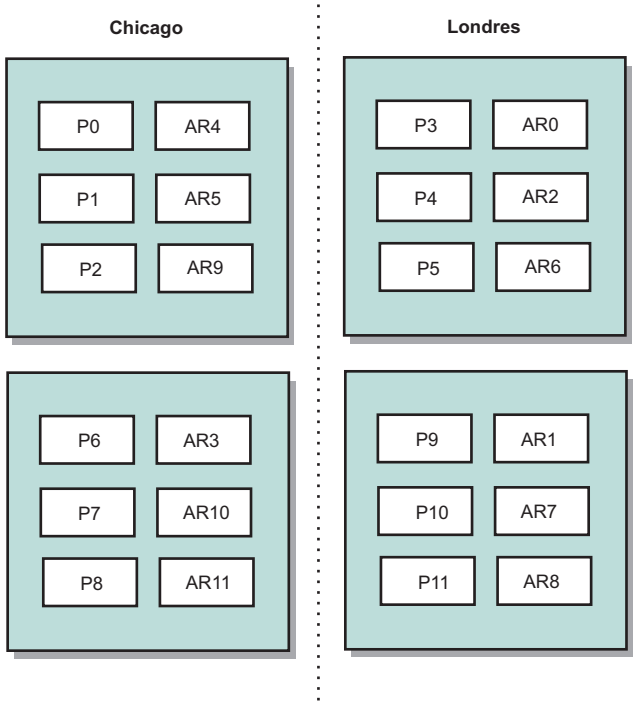


Figura 11. Primários e Réplicas em Zonas

Três definições de configuração no eXtreme Scale controlam o posicionamento do shard:

- Configurar o arquivo de implementação
- Grupo de contêineres
- Especificar regras

As seções a seguir explicam as diferentes opções, apresentadas das mais simples até as mais complicadas.

Modo Desativar Desenvolvimento

No seu arquivo XML de implementação, configure: `developmentMode="false"`.

Esta etapa simples ativa a primeira política de posicionamento de shard do eXtreme Scale.

Para obter mais informações sobre o arquivo XML, consulte Arquivo Descritor XML de Política de Implementação.

Política 1: Os shards para a mesma partição são colocados em servidores físicos separados.

Considere um exemplo simples de uma grade de dados com um shard de réplica. Com esta política, os shards primários e de réplica para cada partição estão em servidores físicos diferentes. Se um único servidor físico falhar, nenhum dado será perdido. Os shards primários ou de réplica de cada partição estão em servidores físicos diferentes que não falhou, ou estão em algum outro servidor físico que também não falhou.

A alta disponibilidade e simplicidade dessa política torna a configuração mais eficiente para todos os ambientes de produção. Em muitos casos, aplicar essa política é a única etapa necessária para controlar efetivamente o posicionamento de shard em seu ambiente.

Para aplicar essa política, um servidor físico é definido por um endereço IP. Os shards são colocados em servidores de contêiner. Os servidores de contêiner possuem um endereço IP, por exemplo, o parâmetro **-listenerHost** no script **startOgServer**. Diversos servidores de contêiner podem ter o mesmo endereço IP.

Como um servidor físico possui diversos endereços IP, considere a próxima etapa para obter mais controle de seu ambiente.

Defina Zonas para Agrupar Servidores de Contêiner

Os servidores de contêiner são designados à zonas com o parâmetro **-zone** no script startOgServer. Em um ambiente do WebSphere Application Server, as zonas são definidas por meio de grupos de nós com um formato de nome específico: `ReplicationZone<Zone>`. Neste modo, você escolhe o nome e a associação de suas zonas. Para obter informações adicionais, consulte Definindo Zonas para Servidores de Contêiner.

Política 2: Os shards da mesma partição são colocados em zonas separadas.

Considere estender o exemplo de uma grade de dados com um shard de réplica ao implementá-lo entre dois datacenters. Defina cada datacenter como uma zona independente. Use um nome de zona de DC1 para os servidores de contêiner no primeiro datacenter e o nome DC2 para os servidores de contêiner no segundo datacenter. Com esta política, os shards primários e de réplica para cada partição estariam em datacenters diferentes. Se um datacenter falhar, nenhum dado será perdido. Para cada partição, o shard primário ou de réplica está no outro datacenter.

Com esta política, é possível controlar o posicionamento do shard ao definir zonas. Pode-se também escolher seu limite físico ou lógico ou um agrupamento desejado. Em seguida, escolha um nome de zona exclusivo para cada grupo e inicie os servidores de contêiner em cada uma de suas zonas com o nome da zona apropriado. Assim, o eXtreme Scale coloca shards de modo que os shards da mesma partição sejam colocados em zonas separadas.

Especificar Regras da Zona

O nível de controle mais fino sobre o posicionamento de shard é alcançado usando regras de zona. As regras de zona são especificadas no elemento zoneMetadata do descritor XML da política de implementação do eXtreme Scale. Uma regra de zona define um conjunto de zonas no qual os shards são colocados. Um elemento shardMapping designa um shard para uma regra de zona. O atributo shard do elemento shardMapping especifica o tipo de shard:

- P especifica o shard primário
- S especifica shards de réplica síncronos
- A especifica shards de réplica assíncronos.

Se mais de uma réplica síncrona ou assíncrona existir, você deverá fornecer elementos shardMapping do tipo de shard apropriado. O atributo exclusivePlacement do elemento zoneRule determina o posicionamento dos shards na mesma partição em zonas. Os valores do atributo exclusivePlacement são:

- `true` (um shard não pode ser colocado na mesma zona do outro shard da mesma partição).

  Lembre-se: Para o caso "true", você deve ter pelo menos tantas zonas na regra quanto possui shards usando-a. Fazer isso assegura que cada shard esteja em sua própria zona.
- `false` (os shards da mesma partição podem ser colocados na mesma zona.

A configuração padrão é true.

Para obter informações adicionais, consulte Exemplo: Definições de Zona no Arquivo Descritor XML de Política de Implementação.

Casos de Uso Estendidos

A seguir há vários casos de uso para estratégias de posicionamento de shard:

Upgrades contínuos

Considere um cenário no qual você deseja aplicar upgrades contínuos nos seus servidores físicos, incluindo a manutenção que requer o reinício de sua implementação. Neste exemplo, suponha que você tenha uma grade de dados espalhada por 20 servidores físicos, definida com uma réplica síncrona. Você deseja encerrar 10 dos servidores físicos de uma vez para manutenção.

Quando você encerra grupos de 10 servidores físicos, nenhuma partição possui shards primário e de réplica nos servidores que estão sendo encerrados. Caso contrário, os dados dessa partição são perdidos.

A solução mais fácil é definir uma terceira zona. Em vez de duas zonas de 10 servidores físicos cada, use três zonas, duas com sete servidores físicos e uma com seis. Espalhar os dados em mais zonas permite melhor disponibilidade de failover.

Em vez de definir outra zona, a outra abordagem é incluir uma réplica.

Fazendo Upgrade do eXtreme Scale

Quando você estiver fazendo upgrade do software do eXtreme Scale de forma contínua com grades de dados contendo dados ativos, considere as seguintes questões. A versão do software de serviço de catálogo deve ser maior ou igual às versões do software do servidor de contêiner. Primeiro faça upgrade de todos os servidores de catálogos com uma estratégia contínua. Leia mais sobre como fazer upgrade de sua implementação no tópico Atualizando Servidores eXtreme Scale.

Alterando o Modelo de Dados

Uma questão relacionada é como alterar o modelo de dados ou o esquema de objetos que são armazenados na grade de dados sem causar tempo de inatividade. Seria problemático alterar o modelo de dados ao parar a grade de dados e reiniciar com as classes do modelo de dados atualizadas no caminho de classe do servidor de contêiner e recarregando a grade de dados. Uma alternativa seria iniciar uma nova grade de dados com o novo esquema, copiar os dados da grade de dados antiga para a nova grade de dados e, em seguida, encerrar a grade de dados antiga.

Cada um desses processos é problemático e resulta em tempos de inatividade. Para alterar o modelo de dados sem causar tempo de inatividade, armazene o objeto em um destes formatos:

- Use o XML como o valor
- Use um blob feito com protocolo de buffers Google
- Use o JavaScript Object Notation (JSON)

Grave serializadores para ir desde o plain old Java object (POJO) até um desses formatos facilmente no lado do cliente. As mudanças de esquema se tornam mais fáceis.

Virtualização

Computação em nuvem e virtualização são tecnologias populares emergentes. Por padrão, o eXtreme Scale garante que dois shards da mesma partição nunca sejam colocados no mesmo endereço IP, conforme descrito na Política 1. Quando você estiver implementando as imagens virtuais, tais como VMware, muitas instâncias do servidor, cada uma com um endereço IP exclusivo, podem ser executadas no mesmo servidor físico. Para assegurar que as réplicas sejam colocadas em servidores físicos separados, é possível usar zonas para resolver o problema. Agrupe seus servidores físicos nas zonas e use as regras de posicionamento de zona para manter os shards primários e de réplica em zonas separadas.

Zones para redes remotas

Você pode querer implementar uma única grade de dados do eXtreme Scale em várias construções ou datacenters com conexões de rede mais lentas. Conexões de rede mais lentas levam a uma largura da banda menor e mais conexões de latência. A possibilidade de partições de rede também aumenta neste modo devido ao congestionamento de rede e outros fatores.

Para lidar com esses riscos, o serviço de catálogo do eXtreme Scale organiza os servidores de contêiner em grupos principais que trocam pulsações para detectar uma falha do servidor de contêiner. Esses grupos principais não se estendem pelas zonas. Um líder dentro de cada grupo principal envia informações de associação no serviço de catálogo. O serviço de catálogo verifica quaisquer falhas relatadas antes de responder a uma informação de associação pela pulsação do servidor de contêiner em questão. Se o serviço de catálogo visualizar uma falsa detecção de falha, o serviço de catálogo não executará nenhuma ação. A partição do grupo principal se recompõe rapidamente. O serviço de catálogo também faz pulsações dos líderes do grupo principal periodicamente a uma taxa mas baixa para tratar o caso de isolamento do grupo principal.

Evictors

Dados removidos dos evictors a partir da grade de dados. É possível configurar um evictor baseado em tempo ou, como os evictors estão associados aos BackingMaps, usar a interface BackingMap para especificar o evictor conectável.

Evictor Padrão

Um evictor TTL padrão é criado com cada mapa de apoio. O evictor padrão remove entradas com base em um conceito de time-to-live. Este comportamento é definido pelo atributo ttlType, que tem os seguintes tipos:

Nenhum

Especifica que as entradas nunca expiram e, portanto, nunca são removidas do mapa.

Tempo de criação

Especifica que as entradas são despejadas dependendo de quando foram criadas.

Se você estiver usando o ttlType CREATION_TIME, o evictor despejará uma entrada quando seu tempo de criação for igual ao valor do seu atributo TimeToLive, que é configurado em milissegundos na configuração do aplicativo. Se você configurar o valor do atributo TimeToLive como 10 segundos, a entrada será despejada automaticamente dez segundos após ser inserida.

É importante ter atenção ao configurar este valor para o CREATION_TIME ttlType. Este evictor é melhor utilizado quando existem quantidades de inclusões no cache razoavelmente altas que são utilizadas apenas durante uma quantidade de tempo configurada. Com esta estratégia, tudo o que é criado é removido após a quantidade de tempo configurada.

O ttlType CREATION_TIME é útil em cenários como a atualização de cotas de estoque a cada 20 minutos ou menos. Suponha que um aplicativo da Web obtenha cotas de estoque e que obter as cotas mais recentes não seja crítico. Neste caso, os preços de ações são armazenados em cache em um ObjectGrid por 20 minutos. Após 20 minutos, as entradas do mapa do ObjectGrid expiram e são liberadas. A cada vinte minutos ou quase, o mapa do ObjectGrid usa o plug-in do Carregador para atualizar os dados do mapa com dados atualizados do banco de dados. O banco de dados é atualizado a cada 20 minutos com os preços de ações mais recentes.

Horário do último acesso

Especifica que as entradas são despejadas dependendo de quando foram acessadas pela última vez, se foram lidas ou atualizadas.

Horário da última atualização

Especifica que as entradas são despejadas dependendo de quando foram atualizadas pela última vez.

Se você estiver usando o atributo ttlType LAST_ACCESS_TIME ou LAST_UPDATE_TIME, configure TimeToLive para um número menor do que se você estivesse usando o ttlType CREATION_TIME, porque o atributo TimeToLive de entradas é reconfigurado toda vez que é acessado. Em outras palavras, se o atributo TimeToLive for igual a 15 e existir uma entrada por 14 segundos, mas for acessada, ela não expirará novamente durante outros 15 segundos. Se você configurar o TimeToLive como um número relativamente alto, muitas entradas poderão nunca ser liberadas. No entanto, se você configurar o valor como algo semelhante a 15 segundos, as entradas poderão ser removidas quando não forem acessadas com freqüência.

O ttlType LAST_ACCESS_TIME ou LAST_UPDATE_TIME é útil em cenários como a manutenção de dados da sessão de um cliente, usando um mapa do ObjectGrid. Os dados de sessão devem ser destruídos se o cliente não utilizá-los por algum período de tempo. Por exemplo, é excedido o tempo limite dos dados de sessão após 30 minutos sem atividade do cliente. Neste caso, usar um tipo TTL de LAST_ACCESS_TIME ou LAST_UPDATE_TIME com o atributo TimeToLive configurado para 30 minutos é adequado para este aplicativo.

Também é possível gravar seus próprios evictors: Para obter informações adicionais, consulte Gravando um Evictor Customizado.

Evictor Conectável

O evictor TTL padrão utiliza uma política de evicção baseada no tempo e o número de entradas no BackingMap não tem efeito sobre o tempo de expiração de uma entrada. É possível utilizar um evictor conectável opcional para despejar entradas com base no número de entradas existentes em vez de no tempo.

Os seguintes evictores conectáveis opcionais fornecem alguns algoritmos comumente utilizados para decidir quais entradas liberar quando um BackingMap crescer além de algum limite de tamanho.

- O evictor LRUEvictor utiliza um algoritmo LRU (Least Recently Used) para decidir quais entradas serão despejadas quando o BackingMap exceder um número máximo de entradas.
- O evictor LFUEvictor utiliza um algoritmo LFU (Least Frequently Used) para decidir quais entradas serão despejadas quando o BackingMap exceder um número máximo de entradas.

O BackingMap informa um evictor conforme as entradas são criadas, modificadas ou removidas de uma transação. O BackingMap acompanha estas entradas e escolhe quando liberar uma ou mais entradas da instância do BackingMap.

Uma instância do BackingMap não possui informações de configuração para um tamanho máximo. Em vez disso, as propriedades do evictor são configuradas para controlar o comportamento do evictor. O LRUEvictor e o LFUEvictor possuem uma propriedade de tamanho máximo utilizada para fazer o evictor começar a liberar entradas quando o tamanho máximo for excedido. Assim como o evictor TTL, os evictores LRU e LFU podem não liberar imediatamente uma entrada quando o número máximo de entradas for atingido para minimizar o impacto no desempenho.

Se o algoritmo LRU ou LFU não for adequado para um determinado aplicativo, será possível redigir seus próprios evictors para criar a sua estratégia de despejo.

Despejo Baseado em Memória

Importante: O despejo baseado em memória é suportado somente no Java Platform, Enterprise Edition Versão 5 ou posterior.

Todos os evictores integrados suportam despejo baseado em memória, que pode ser ativado na interface BackingMap por meio da configuração do atributo evictionTriggers de BackingMap como `MEMORY_USAGE_THRESHOLD`. Para obter mais informações sobre como configurar o atributo evictionTriggers no BackingMap, consulte o Interface BackingMap e o Arquivo XML descritor do ObjectGrid.

O despejo baseado em memória é baseado no limite de uso do heap. Quando o despejo baseado em memória é ativado no BackingMap e o BackingMap possui algum evictor integrado, o limite de uso é configurado com uma porcentagem padrão de memória total se o limite ainda não tiver sido configurado anteriormente.

Ao usar o despejo baseado em memória, você deveria configurar o limite de coleta de lixo para o mesmo valor que a utilização de heap de destino. Por exemplo, se o

limite de despejo baseado em memória for configurado em 50 por cento e o limite de coleta de lixo estiver configurado no nível padrão de 70 por cento, então a utilização de heap pode chegar no máximo a 70 por cento. Esta aumento da utilização de heap ocorre proque o despejo baseado em memória é acionado somente depois de um ciclo de coleta de lixo.

Para alterar a porcentagem de limite de uso, configure a propriedade memoryThresholdPercentage no contêiner e o arquivo de propriedades do servidor para o processo do servidor do eXtreme Scale. Para configurar o limite de uso de destino em um processo de cliente, é possível usar o MemoryPoolMXBean.

O algoritmo de despejo baseado em memória usado pelo WebSphere eXtreme Scale é sensível ao comportamento do algoritmo de coleta de lixo em uso. O melhor algoritmo para despejo baseado em memória é o padrão da IBM através do coletor. A geração de algoritmos de coleta de lixo podem provocar comportamento indesejado e, dessa forma, você não deve usar estes algoritmos com o despejo baseado em memória.

Para alterar a porcentagem de limite de uso, configure a propriedade memoryThresholdPercentage no contêiner e os arquivos de propriedades do servidor para os processos do servidor do eXtreme Scale.

Durante o tempo de execução, se o uso da memória exceder o limite de uso destinado, os evictors baseados em memória iniciam o despejo de entradas e tentam manter o uso da memória abaixo do limite de uso destinado. Porém, não há garantia de que a velocidade do despejo seja rápida o suficiente para evitar um possível erro de falta de memória se o tempo de execução do sistema continuar consumindo memória rapidamente.

Visão Geral da Estrutura do OSGi

O OSGi define um sistema módulo dinâmico para Java. A plataforma de serviço OSGi possui uma arquitetura em camadas e é projetada para ser executada em vários perfis padrão Java. É possível iniciar servidores e clientes do WebSphere eXtreme Scale em um contêiner OSGi.

Benefícios de Executar Aplicativos no Contêiner OSGi

O suporte do WebSphere eXtreme Scale OSGi permite implementar o produto na estrutura do Eclipse Equinox OSGi. Anteriormente, se você desejava atualizar os plug-ins usados pelo eXtreme Scale, era necessário reiniciar a Java Virtual Machine (JVM) para aplicar as novas versões dos plug-ins. Com a capacidade de atualização dinâmica que a estrutura OSGi fornece, agora é possível atualizar as classes de plug-in sem reiniciar a JVM. Esses plug-ins são exportados pelos pacotes configuráveis do usuário como serviços. O WebSphere eXtreme Scale acessa o serviço ou serviços consultando-os no registro OSGi.

Os contêineres do eXtreme Scale podem ser configurados para iniciar mais fácil e dinamicamente usando o serviço administrativo de configuração do OSGi ou com o OSGi Blueprint. Se desejar implementar uma nova grade de dados com sua estratégia de posicionamento, será possível fazer isso criando uma configuração de OSGi ou implementando um pacote configurável com arquivos XML do descritor eXtreme Scale. Com o suporte do OSGi, os pacotes configuráveis de aplicativo que contém dados de configuração do eXtreme Scale podem ser instalados, iniciados,

interrompidos, atualizados e desinstalados sem reiniciar o sistema inteiro. Com esta capacidade, é possível fazer upgrade do aplicativo sem interromper a grade de dados.

Beans de plug-in e serviços podem ser configurados com escopos de shard customizados, permitindo opções de integração sofisticadas com outros serviços em execução na grade de dados. Cada plug-in pode usar classificações do OSGi Blueprint para verificar se cada instância do plug-in ativada está na versão correta. Um bean gerenciado por OSGi (MBean) e o utilitário **xscmd** são fornecidos, o que permite que você consulte os serviços OSGi de plug-in do eXtreme Scale e suas classificações.

Este recurso permite que os administradores reconheçam rapidamente erros de configuração e administração em potencial e atualize as classificações de serviço de plug-in em uso pelo eXtreme Scale.

Pacotes Configuráveis OSGi

Para interagir com, e implementar, plug-ins na estrutura do OSGi, você deve usar os *pacotes configuráveis*. Na plataforma de serviço OSGi, um pacote configurável é um arquivo Java archive (JAR) que contém código Java, recursos e um manifesto que descrevem o pacote configurável e suas dependências. O pacote configurável é a unidade de implementação para um aplicativo. O produto eXtreme Scale suporta os seguintes tipos de pacotes configuráveis:

Pacote configurável do servidor
O pacote configurável do servidor é o arquivo `objectgrid.jar` e é instalado com a instalação do servidor independente do eXtreme Scale e é necessário para executar servidores do eXtreme Scale e também pode ser usado para executar clientes do eXtreme Scale ou caches na memória locais. O ID do pacote configurável para o arquivo `objectgrid.jar` é com.ibm.websphere.xs.server_`<version>`, em que a versão está no formato: `<Version>.<Release>.<Modification>`. Por exemplo, o pacote configurável do servidor para o eXtreme Scale versão 7.1.1 é com.ibm.websphere.xs.server_7.1.1.

Pacote configurável do cliente
O pacote configurável do cliente é o arquivo `ogclient.jar` e é instalado com instalações independentes e do cliente do eXtreme Scale e é usado para executar os clientes do eXtreme Scale ou caches na memória locais. O ID do pacote configurável para o arquivo `ogclient.jar` é com.ibm.websphere.xs.client_`version`, em que a versão está no formato: `<Version>.<Release>.<Modification>`. Por exemplo, o pacote configurável do cliente para o eXtreme Scale versão 7.1.1 é com.ibm.websphere.xs.client_7.1.1.

Limitações

Não é possível reiniciar o pacote configurável do eXtreme Scale porque você não pode reiniciar o object request broker (ORB). Para reiniciar o servidor do eXtreme Scale, você deve reiniciar a estrutura do OSGi.

Visão Geral da Integração de Cache

O elemento crucial que permite ao WebSphere eXtreme Scale executar com tal versatilidade e confiabilidade é seu aplicativo de conceitos de armazenamento em cache para otimizar a persistência e a recoleção de dados em praticamente qualquer ambiente de implementação.

Plug-in do Cache JPA Nível 2 (L2)

O WebSphere eXtreme Scale inclui plug-ins de cache de nível 2 (L2) para ambos os provedores OpenJPA e Hibernate Java Persistence API (JPA). Quando você usar um desses plug-ins, seu aplicativo usará a API do JPA. Uma grade de dados é introduzida entre o aplicativo e o banco de dados, melhorando os tempos de resposta.

Usar o eXtreme Scale como um provedor de cache L2 aumenta o desempenho na leitura e consulta de dados e reduz a carga para o banco de dados. O WebSphere eXtreme Scale tem vantagens sobre as implementações de cache integradas porque o cache é automaticamente replicado entre todos os processos. Quando um cliente armazena em cache um valor, todos os outros clientes conseguem usar o valor armazenado em cache que está localmente na memória.

É possível configurar a topologia e as propriedades para o provedor de cache L2 no arquivo `persistence.xml`. Para obter mais informações sobre como configurar essas propriedades, consulte Propriedades de Configuração do Cache JPA.

Dica: O plug-in do cache JPA L2 requer um aplicativo que usa as APIs do JPA. Se desejar usar as APIs do WebSphere eXtreme Scale para acessar uma origem de dados JPA, use o carregador JPA. Para obter informações adicionais, consulte “Carregadores JPA” na página 66.

Considerações sobre Topologia de Cache do JPA L2

Os fatores a seguir afetam qual tipo de topologia configurar:

1. **Quantos dados você espera que sejam armazenados em cache?**
   - Se os dados puderem ser ajustados em um único heap de JVM, use o “Topologia Integrada” na página 28 ou o “Topologia Intradomínio” na página 27.
   - Se os dados não puderem ser ajustados em um único heap de JVM, use o “Topologia Integrada e Particionada” na página 29 ou o “Topologia Remota” na página 31
2. **Qual é a proporção de "leitura para gravação" esperada?**

   A proporção de "leitura para gravação" afeta o desempenho do cache L2. Cada topologia manipula as operações de leitura e gravação de forma diferente.
   - “Topologia Integrada” na página 28: leitura local, gravação remota
   - “Topologia Intradomínio” na página 27: leitura local, gravação local
   - “Topologia Integrada e Particionada” na página 29: Particionado: leitura remota, gravação remota
   - “Topologia Remota” na página 31: leitura remota, gravação remota.

   Aplicativos que são principalmente somente leitura devem usar topologias integradas e intradomínio quando possível. Aplicativos que executam mais gravação devem usar topologias intradomínio.

3. **Qual é a porcentagem de dados consultados versus localizados por uma chave?**

   Quando ativado, as operações de consulta usam o cache de consulta de JPA. Ative o cache de consulta de JPA somente para aplicativos com altas proporções de leitura para gravação, por exemplo quando você está se aproximando de 99% de operações de leitura. Se você usar o cache de consulta de JPA, deverá usar o "Topologia Integrada" na página 28 ou "Topologia Intradomínio".

   A operação de localização por chave busca uma entidade de destino se a entidade de destino não tem nenhum relacionamento. Se a entidade de destino tiver relacionamentos com o tipo de busca EAGER, esses relacionamentos serão buscados juntamente com a entidade de destino. No cache de dados de JPA, a busca desses relacionamentos faz com que alguns acertos do cache obtam todos os dados de relacionamento.

4. **Qual é o nível de deterioração tolerado dos dados?**

   Em um sistema com algumas JVMs, existe latência de replicação de dados para operações de gravação. O objetivo do cache é manter uma visualização de dados sincronizados final entre todas as JVMs. Quando você estiver usando a topologia intradomínio, existirá um atraso de replicação de dados para operações de gravação. Aplicativos que usam essa topologia devem ser capazes de tolerar leituras antigas e gravações simultâneas que podem sobrescrever os dados.

7.1.1+ Topologia Intradomínio

Com uma topologia intradomínio, os shards primários são colocados em cada servidor de contêiner na topologia. Esses shards primários contêm o conjunto completo de dados para a partição. Qualquer um desses shards primários também podem concluir as operações de gravação em cache. Essa configuração elimina o gargalo na topologia integrada na qual todas as operações de gravação de cache deve passar por um shard primário único.

Em uma topologia intradomínio, nenhum shard de réplica é criado, mesmo se as réplicas tiverem sido definidas em seus arquivos de configuração. Cada shard primário redundante contém uma cópia completa dos dados, de modo que cada shard primário também pode ser considerado como um shard de réplica. Essa configuração usa uma única partição, semelhante à topologia integrada.

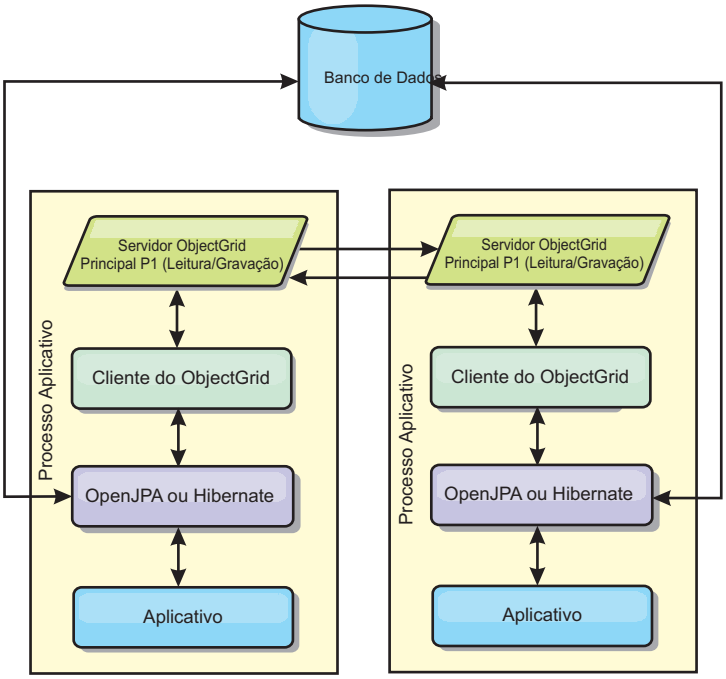


Figura 12. Topologia de Intradomínio do JPA

As propriedades de configuração de cache JPA relacionadas para a topologia intradomínio são:

```
ObjectGridName=objectgrid_name,ObjectGridType=EMBEDDED,PlacementScope=CONTAINER_SCOPE,PlacementScopeTopology=HUB | RING
```

Vantagens:

- As leituras e atualizações do cache serão locais.
- Simples para configurar.

Limitações:

- Essa topologia é mais adequada para quando os servidores de contêiner puderem conter o conjunto inteiro de dados da partição.
- Os shards de réplica, mesmo se estiverem configurados, nunca são colocados porque cada servidor de contêiner hospeda um shard primário. No entanto, todos os shards primários são replicados com os outros shards primários, portanto, esses shards primários se tornam réplicas entre si.

Topologia Integrada

Dica: Considere usar uma topologia intradomínio para obter melhor desempenho.

Uma topologia integrada cria um servidor de contêiner dentro do espaço do processo de cada aplicativo. O OpenJPA e o Hibernate lêem a cópia na memória do cache diretamente e gravam para todas as outras cópias. É possível melhorar o desempenho de gravação usando replicação assíncrona. Esta topologia padrão executa melhor quando a quantidade de dados armazenados em cache é pequena o suficiente para ajustar-se a um único processo. Com uma topologia integrada, crie uma única partição para os dados.

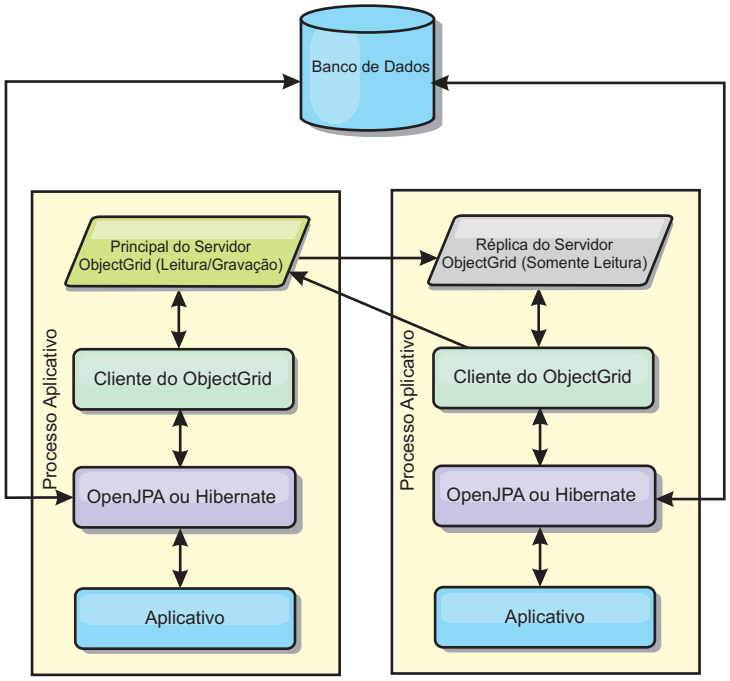


Figura 13. Topologia Integrado do JPA

As propriedades de configuração de cache JPA relacionadas para a topologia integrada são:

```
ObjectGridName=objectgrid_name,ObjectGridType=EMBEDDED,MaxNumberOfReplicas=num_replicas,ReplicaMode=SYNC | ASYNC | NONE
```

Vantagens:

- Todas as leituras de cache são acessos locais rápidos.
- Simples para configurar.

Limitações:

- A quantidade de dados é limitada ao tamanho do processo.
- Todas as atualizações de cache são enviadas por meio de um shard primário, que cria um gargalo.

Topologia Integrada e Particionada

Dica: Considere usar uma topologia intradomínio para obter melhor desempenho.

CUIDADO:
Não use o cache de consulta de JPA com uma topologia particionada integrada. O cache de consulta armazena os resultados da consulta que são uma coleção de chaves de entidade. O cache de consulta vai para o cache de dados para buscar todos os dados da entidade. Como o cache de dados é dividido entre diversos processos, essas chamadas adicionais podem negar os benefícios do cache L2.

Quando os dados em cache são muito grandes para se ajustarem em um único processo, é possível usar a topologia particionada integrada. Esta topologia divide os dados sobre diversos processos. Os dados são divididos entre os shards primários, portanto, cada shard primário contém um subconjunto dos dados. Ainda é possível usar esta opção quando a latência do banco de dados é alta.

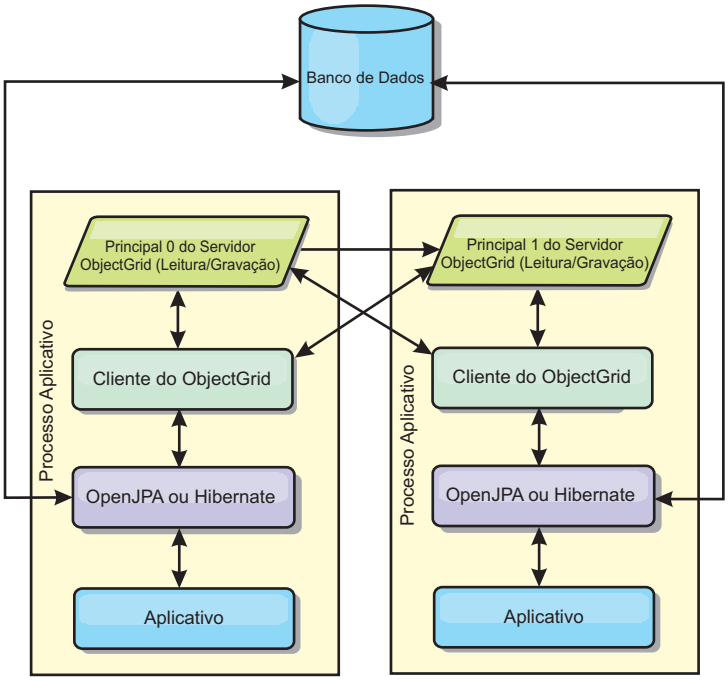


Figura 14. Topologia Particionada Integrada do JPA

As propriedades de configuração de cache do JPA relacionadas à topologia particionada integrada são:

```
ObjectGridName=objectgrid_name,ObjectGridType=EMBEDDED_PARTITION,ReplicaMode=SYNC | ASYNC | NONE,
NumberOfPartitions=num_partitions,ReplicaReadEnabled=TRUE | FALSE
```

Vantagens:

- Armazena grandes quantidades de dados.
- Simples para configurar
- As atualizações de cache são propagadas para vários processos.

Limitação:

- A maioria das leituras e atualizações de cache é remota.

Por exemplo, para armazenar em cache 10 GB de dados com um máximo de 1 GB por JVM, 10 Java Virtual Machines são necessários. Portanto, o número de partições deve ser configurado para 10 ou mais. De maneira ideal, o número de partições deve ser configurado com um número primo no qual cada shard armazena uma quantidade razoável de memória. Geralmente, a configuração de numberOfPartitions é igual ao número de Java Virtual Machines. Com esta configuração, cada JVM armazena uma partição. Se você ativar replicação, será necessário aumentar o número de Java Virtual Machines no sistema. Caso contrário, cada JVM também armazenará uma partição de réplica, que consome tanta memória quanto uma partição primária.

Leia sobre o dimensionamento de memória e o cálculo da contagem de partições no *Guia de Administração* para maximizar o desempenho da sua configuração escolhida.

Por exemplo, em um sistema com quatro Java Virtual Machines e com o valor da configuração de numberOfPartitions de 4, cada JVM hospeda uma partição primária. Uma operação de leitura tem 25% mais chance de buscar dados de uma

partição disponível localmente, o que é muito mais rápido comparado ao obter dados de uma JVM remota. Se uma operação de leitura, como a execução de uma consulta, precisar buscar uma coleta de dados que envolva 4 partições igualmente, 75% das chamadas são remotas e 25% das chamadas são locais. Se a configuração de ReplicaMode for definida para `SYNC` ou `ASYNC` e a configuração de ReplicaReadEnabled for definida para `true`, as quatro partições de réplica são criadas e espalhadas pelos quatro Java Virtual Machines. Cada JVM hospeda uma partição primária e uma partição de réplica. A chance de que a operação de leitura execute localmente aumenta em 50%. A operação de leitura que busca uma coleta de dados que envolve quatro partições igualmente tem somente 50% de chamadas remotas e 50% de chamadas locais. Chamadas locais são muito mais rápidas do que chamadas remotas. Sempre que ocorrem chamada remotas, o desempenho cai.

Topologia Remota

CUIDADO:
Não use o cache de consulta de JPA com uma topologia remota. O cache de consulta armazena os resultados da consulta que são uma coleção de chaves de entidade. O cache de consulta vai para o cache de dados para buscar todos os dados da entidade. Como o cache de dados é remoto, estas chamadas adicionais podem negar os benefícios do cache L2.

Dica: Considere usar uma topologia intradomínio para obter melhor desempenho.

Uma topologia remota armazena todos os dados armazenados em cache em um ou mais processos separados, reduzindo o uso da memória dos processos do aplicativo. É possível obter vantagem da distribuição de dados em processos separados ao implementar uma grade de dados replicada e particionada do eXtreme Scale. Ao contrário das configurações integradas e particionadas integradas descritas nas seções anteriores, se desejar gerenciar a grade de dados remota, você deverá fazer isso independente do aplicativo e do provedor JPA.

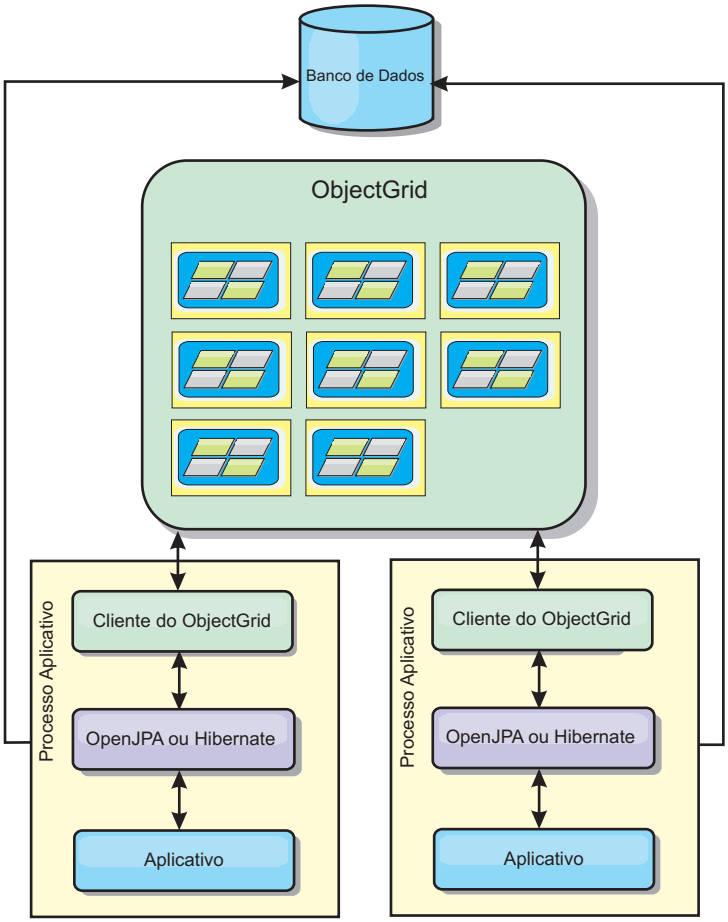


Figura 15. Topologia Remota do JPA

As propriedades de configuração de cache JPA relacionadas à topologia remota são:
`ObjectGridName=`*`objectgrid_name`*`,ObjectGridType=REMOTE`

O tipo de ObjectGrid REMOTE não requer nenhuma configuração de propriedade porque a política de ObjectGrid e de implementação é definida separadamente a partir do aplicativo JPA. O plug-in do cache JPA conecta-se remotamente a um ObjectGrid remoto existente.

Como toda a interação com o ObjectGrid é remota, essa topologia possui o menor desempenho entre todos os tipos de ObjectGrid.

Vantagens:

- Armazena grandes quantidades de dados.
- O processo aplicativo é livre de dados em cache.
- As atualizações de cache são propagadas para vários processos.
- Opções de configuração flexíveis.

Limitação:

- Todas as leituras e atualizações de cache são remotas.

Gerenciando de Sessões HTTP

O gerenciador de replicação de sessão que é fornecido com o WebSphere eXtreme Scale pode trabalhar com o gerenciador de sessões padrão no servidor de aplicativos. Os dados da sessão são replicados de um processo para um outro processo para suportar alta disponibilidade de dados da sessão do usuário.

Características

O gerenciador de sessões foi projetado para que ele possa ser executado em qualquer contêiner do Java Platform, Enterprise Edition Versão 5 ou posterior. Como o gerenciador de sessões não tem nenhuma dependência nas APIs do WebSphere, ele pode suportar várias versões do WebSphere Application Server e também ambientes do servidor de aplicativos do fornecedor.

O gerenciador de sessões HTTP fornece recursos de replicação de sessão para um aplicativo associado. O gerenciador de replicação de sessão funciona com o gerenciador de sessões para o contêiner da web. Juntos, o gerenciador de sessões e o contêiner da web criam sessões HTTP e gerenciam os ciclos de vida de sessões HTTP que estão associadas ao aplicativo. Estas atividades de gerenciamento de ciclo de vida incluem: a invalidação de sessões baseada em um tempo limite ou um servlet explícito ou chamada JavaServer Pages (JSP) e a chamada de listeners de sessão que estão associados com a sessão ou ao aplicativo da web. O gerenciador de sessões persiste suas sessões em uma grade de dados particionados, totalmente replicada e em cluster. O uso do gerenciador de sessões do WebSphere eXtreme Scale permite que o gerenciador de sessões forneça suporte de failover da sessão HTTP quando os servidores de aplicativos são encerrados ou terminam de forma inesperada. O gerenciador de sessões também pode trabalhar em ambientes que não suportam afinidade, quando a afinidade não é forçada por uma camada do balanceador de carga que pulveriza pedidos para a camada do servidor de aplicativos.

Cenários de Uso

O gerenciador de sessões pode ser usado nos seguintes cenários:

- Em ambientes que usam servidores de aplicativos em diferentes versões do WebSphere Application Server, tal como em um cenário de migração.
- Em implementações que utilizam servidores de aplicativos de diferentes fornecedores. Por exemplo, um aplicativo que está sendo desenvolvido em servidores de aplicativos de software livre e que é hospedado no WebSphere Application Server. Outro exemplo é um aplicativo que está sendo promovido do servidor intermediário para produção. A migração contínua destas versões do servidor de aplicativos é possível enquanto todas as sessões HTTP estão ativas e recebendo manutenção.
- Em ambientes que requerem que o usuário persista sessões com níveis de qualidade de serviço (QoS) mais altos. A disponibilidade da sessão é melhor garantida durante failover do servidor do que em níveis padrão de QoS do WebSphere Application Server.
- Em um ambiente no qual a afinidade de sessão não pode ser garantida ou ambientes nos quais a afinidade é mantida por um balanceador de carga do fornecedor. Com um balanceador de carga do fornecedor, o mecanismo de afinidade deve ser customizado com esse balanceador de carga.
- Em qualquer ambiente para transferir o processamento requerido para gerenciamento e armazenamento de sessões para um processo Java externo.
- Em várias células para ativar failover de sessão entre células.

- Em datacenters múltiplos ou zonas múltiplas.

Como o Gerenciador de Sessões Funciona

O gerenciador de replicação de sessão usa um listener de sessão para atender às mudanças de dados da sessão. O gerenciador de replicação de sessão persiste os dados da sessão para uma instância do ObjectGrid local ou remotamente. É possível incluir o listener de sessão e o filtro de servlet em cada módulo da web em seu aplicativo com o conjunto de ferramentas que é fornecido com o WebSphere eXtreme Scale. Você também pode incluir manualmente esses listeners e filtros no descritor de implementação da web de seu aplicativo.

Este gerenciador de replicação de sessão trabalha com cada gerenciador de sessões de contêiner da web do fornecedor para replicar dados da sessão entre Java virtual machines. Quando o servidor original para de funcionar, os usuários podem recuperar dados da sessão de outros servidores.

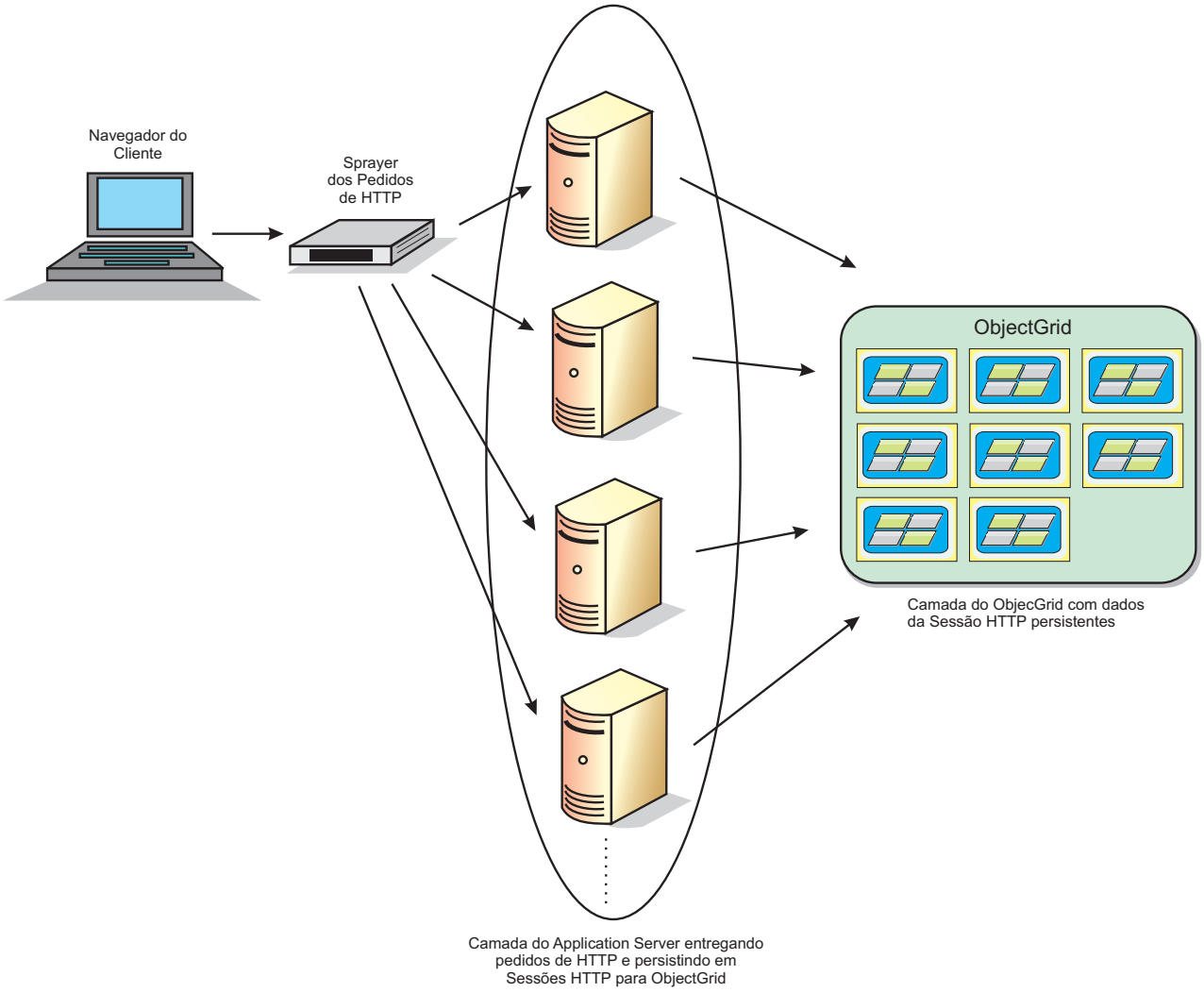


Figura 16. Topologia do Gerenciamento de Sessões HTTP com uma Configuração de Contêiner Remoto

Topologias de Implementação

O gerenciador de sessões pode ser configurado utilizando dois diferentes cenários de implementação dinâmicos:

Servidores de contêiner do eXtreme Scale conectados em rede, integrados
Neste cenário, os servidores eXtreme Scale são colocados nos mesmos processos que os servlets. O gerenciador de sessões pode ser comunicar diretamente com a instância do local, evitando atrasos de rede caros. Este cenário é preferível ao executar com afinidade e o desempenho for crítico.

Servidores de contêiner do eXtreme Scale conectados em rede, remotos
Neste cenário, os servidores do eXtreme Scale são executados em processos externos a partir do processo no qual os servlets são executados. O gerenciador de sessões se comunica com uma grade do servidor eXtreme Scale remoto. Este cenário é preferível quando a camada do contêiner da web não tem a memória para armazenar os dados da sessão. Os dados da sessão são transferidos para uma camada separada, o que resulta em uso de memória inferior na camada do contêiner da web. A latência mais alta ocorre porque os dados estão em um local remoto.

Inicialização do Contêiner Integrado Genérico

O eXtreme Scale inicia automaticamente um contêiner do ObjectGrid integrado dentro de qualquer processo do servidor de aplicativos quando o contêiner da web inicializa o listener de sessão ou o filtro de servlet, se a propriedade do objectGridType é configurada como `EMBEDDED`. Veja detalhes em Parâmetros de inicialização do contexto do servlet.

Não é necessário empacotar um arquivo `ObjectGrid.xml` e um arquivo `objectGridDeployment.xml` em seu arquivo WAR ou EAR do aplicativo da web. Os arquivos `ObjectGrid.xml` e `objectGridDeployment.xml` padrão são empacotados no arquivo JAR do produto. Mapas dinâmicos são criados para vários contextos de aplicativos da web por padrão. Mapas estáticos do eXtreme Scale continuam a ser suportados.

Esta abordagem para iniciar contêineres do ObjectGrid integrados é aplicada a qualquer tipo de servidor de aplicativos. As abordagens envolvendo um componente do WebSphere Application Server ou GBean do WebSphere Application Server Community Edition são reprovadas.

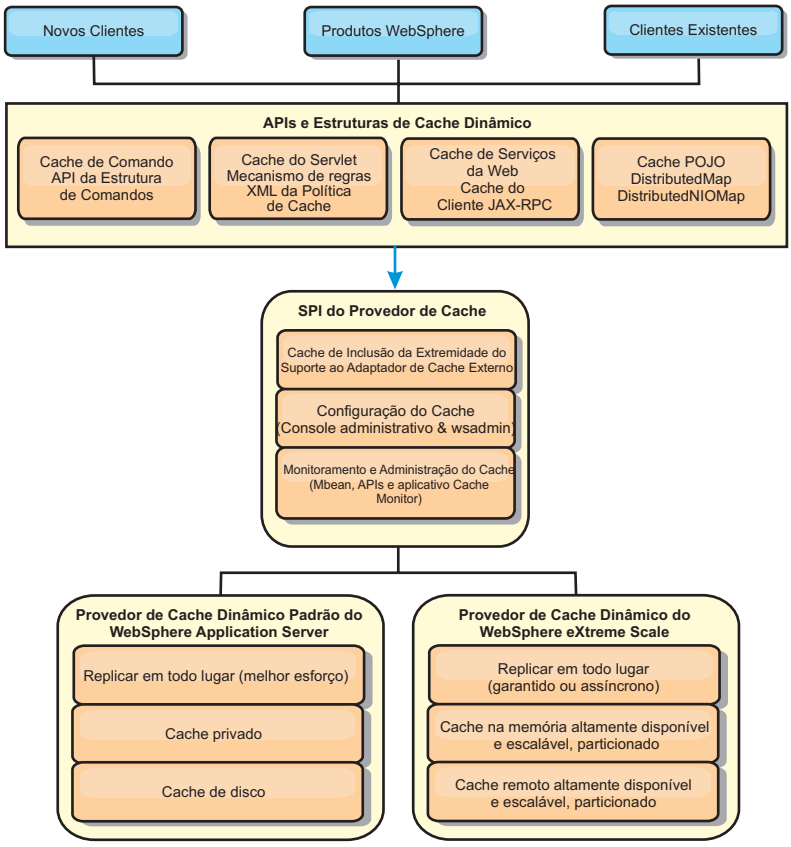
Provedor de Cache Dinâmico

A API de Cache Dinâmico está disponível para os aplicativos Java EE que são implementados no WebSphere Application Server. O provedor de cache dinâmico pode ser potencializado para dados de negócios em cache, HTML gerado ou para sincronizar os dados em cache na célula utilizando o data replication service (DRS).

Visão Geral

Previamente, o único provedor de serviços para a API de Cache Dinâmico era o mecanismo de cache dinâmico padrão construído dentro doWebSphere Application Server. Os clientes podem usar a interface do provedor de serviço de cache dinâmico no WebSphere Application Server para plugar o eXtreme Scale no cache dinâmico. Ao configurar essa capacidade, é possível ativar aplicativos que foram gravados com a API de Cache Dinâmico ou os aplicativos usando o

armazenamento em cache de nível do contêiner (como servlets) para alavancar os recursos e as capacidades de desempenho do WebSphere eXtreme Scale.



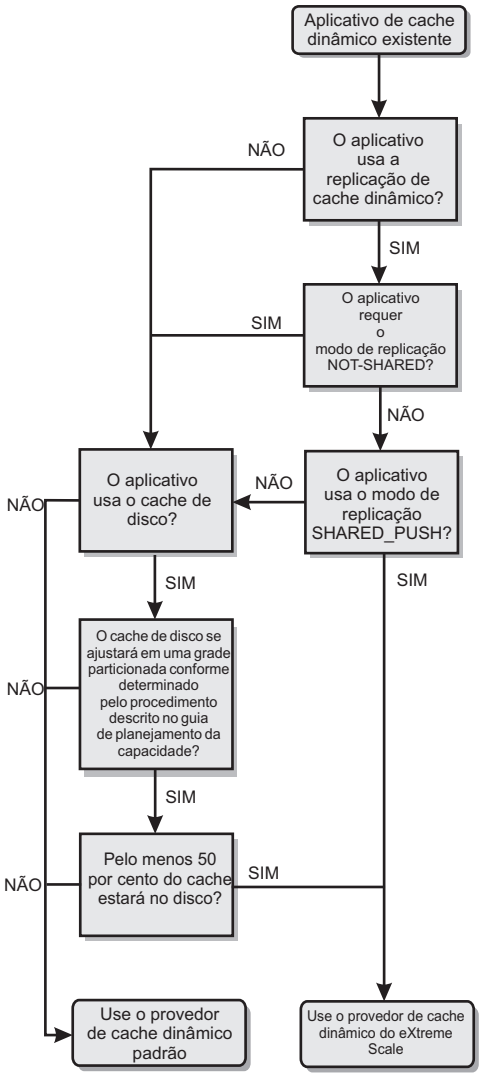
É possível instalar e configurar o provedor de cache dinâmico conforme descrito em Configurando o Provedor de Cache Dinâmico para o WebSphere eXtreme Scale.

Decidindo como Potencializar o WebSphere eXtreme Scale

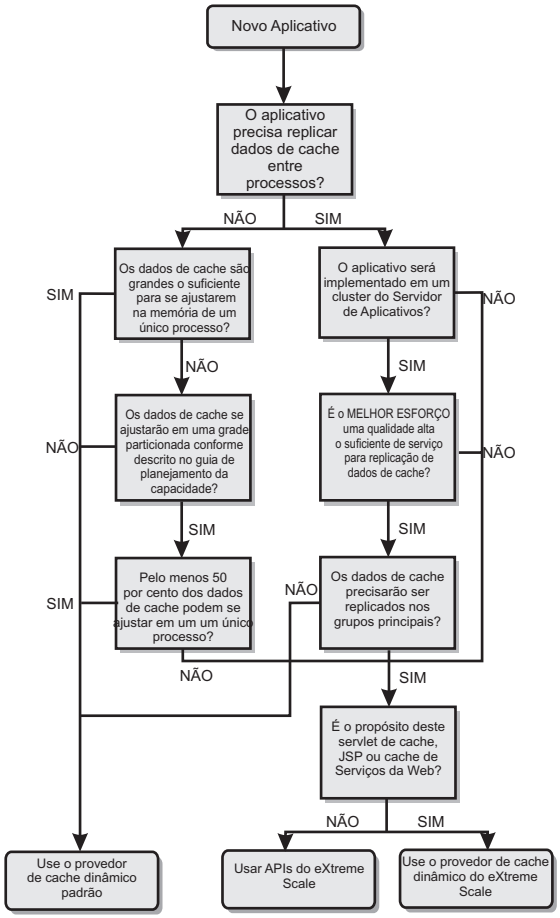
Os recursos disponíveis noWebSphere eXtreme Scale aumentam significantemente os recursos distribuídos da API do cache dinâmico além do que é oferecido pelo mecanismo de cache dinâmico e serviço de replicação de dados. Com o eXtreme Scale, é possível criar caches que são verdadeiramente distribuídos entre múltiplos servidores, em em de simplesmente replicados e sincronizados entre os servidores. Também, os caches do eXtreme Scale são transacionais e altamente disponíveis, garantindo que cada servidor veja o mesmo conteúdo para o serviço de cache dinâmico. O WebSphere eXtreme Scale oferece um qualidade de serviço mais alta para replicação de cache do que o DRS.

Porém, essas vantagens não significam que o provedor de cache dinâmico doeXtreme Scale seja a escolha certa para cada aplicativo. Use as árvores de decisão e matriz e comparação de recursos abaixo para determinar qual tecnologia se encaixa melhor no seu aplicativo.

Árvore de Decisão para Migrar Aplicativos de Cache Dinâmico Existente



Árvore de Decisão para Escolher um Provedor de Cache para Novos Aplicativos



Comparação de Recursos

Tabela 1. Comparação de Recursos

Recursos do cache	Provedor padrão	Provedor do eXtreme Scale	eXtreme Scale API
Cache na memória, local	x	x	x
Armazenamento em cache distribuído	Integrado	Integrado, particionado integrado e particionado remoto	Múltiplo
Linearmente escalável		x	x
Replicação confiável (síncrona)		ORB	ORB
Estouro de disco	x		
Despejo	LRU/TTL/baseado em heap	LRU/TTL (por partição)	Múltiplo
Invalidação	x	x	x

Tabela 1. Comparação de Recursos (continuação)

Recursos do cache	**Provedor padrão**	**Provedor do eXtreme Scale**	**eXtreme Scale API**
Relacionamentos	IDs de dependência, modelos	IDs de dependência, modelos	x
Consultas sem chave			Consulta e índice
Integração de backend			Utilitários de Carga
Transacional		Implícito	x
Armazenamento baseado em chave	x	x	x
Eventos e listeners	x	x	x
Integração do WebSphere Application Server	Somente célula única	Célula múltipla	Célula independente
Suporte à Java Standard Edition		x	x
Monitoramento e estatística	x	x	x
Segurança	x	x	x

Tabela 2. Integração de Tecnologia Transparente

Recursos do cache	**Provedor padrão**	Provedor do **eXtreme Scale**	**eXtreme Scale API**
Armazenamento em cache dos resultados do servlet/JSP do WebSphere Application Server	V5.1+	V6.1.0.25+	
Armazenamento em cache do resultado do WebSphere Application Server Web Services (JAX-RPC)	V5.1+	V6.1.0.25+	
Armazenamento em cache de sessão HTTP			x
Provedor de cache para OpenJPA e Hibernate			x
Sincronização de banco de dados usando OpenJPA e Hibernate			x

Tabela 3. Interfaces de Programação

Recursos do cache	Provedor padrão	Provedor do eXtreme Scale	eXtreme Scale API
API baseada em comandos	API da estrutura de comandos	API da estrutura de comandos	API de DataGrid
API baseada em mapa	API de DistributedMap	API de DistributedMap	API do ObjectMap
API do EntityManager			x

Para obter uma descrição mais detalhada sobre como os caches distribuídos do eXtreme Scale funcionam, consulte o informações de configuração de implementação no *Guia de Administração*.

Nota: Um cache distribuído do eXtreme Scale somente pode armazenar entradas nas quais ambos, a chave e o valor, implementam a interface java.io.Serializable.

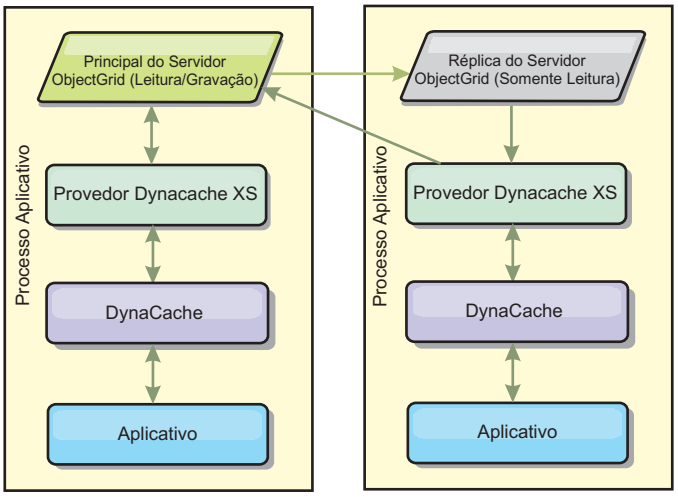
Tipos de topologia

Um serviço de cache dinâmico criado com o provedor doeXtreme Scale pode ser implementado em qualquer uma de três topologias disponíveis, permitindo que você padronize o cache especificamente para desempenho, recurso e necessidades administrativas. Essas topologias são integradas, particionadas integradas e remotas.

Topologia Integrada

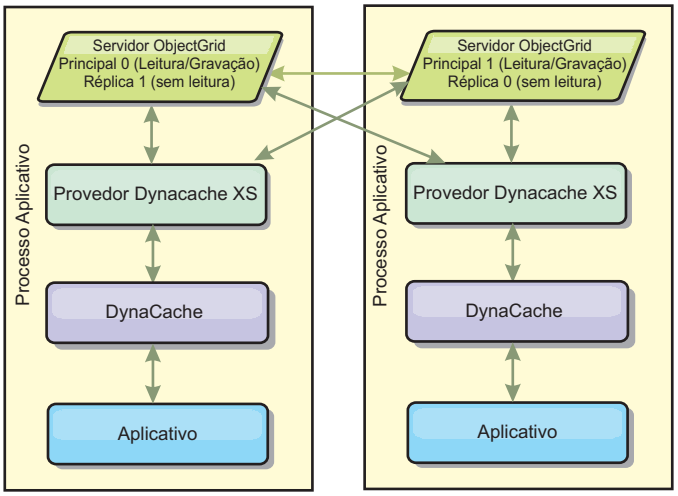
A topologia integrada é similar ao cache dinâmico padrão e ao provedor DRS. Instâncias de cache distribuído criados com a topologia integrada mantêm uma cópia integral do cache dentro de cada processo doeXtreme Scale que acessa o serviço de cache dinâmico, permitindo que todas as operações de leitura ocorram localmente. Todas as operações de gravação passam por um processo de servidor único, no qual os bloqueios transacionais são gerenciados, antes de serem replicados para o restante dos servidores. Consequentemente, esta topologia é melhor para cargas de trabalho nas quais as operações de leitura de cache excedem grandemente as operações de gravação em cache.

Com a topologia integrada, entradas de cache novas e atualizadas não são imediatamente visíveis em cada processo de servidor único. Uma entrada de cache não estará visível, mesmo para o servidor que a gerou, até ser propagada por meio dos serviços de replicação assíncrona do WebSphere eXtreme Scale. Esses serviços operam tão rapidamente quanto o hardware permitir, mas ainda há um pequeno atraso. A topologia integrada é mostrada na seguinte imagem:



Topologia particionada integrada

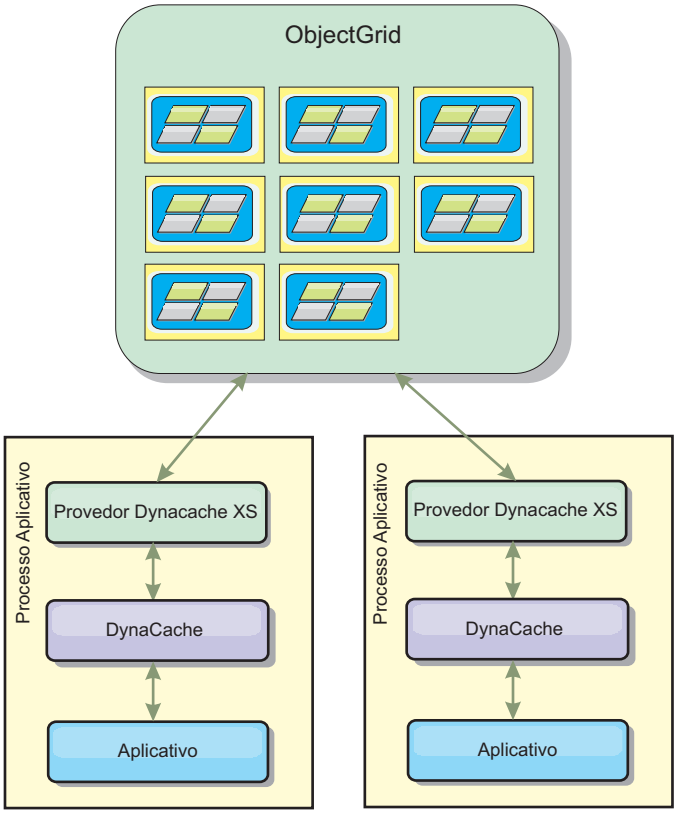
Para cargas de trabalho nas quais as gravações em cache ocorrem tão frequentemente quanto ou mais frequentemente que as leituras, as topologias remotas ou particionadas integradas são recomendadas. A topologia particionada integrada mantém todos os dados do cache dentro dos processos do WebSphere Application Server que acessam o cache. Porém, cada processo somente armazena uma parte dos dados do cache. Todas as leituras e gravações para os dados localizados nesta "partição" passam pelo processo, o que significa que a maioria dos pedidos para o cache será preenchido com uma chamada de procedimento remoto. Isto resulta em uma latência mais alta para operações de leitura do que a topologia integrada, mas a capacidade do cache distribuído de manipular operações de leitura e gravação escalarão linearmente com o número de processos doWebSphere Application Server que acessa o cache. Também, com esta topologia, o tamanho máximo do cache não é limitado pelo tamanho de um único processo doWebSphere. Porque cada processo somente retém uma parte do cache, o tamanho de cache máximo se torna o tamanho agregado de todos os processos, menos o gasto adicional do processo. A topologia particionada integrada é mostrada na seguinte imagem:



Por exemplo, suponha que você tem uma grade de processos do servidor com 256 megabytes de heap livre em cada um deles para hospedar um serviço de cache dinâmico. O provedor de cache dinâmico padrão e o provedor do eXtreme Scale que usa a topologia integrada seriam, ambos, limitados a um tamanho de cache de memória de 256 megabytes menos o gasto adicional. Consulte a seção Planejamento de Capacidade e Alta Disponibilidade, posteriormente neste documento. O provedor doeXtreme Scale que usa a topologia particionada integrada seria limitado a um tamanho de cache de um gigabyte menos o gasto adicional. Desta maneira, o provedor do WebSphere eXtreme Scale possibilita ter serviços de cache dinâmico na memória maiores que o tamanho de um único processo do servidor. O provedor de cache dinâmico padrão conta com o uso de um cache de disco para permitir que instâcias de cache cresçam além do tamanho de um processo único. Em muitas situações, o provedor doWebSphere eXtreme Scale pode eliminar a necessidade de um cache de disco e os dispendiosos sistemas de armazenamento em disco necessários para fazê-los executar.

Topologia Remota

A topologia remota também pode ser usada para eliminar a necessidade de um cache de disco. A única diferença entre as topologias remotas e particionadas integradas é que todas os dados em cache são armazenados fora dos processos do WebSphere Application Server quando você está usando a topologia remota. O WebSphere eXtreme Scale suporta processos de contêiner independentes para dados do cache. Esses processos de contêiner têm um gasto adicional mais baixo do que um processo do WebSphere Application Server e também não são limitados ao uso de uma Java Virtual Machine (JVM) particular. Por exemplo, os dados para um serviço de cache dinâmico que está sendo acessado por um processo do WebSphere Application Server de 32 bits poderiam ser alocados em um processo de contêiner do eXtreme Scale que estivesse executando em uma JVM de 64 bits. Isso permite aos usuários potencializar a capacidade de memória aumentada de processos de 64 bits para armazenamento em cache, sem incorrer um gasto adicional de 64 bits para processos do servidor de aplicativos. A topologia remota é mostrada na seguinte imagem:



Compactação de dados

Outro recurso de desempenho oferecido pelo provedor de cache dinâmico do WebSphere eXtreme Scale que pode ajudar os usuários a gerenciar o gasto adicional de cache é a compactação. O provedor de cache dinâmico padrão não permite compactação de dados em cache na memória. Com o provedor doeXtreme Scale, isso se torna possível. A compactação de cache que usa o algoritmo de deflação pode ser ativada em qualquer uma das três topologias distribuídas. Ativar a compactação aumentará o gasto adicional para operações de leitura e gravação, mas aumentará drasticamente a densidade do cache para aplicações como armazenamento em cache de servlet e JSP.

Cache de Memória Local

O provedor de cache dinâmico do WebSphere eXtreme Scale também pode ser usado para retornar as instâncias do cache dinâmico que têm a **replicação desativada**. Como o provedor de cache dinâmico padrão, esses caches podem armazenar dados não serializáveis. Eles também podem oferecer melhor desempenho que o provedor de cache dinâmico padrão em servidores corporativos de multiprocessador grande porque o caminho do código do eXtreme Scale é projetado para maximizar a simultaneidade do cache em memória.

Mecanismo de Cache Dinâmico e Diferenças Funcionais do eXtreme Scale

No caso de caches em memória locais nos quais a replicação está desativada, não deve haver nenhuma diferença funcional apreciável entre os caches retornados pelo provedor de cache dinâmico padrão e o WebSphere eXtreme Scale. Os usuários não devem observar uma diferença funcional entre os dois caches, exceto que os caches

retornados pelo WebSphere eXtreme Scale não suportam a transferência de disco ou estatísticas e operações relacionadas ao tamanho do cache em memória.

No caso de caches nos quais a replicação está ativada, não deve haver nenhuma diferença funcional apreciável nos resultados retornados pela maioria das chamadas de API de Cache Dinâmico, independentemente de o cliente estar usando o provedor de cache dinâmico padrão ou o provedor de cache dinâmico do eXtreme Scale. Para algumas operações não é possível emular o comportamento do mecanismo de cache dinâmico usando o eXtreme Scale.

Estatísticas do Cache Dinâmico

As estatísticas de cache dinâmico são relatadas por meio do aplicativo CacheMonitor ou do MBean de cache dinâmico. Quando o provedor de cache dinâmico do eXtreme Scale for usado, as estatísticas ainda serão reportadas através dessas interfaces, mas o contexto dos valores estatísticos serão diferente.

Se uma instância do cache dinâmico é compartilhada entre três servidores nomeados A, B e C, então o objeto das estatísticas do cache dinâmico somente retorna estatísticas para a cópia do cache no servidor no qual a chamada é feita. Se as estatísticas são recuperadas no servidor A, elas refletem somente a atividade no servidor A.

Com o eXtreme Scale, existe somente um único cache distribuído compartilhado entre todos os servidores, assim, é impossível controlar a maioria das estatísticas em uma base servidor-a-servidor como o provedor de cache dinâmico padrão faz. Uma lista das estatísticas reportadas pela API de Estatísticas de Cache e o que elas representam quando você está usando o provedor de cache dinâmico do WebSphere eXtreme Scale estão a seguir. Como o provedor padrão, essas estatísticas não são sincronizadas e, portanto, podem variar até 10% para cargas de trabalho simultâneas.

- **Acessos ao Cache** : Os acessos ao cache são controlados por servidor. Se o tráfego no Servidor A gerar 10 acessos ao cache e o tráfego no Servidor B gerar 20 acessos ao cache, as estatísticas do cache relatarão 10 acessos ao cache no Servidor A e 20 acessos ao cache no Servidor B.
- **Perdas de Acertos no Cache**: As perdas de acerto no cache são controladas por servidor assim como os acessos ao cache.
- **Entradas do Cache de Memória**: Esta estatística relata o número de entradas de cache no cache distribuído. Cada servidor que acessa o cache relatará o mesmo valor para esta estatística, e esse valor será o número total de entradas do cache de memória sobre todos os servidores.
- **Tamanho do Cache de Memória em MB**: Esta métrica é suportada apenas por caches que usam as topologias remota, integrada ou integrada_particionada. Ela relata o número de megabytes do espaço de heap Java consumido pelo cache, na grade inteira. Esta estatística relata o uso de heap apenas para as partições primárias; você deve levar as réplicas em conta. Como a configuração padrão para as topologias remota e integrada_particionada é uma réplica assíncrona, dobre este número para obter o consumo real de memória do cache.
- **Remoções do Cache**: Esta estatística relata o número total de entradas removidas do cache por qualquer método, e é um valor agregado para o cache distribuído inteiro. Se o tráfego no Servidor A gerar 10 invalidações e o tráfego no Servidor B gerar 20 invalidações, então o valor em ambos os servidores será 30.

- **Remoções de Cache Menos Usado Recentemente (LRU)**: Esta estatística é agregada, como as remoções de cache. Ele controla o número de entradas que foram removidas para manter o cache sob seu tamanho máximo.
- **Invalidações de Tempo Limite**: Esta também é uma estatística agregada, e ela controla o número de entradas que foram removidas devido a tempo limite.
- **Invalidações Explícitas**: Também uma estatística agregada, ela controla o número de entradas que foram removidas com invalidação direta por chave, ID de dependência ou modelo.
- **Estatísticas Estendidas** : O provedor de cache dinâmico do eXtreme Scale exporta as seguintes cadeias de chave de estatística estendida.
  - **com.ibm.websphere.xs.dynacache.remote_hits**: O número total de acessos ao cache controlados no contêiner do eXtreme Scale. Esta é uma estatística agregada, e seu valor no mapa de estatísticas estendidas é um `longo`.
  - **com.ibm.websphere.xs.dynacache.remote_misses**: O número total de perdas de acerto no cache controladas no contêiner do eXtreme Scale. Uma estatística agregada, seu valor no mapa de estatísticas estendidas é um `longo`.

Relatando Estatísticas de Reconfiguração

O provedor de cache dinâmico permite reconfigurar as estatísticas de cache. Com o provedor padrão a operação de reconfiguração somente limpa as estatísticas no servidor afetado. O provedor de cache dinâmico do eXtreme Scale controla a maioria de seus dados estatísticos nos contêineres de cache remoto. Estes dados não são limpos ou alterados quando as estatísticas são reconfiguradas. Em vez disso, o comportamento do cache dinâmico padrão é simulado no cliente por meio do relato da diferença entre o valor atual de uma determinada estatística e o valor dessa estatística na última vez que a reconfiguração foi chamada nesse servidor.

Por exemplo, se o tráfego no Servidor A gerar 10 remoções de cache, as estatísticas no Servidor A e no Servidor B relatarão 10 remoções. Agora, se as estatísticas no Servidor B são reconfiguradas e o tráfego no Servidor A gerar 10 remoções adicionais, as estatísticas no Servidor A relatarão 20 remoções e as estatísticas no Servidor B relatarão 10 remoções.

Eventos do Cache Dinâmico

A API do Cache Dinâmico permite aos usuários registrar listeners de evento. Quando estiver usando o eXtreme Scale como o provedor de cache dinâmico, os listeners de evento funcionarão como esperado para caches na memória local.

Para caches distribuídos, o comportamento de evento dependerá da topologia que estiver sendo usada. Para caches que usam a topologia integrada, os eventos serão gerados no servidor que manipula as operações de gravação, também conhecidos como o shard primário. Isso significa que somente um servidor receberá notificações de evento, mas ele terá todas as notificações de evento normalmente esperadas do provedor de cache dinâmico. Porque o WebSphere eXtreme Scale escolhe o shard primário no tempo de execução, é impossível garantir que um processo de servidor particular sempre receba esses eventos.

Caches particionados integrados gerarão eventos em qualquer servidor que hospede uma partição do cache. Portanto, se um cache tiver 11 partições e cada servidor em uma grade do WebSphere Application Server Network Deployment de 11 servidores hospedar uma das partições, cada servidor receberá os eventos do cache dinâmico para as entradas de cache que hospedar. Nenhum processo de servidor único veria todos os eventos a menos que todas as 11 partições estivessem

hospedadas nesse processo de servidor. Assim como ocorre com a topologia integrada, é impossível garantir que um processo de servidor particular receberá um conjunto particular de eventos ou quaisquer eventos.

Os caches que usam a topologia remota não suportam eventos de cache dinâmico.

Chamadas de MBean

O provedor de cache dinâmico do WebSphere eXtreme Scale não suporta o armazenamento em disco. Quaisquer chamadas de MBean relacionadas ao armazenamento em disco não funcionarão.

Mapeamento da Política de Replicação de Cache Dinâmico

O provedor de cache dinâmico integrado do WebSphere Application Server suporta múltiplas políticas de replicação de cache. Essas políticas podem ser configuradas globalmente ou em cada entrada de cache. Consulte a documentação de cache dinâmico para obter umadescrição dessas políticas de replicação.

O provedor de cache dinâmico do eXtreme Scale não segue essas políticas diretamente. As características de replicação de um cache são determinadas pelo tipo de topologia distribuídas do eXtreme Scale configurado e se aplicam a todos os valores colocados neste cache, independentemente do conjunto de políticas de replicação na entrada pelo serviço de cache dinâmico. A seguir há uma lista de todas as políticas de replicação suportadas pelo serviço de cache dinâmico e a ilustração de qual topologia do eXtreme Scale fornece características de replicação similares.

Note que o provedor de cache dinâmico do eXtreme Scale ignora as configurações da política de replicação DRS em um cache ou entrada de cache. Os usuários devem escolher a topologia que se adéqua às suas necessidades de replicação.

- NOT_SHARED – atualmente nenhuma das topologias fornecidas pelo provedor de cache dinâmico do eXtreme Scale consegue se aproximar dessa política. Isso significa que todos os dados armazenados no cache devem ter chaves e valores que implementem java.io.Serializable.
- SHARED_PUSH – A topologia integrada se aproxima dessa política de replicação. Quando uma entrada de cache é criada, ela é replicada para todos os servidores. Os servidores somente procuram entradas de cache localmente. Se uma entrada não é localizada localmente, presume-se que ela não existe e os outros servidores não são consultados quanto a ela.
- SHARED_PULL and SHARED_PUSH_PULL – As topologias remotas e particionadas se aproximam desta política de replicação. O estado distribuído do cache é completamente consistente entre todos os servidores.

Estas informações são fornecidas principalmente para que você possa se certificar de que a topologia satisfaz suas necessidades de consistência distribuída. Por exemplo, se a topologia integrada é uma melhor escolha para as suas necessidades de desempenho e implementação, mas você precisa do nível da consistência de cache fornecido por SHARED_PUSH_PULL, então considere usar a particionada integrada, ainda que o desempenho possa ser ligeiramente mais lento.

Segurança

Você pode proteger as instâncias de cache dinâmico que estão executando em topologias particionadas integradas ou topologias integradas com a funcionalidade

de segurança compilada no WebSphere Application Server. Consulte a documentação em Protegendo servidores de aplicativos no WebSphere Application Server Centro de Informações.

Quando um cache está executando em topologia remota, é possível para um cliente eXtreme Scale independente se conectar ao cache e afetar o conteúdo da instância do cache dinâmico. O provedor de cache dinâmico do eXtreme Scale tem um recurso de criptografia de gasto adicional baixo que pode evitar que os dados do cache sejam lidos ou alterados por clientes não-WebSphere Application Server. Para ativar esse recurso, configure o parâmetro opcional **com.ibm.websphere.xs.dynacache.encryption_password** para o mesmo valor em cada instância do WebSphere Application Server que acesse o provedor de cache dinâmico. Isso irá criptografar o valor e os metadados do usuário para o CacheEntry usando criptografia AES de 128 bits. É muito importante que o mesmo valor seja configurado em todos os servidores. Os servidores não poderão ler dados colocados em cache por servidores com um valor diferentes para este parâmetro.

Se o provedor do eXtreme Scale detectar que diferentes valores estão configurados para esta variável no mesmo cache, ele gera um aviso no log do processo do contêiner do eXtreme Scale.

Consulte a documentação do eXtreme Scale no "Visão Geral de Segurança" na página 132 se autenticação SSL ou do cliente for necessária.

Informações adicionais

- Redbook do Cache Dinâmico
- Documentação do Cache Dinâmico
  - WebSphere Application Server 7.0
  - WebSphere Application Server 6.1
- Documentação do DRS
  - WebSphere Application Server 7.0
  - WebSphere Application Server 6.1

Integração com o Banco de Dados: Armazenamento em Cache Write-behind, Sequencial e Lateral

O WebSphere eXtreme Scale é usado para colocar um banco de dados tradicional na frente e eliminar a atividade de leitura que normalmente é armazenada no banco de dados. Um cache coerente pode ser utilizado com um aplicativo direta ou indiretamente, utilizando um mapeador relacional de objeto. O cache coerente pode transferir o banco de dados ou o backend a partir das leituras. Em um cenário levemente mais complexo, tal como o acesso transacional a um conjunto de dados no qual apenas parte dos dados requer garantias de persistência tradicional, a filtragem pode ser utilizada para transferir até mesmo transações de gravação.

É possível configurar o WebSphere eXtreme Scale para funcionar como um espaço de processamento de banco de dados em memória altamente flexível. Entretanto, o WebSphere eXtreme Scale não é um object relational mapper (ORM). Ele não reconhece de onde vieram os dados na grade de dados. Um aplicativo ou um ORM pode colocar dados em um servidor eXtreme Scale. É responsabilidade da origem dos dados certificar-se de que eles permaneçam consistentes com o banco de dados no qual os dados se originaram. Isto significa que o eXtreme Scale não pode invalidar dados que são extraídos de um banco de dados automaticamente.

O aplicativo ou mapeador deve fornecer esta função e gerenciar os dados armazenados no eXtreme Scale.

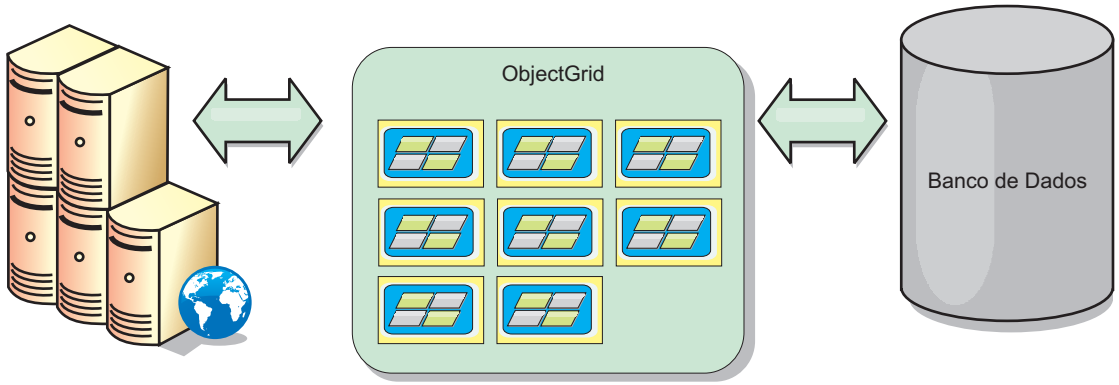


Figura 17. ObjectGrid como um Buffer de Banco de Dados

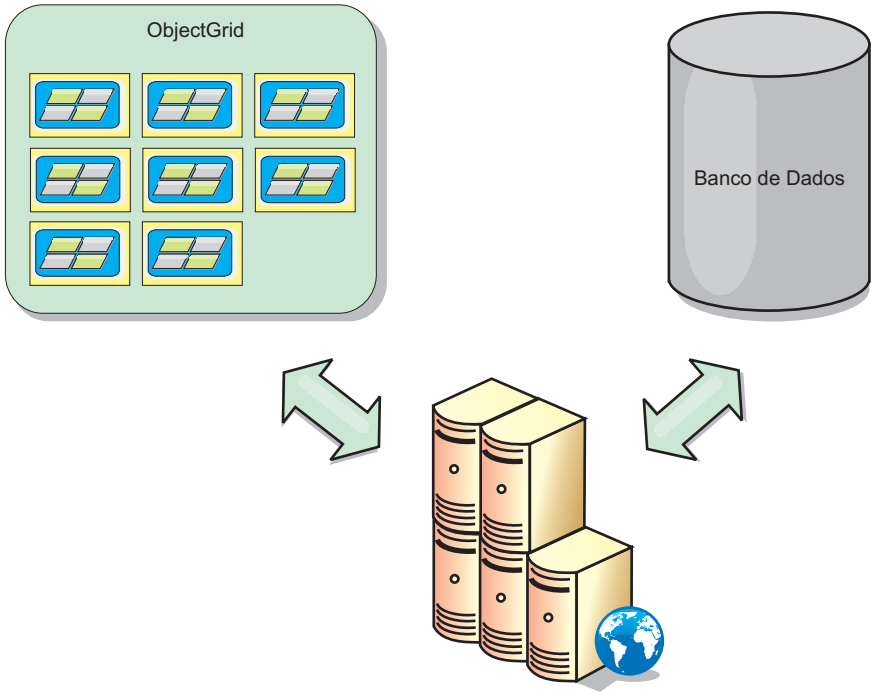


Figura 18. ObjectGrid como um Cache Secundário

Cache Disperso e Completo

O WebSphere eXtreme Scale pode ser utilizado como um cache disperso ou um cache completo. Um cache disperso mantém apenas um subconjunto do total de dados, enquanto que um cache completo mantém todos os dados. Ele também pode ser preenchido gradualmente, conforme os dados são necessários. Os caches dispersos normalmente são acessados usando chaves (ao invés de índices ou consultas) porque os dados estão disponíveis apenas parcialmente.

Cache Disperso

Quando uma chave não está presente em um cache disperso, ou os dados não estão disponíveis e uma falta de cache ocorre, a próxima camada é chamada. Os dados são buscados, a partir de um banco de dados, por exemplo, e inseridos na

camada de cache da grade de dados. Se estiver usando uma consulta ou um índice, apenas os valores atualmente carregados serão acessados e as solicitações não serão encaminhadas para as outras camadas.

Cache Completo

Um cache completo contém todos os dados necessários e pode ser acessado usando atributos não-chaves com índices ou consultas. Um cache completo é pré-carregado com dados a partir do banco de dados antes que o aplicativo tente acessar os dados. Um cache completo pode funcionar como uma substituição do banco de dados após os dados serem carregados. Como todos os dados estão disponíveis, as consultas e índices podem ser usados para localizar e agregar dados.

Cache Secundário

Quando o WebSphere eXtreme Scale é usado como um cache secundário, o backend é usado com a grade de dados.

Cache Secundário

É possível configurar o produto como um cache secundário para a camada de acesso a dados de um aplicativo. Neste cenário, o WebSphere eXtreme Scale é utilizado para armazenar temporariamente objetos que normalmente poderiam ser recuperados de um banco de dados de backend. Aplicativos verificam se a grade de dados contém os dados. Se os dados estiverem na grade de dados, eles serão retornados para o responsável pela chamada. Se os dados não existirem, eles serão recuperados a partir do banco de dados de backend. Os dados são então inseridos na grade de dados para que a próxima solicitação possa usar a cópia em cache. O diagrama a seguir ilustra como o WebSphere eXtreme Scale pode ser usado como um cache secundário com uma camada de acesso a dados arbitrários, como OpenJPA ou Hibernate.

Plug-ins do cache secundário para Hibernate e OpenJPA

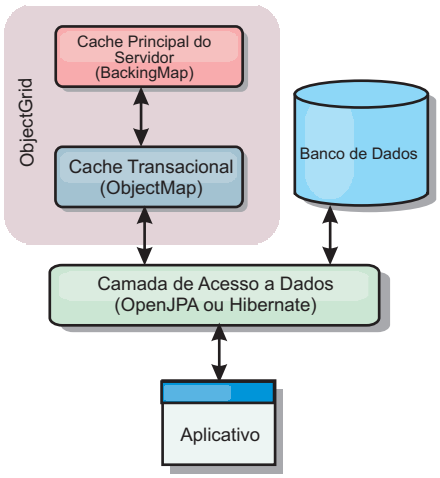


Figura 19. Cache Secundário

Os plug-ins de cache para ambos OpenJPA e Hibernate são incluídos no WebSphere eXtreme Scale, o que permite usar o produto como um cache secundário automático. Usar o WebSphere eXtreme Scale como um provedor de cache aumenta o desempenho ao ler e enfileirar dados e reduz a carga para o banco de dados. Existem vantagens que o WebSphere eXtreme Scale tem sobre as

implementações de cache integrado porque o cache é automaticamente replicado entre todos os processos. Quando um cliente armazena em cache um valor, todos os outros clientes podem usar o valor em cache.

Cache Sequencial

É possível configurar em cache sequencial para um backend de banco de dados ou como um cache secundário para um banco de dados. O armazenamento em cache sequencial utiliza o eXtreme Scale como o meio principal de interação com os dados. Quando o eXtreme Scale é usado como um cache sequencial, o aplicativo interage com o backend usando um plug-in Loader.

Cache Sequencial

Quando usado como um cache sequencial, o WebSphere eXtreme Scale interage com o backend usando um plug-in Loader. Este cenário pode simplificar o acesso a dados porque os aplicativos podem acessar as APIs do eXtreme Scale diretamente. Vários cenários de armazenamento em cache diferentes são suportados no eXtreme Scale para garantir que os dados no cache e os dados no backend sejam sincronizados. O diagrama a seguir ilustra como um cache sequencial interage com o aplicativo e o back end.

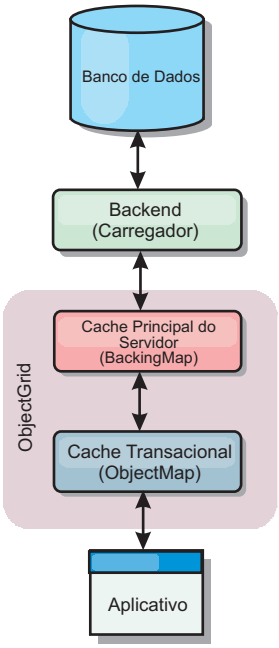


Figura 20. Cache Sequencial

A opção de armazenamento em cache em linha simplifica o acesso aos dados pois ela permite que os aplicativos acessem diretamente as APIs do eXtreme Scale. O WebSphere eXtreme Scale suporta diversos cenários de armazenamento em cache em linha, como os seguintes:

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

Cenário de Armazenamento em Cache Read-through

Um cache read-through é um cache disperso que lentamente carrega entradas de dados por chave à medida que elas são solicitadas. Isto é feito sem exigir que o responsável pela chamada saiba quais entradas estão preenchidas. Se os dados não puderem ser localizados no cache do eXtreme Scale, o eXtreme Scale irá recuperar os dados ausentes do plug-in do utilitário de carga, que carrega os dados do banco de dados backend e insere os dados no cache. Pedidos subsequentes para a mesma chave de dados serão localizados no cache até que ele possa ser removido, invalidado ou despejado.

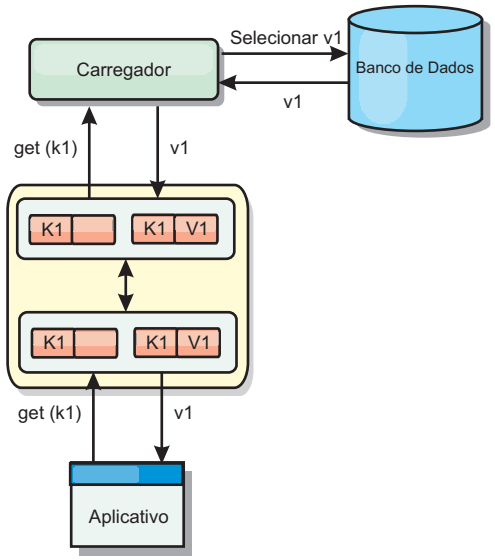


Figura 21. Armazenamento em Cache Read-through

Cenário de Armazenamento em Cache Write-through

Em um cache write-through, cada gravação no cache é gravada de maneira síncrona no banco de dados utilizando o Utilitário de Carga. Este método fornece consistência com o backend, mas diminui o desempenho de gravação pois a operação do banco de dados é síncrona. Como o cache e o banco de dados são ambos atualizados, as leituras subsequentes para os mesmos dados serão localizadas no cache, evitando a chamada do banco de dados. Um cache write-through sempre é utilizado em conjunto com um cache read-through.

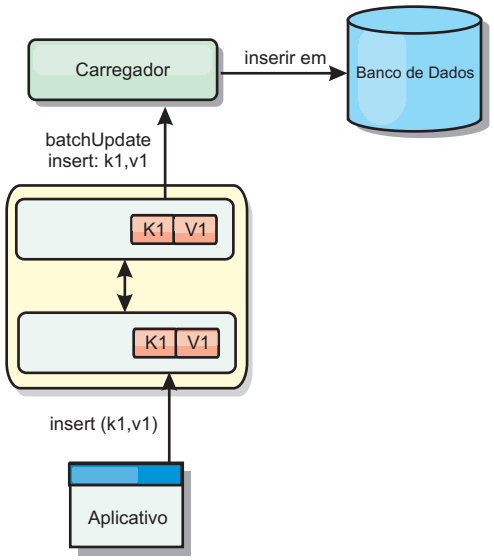


Figura 22. Armazenamento em Cache Write-through

Cenário de Armazenamento em Cache Write-behind

A sincronização do banco de dados pode ser aprimorada pela gravação de alterações de maneira assíncrona. Isto é conhecido como um cache write-behind ou write-back. Alterações que normalmente poderiam ser gravadas de maneira síncrona no utilitário de carga são, ao invés disso, armazenadas em buffer no eXtreme Scale e gravadas no banco de dados utilizando um encadeamento secundário. O desempenho de gravação é significativamente aumentado pois a operação do banco de dados é removida da transação do cliente e as gravações do banco de dados podem ser compactadas.

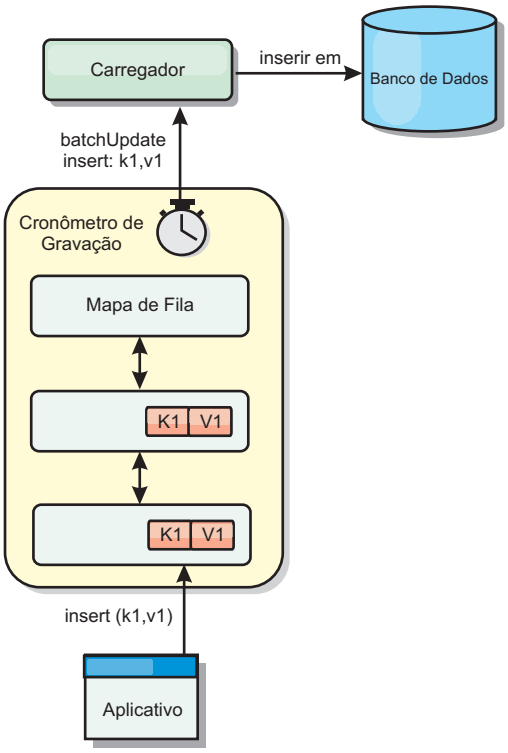


Figura 23. Armazenamento em Cache Write-behind

Armazenamento em Cache Write-behind

É possível utilizar armazenamento em cache write-behind para reduzir o gasto adicional que ocorre durante a atualização de um banco de dados que você está utilizando como back end.

Visão Geral do Armazenamento em Cache Write-Behind

O armazenamento em cache write-behind enfileira assincronamente as atualizações no plug-in do Utilitário de Carga. É possível melhorar o desempenho desconectando atualizações, inserções e remoções para um mapa, a sobrecarga de atualização do banco de dados de backend. A atualização assíncrona é executada após um atraso baseado em tempo (por exemplo, cinco minutos) ou um atraso baseado em entradas (1000 entradas).

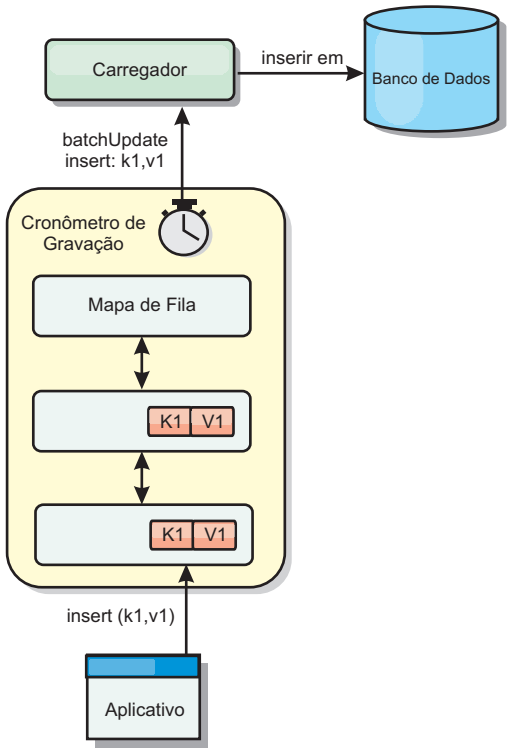


Figura 24. Armazenamento em Cache Write-behind

A configuração write-behind em um BackingMap cria um encadeamento entre o utilitário de carga e o mapa. O utilitário de carga então delega pedidos de dados através do encadeamento de acordo com as definições da configuração no método BackingMap.setWriteBehind. Quando uma transação do eXtreme Scale insere, atualiza ou remove uma entrada de um mapa, um objeto LogElement é criado para cada um destes registros. Estes elementos são enviados para o utilitário de carga write-behind e enfileirados em um ObjectMap especial denominado mapa de fila. Cada mapa de apoio com a configuração write-behind ativada possui seus próprios mapas de fila. Um encadeamento write-behind remove periodicamente os dados enfileirados dos mapas de fila e executa o push deles para o utilitário de carga de backend real.

O utilitário de carga write-behind enviará apenas os tipos insert, update e delete dos objetos LogElement para o utilitário de carga real. Todos os outros tipos de objetos LogElement, por exemplo, o tipo EVICT, são ignorados.

O suporte write-behind *é* uma extensão do plug-in do Carregador, que você usa para integrar o eXtreme Scale ao banco de dados. Por exemplo, consulte as informações do Configurando Utilitários de Carga do JPA sobre como configurar um carregador JPA.

Benefícios

Ativar o suporte write-behind possui os seguintes benefícios:

- **Isolamento de falha de backend:** O armazenamento em cache write-behind fornece uma camada de isolamento das falhas de backend. Quando o banco de dados de backend falha, as atualizações são enfileiradas no mapa de fila. Os

aplicativos podem continuar a conduzir transações para o eXtreme Scale. Quando o backend se recupera, os dados no mapa de fila são enviados para o backend.

- **Carga de backend reduzida:** O utilitário de carga write-behind mescla as atualizações em uma base de chave, portanto, apenas uma atualização mesclada por chave existe no mapa de fila. Esta mesclagem diminui o número de atualizações no backend.
- **Desempenho de transação aprimorado:** Tempos de transação do eXtreme Scale individuais são reduzidos porque a transação não precisa aguardar até que os dados sejam sincronizados com o backend.

Considerações de Design do Aplicativo

Ativar o suporte write-behind é simples, mas o design de um aplicativo para trabalhar com o suporte write-behind precisa de consideração cuidadosa. Sem o suporte de write-behind, a transação de ObjectGrid engloba a transação de backend. A transação do ObjectGrid inicia antes da transação de backend iniciar e termina após a transação de backend terminar.

Com suporte write-behind ativado, a transação do ObjectGrid é concluída antes que a transação de backend inicie. A transação do ObjectGrid e a transação de backend não estão acopladas.

Limitadores de Integridade Referencial

Cada mapa de apoio que é configurado com suporte write-behind possui seu próprio encadeamento write-behind para enviar os dados para o backend. Portanto, os dados que são atualizados em diferentes mapas em uma transação do ObjectGrid são atualizadas no backend em diferentes transações de backend. Por exemplo, a transação T1 atualiza a chave key1 no mapa Map1 e a chave key2 no mapa Map2. A atualização da key1 para o mapa Map1 é atualizada no backend em uma transação de backend e a key2 atualizada para o mapa Map2 é atualizada no backend em outra transação de backend por encadeamentos write-behind diferentes. Se os dados armazenados no Map1 e Map2 possuírem relações, tais como limitadores de chave estrangeira no backend, as atualizações podem falhar.

Ao projetar os limitadores de integridade referencial em seu banco de dados de backend, certifique-se de que atualizações fora de ordem sejam permitidas.

Comportamento do Bloqueio de Mapa de Fila

Outra grande diferença de comportamento da transação é o comportamento do bloqueio. O ObjectGrid suporta três diferentes estratégias de bloqueio: PESSIMISTIC, OPTIMISITIC e NONE. Os mapas de fila write-behind utilizam a estratégia de bloqueio pessimista não importando qual estratégia de bloqueio está configurada para seu mapa de apoio. Existem dois diferentes tipos de operações que adquirem um bloqueio no mapa de fila:

- Quando uma transação do ObjectGrid é confirmada ou um flush (flush de mapa ou flush de sessão) acontece, a transação lê a chave no mapa de fila e coloca um bloqueio S na chave.
- Quando uma transação do ObjectGrid é confirmada, a transação tenta atualizar o bloqueio S para o bloqueio X na chave.

Devido a este comportamento do mapa de fila extra, é possível visualizar algumas diferenças de comportamento de bloqueio.

- Se o mapa do usuário for configurado como a estratégia de bloqueio PESSIMISTIC, não há muita diferença no comportamento de bloqueio. Sempre que um flush ou commit é chamado, um bloqueio S é colocado na mesma chave no mapa de fila. Durante o momento do commit, um bloqueio X não é adquirido apenas para a chave no mapa do usuário, ele também é adquirido para a chave no mapa de fila.
- Se o mapa do usuário for configurado com a estratégia de bloqueio OPTIMISTIC ou NONE, a transação do usuário seguirá o padrão de estratégia de bloqueio PESSIMISTIC. Sempre que um flush ou commit é chamado, um bloqueio S é adquirido para a mesma chave no mapa de fila. Durante o momento do commit, um bloqueio X é adquirido para a chave no mapa de fila utilizando a mesma transação.

Novas Tentativas de Transações do Utilitário de Carga

O ObjectGrid não suporta transações 2-phase ou XA. O encadeamento write-behind remove registros do mapa de fila e atualiza os registros no backend. Se o servidor falhar no meio da transação, algumas atualizações de backend podem ser perdidas.

O utilitário de carga write-behind automaticamente tentará gravar novamente transações falhas e enviará uma LogSequence duvidosa para o backend para evitar a perda de dados. Esta ação requer que o utilitário de carga seja idempotente, o que significa que o Loader.batchUpdate(TxId, LogSequence) é chamado duas vezes com o mesmo valor, ele fornece o mesmo resultado como se tivesse sido aplicado uma vez. As implementações do utilitário de carga devem implementar a interface RetryableLoader para ativar este recurso. Consulte a documentação da API para obter mais detalhes.

Falha do Utilitário de Carga

O plug-in do utilitário de carga pode falhar quando não consegue se comunicar com o back end do banco de dados. Isto pode acontecer se o servidor de banco de dados ou a conexão de rede estiver inativa. O utilitário de carga write-behind irá enfileirar as atualizações e tentará executar o push das alterações de dados para o utilitário de carga periodicamente. O utilitário de carga deve notificar o tempo de execução do ObjectGrid que há um problema de conectividade do banco de dados lançando uma exceção LoaderNotAvailableException.

Portanto, a implementação do Utilitário de Carga deve poder distinguir uma falha de dados ou uma falha de utilitário de carga físico. A falha de dados deve ser lançada ou relançada como uma LoaderException ou uma OptimisticCollisionException, mas uma falha de utilitário de carga físico deve ser lançada ou relançada como uma LoaderNotAvailableException. O ObjectGrid manipula estas exceções de maneira diferente:

- Se uma LoaderException for capturada pelo utilitário de carga write-behind, o utilitário de carga write-behind a considerará falha devido a alguma falha de dados, tal como uma falha de chave duplicada. O utilitário de carga write-behind irá remover a atualização do lote e tentará atualizar um registro em um momento para isolar a falha de dados. Se uma {{LoaderException}} for capturada durante uma atualização de registro, um registro de atualização falho é criado e registrado no mapa de atualização falho.
- Se uma LoaderNotAvailableException for capturada pelo utilitário de carga write-behind, o utilitário de carga write-behind a considerará falha porque não pode se conectar ao final do banco de dados, por exemplo, porque o backend do

banco de dados estiver inativo, uma conexão com o banco de dados não estiver disponível ou a rede estiver inativa. O utilitário de carga write-behind aguardará por 15 segundo e, em seguida, tentará novamente executar uma atualização de lote no banco de dados.

O erro comum é lançar uma LoaderException enquanto uma LoaderNotAvailableException deve ser lançada. Todos os registros enfileirados no utilitário de carga write-behind se tornarão atualizações de registro falhas, o que frustra o propósito do isolamento de falha do backend.

Considerações sobre Desempenho

O suporte ao armazenamento em cache write-behind aumenta o tempo de resposta removendo a atualização do utilitário de carga da transação. Ele também aumenta o rendimento do banco de dados porque as atualizações de banco de dados são combinadas. É importante compreender o gasto adicional introduzido pelo encadeamento write-behind, que executa o pull dos dados da mapa de fila e executa o push para o utilitário de carga.

A contagem máxima de atualização ou o tempo máximo de atualização necessário a ser ajustado com base nos padrões de uso e no ambiente esperados. Se o valor da contagem máxima de atualização ou o tempo máximo de atualização for muito pequeno, o gasto adicional do encadeamento write-behind pode exceder os benefícios. Configurar um valor maior para estes dois parâmetros também pode aumentar o uso da memória para enfileirar os dados e aumentar o tempo de envelhecimento dos registros do banco de dados.

Para obter um melhor desempenho, ajuste os parâmetros write-behind com base nos seguintes fatores:

- Proporção de transações de leitura e gravação
- Mesma frequência de atualização de registro
- Latência de atualização de banco de dados.

Utilitários de Carga

Com um plug-in Carregador, uma grade de dados pode se comportar como um cache de memória para dados que normalmente são mantidos em um armazenamento persistente no mesmo sistema ou em outro sistema. Geralmente, um banco de dados ou sistema de arquivos é utilizado como o armazenamento persistente. Uma JVM (Java Virtual Machine) também pode ser usada como a origem de dados, permitindo que caches baseados em hub seja construído usando o eXtreme Scale. Um utilitário de carga possui a lógica para leitura e gravação de dados para um armazenamento persistente e a partir dele.

Visão Geral

Os utilitários de carga são plug-ins de mapa de apoio que são chamados quando são feitas alterações no mapa de apoio ou quando o mapa de apoio não pode atender a um pedido de dados (um erro de cache). O utilitário de carga é chamado quando o cache não pode satisfazer uma solicitação para uma chave, fornecendo capacidade read-through e lazy-population do cache. Um utilitário de carga também permite atualizações no banco de dados quando os valores do cache mudam. Todas as mudanças dentro de uma transação são agrupadas para permitir que o número de interações do banco de dados seja minimizado. Um plug-in TransactionCallback é usado em conjunto com o utilitário de carga para acionar a demarcação da transação backend. O uso deste plug-in é importante quando

múltiplos mapas são incluídos em uma única transação ou quando os dados da transação forem enviados para o cache sem consolidação.

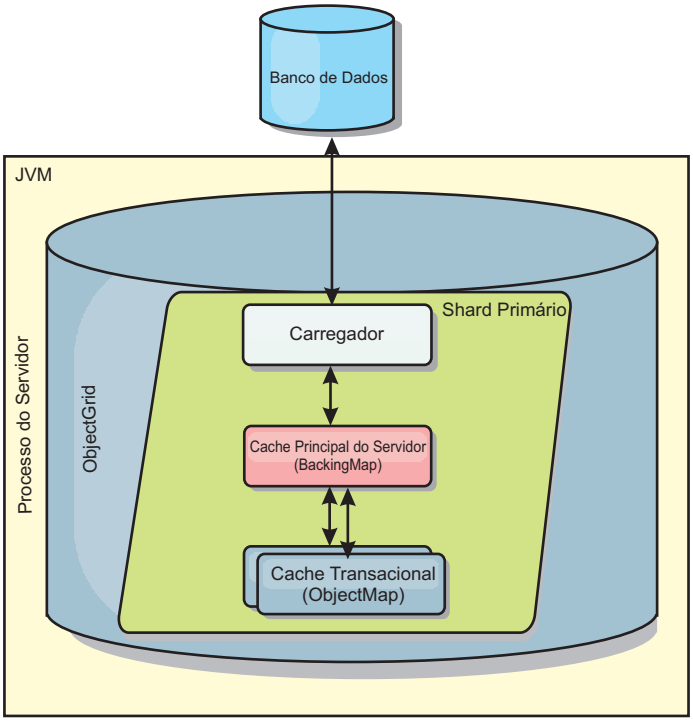


Figura 25. Utilitário de Carga

O utilitário de carga também pode usar atualizações super qualificadas para evitar manter bloqueios do banco de dados. Armazenar um atributo de versão no valor do cache, permite ao utilitário de carga ver a imagem antes e depois do valor quando ele for atualizado no cache. Este valor pode assim ser usado ao atualizar o banco de dados ou backend para verificar se os dados foram atualizados. Um Loader também pode ser configurado para pré-carregar a grade de dados quando for iniciado. Quando particionado, uma instância do utilitário de carga é associada a cada partição. Se o Mapa "Company" tiver dez partições, haverá dez instâncias do utilitário de carga, uma por partição primária. Quando shard primário para o Mapa é ativado, o método preloadMap para o utilitário de carga é chamado síncrona ou assincronamente, o qual permite o carregamento da partição do mapa com dados a partir do backend ocorra automaticamente. Quando chamado sincronamente, todas as transações do cliente são bloqueadas, evitando o acesso inconsistente à grade de dados. Como alternativa, um pré-utilitário de cliente pode ser usado para carregar a grade de dados inteira.

Dois utilitários de carga integrados podem simplificar muito a integração com back ends de banco de dados relacional. Os utilitários de carga JPA utilizam os recursos ORM (Object-Relational Mapping) de ambas as implementações OpenJPA e Hibernate da especificação JPA (Java Persistence API). Consulte “Carregadores JPA” na página 66 para obter mais informações.

Se estiver usando carregadores em uma configuração de diversos datacenters, você deverá considerar como dados de revisão e a consistência de cache são mantidos entre as grades de dados. Para obter informações adicionais, consulte

“Considerações Sobre o Carregador em uma Topologia Multimestre” na página 168.

Configuração do Utilitário de Carga

Para incluir um Utilitário de Carga na configuração do BackingMap, é possível utilizar a configuração programática ou a configuração do XML. Um utilitário de carga possui o seguinte relacionamento com um mapa de apoio.

- Um mapa de apoio pode ter apenas um utilitário de carga.
- Um mapa de apoio de cliente (cache local) não pode ter um utilitário de carga.
- Uma definição de utilitário de carga pode ser aplicado a múltiplos mapas de apoio, mas cada mapa de apoio possui sua própria instância do utilitário de carga.

Pré-carregamento de Dados e Aquecimento

Em vários cenários que incorporam o uso de um carregador, é possível preparar sua grade de dados ao pré-carregá-lo com dados.

Quando usado como um cache completo, a grade de dados deve manter todos os dados e deve ser carregada antes que quaisquer clientes possam se conectar a ele. Quando estiver usando um cache esparso, é possível efetuar um warm-up do cache com dados para que os clientes possam ter acesso imediato aos dados quando eles se conectarem.

Existem duas abordagens para o pré-carregamento de dados na grade de dados: Usando um plug-in do Carregador ou usando um carregador do cliente, conforme descrito nas seguintes seções.

Plug-in do Utilitário de Carga

O plug-in do carregador é associado a cada mapa e é responsável pela sincronização de um único shard de partição primário com o banco de dados. O método preloadMap do plug-in do utilitário de carga é chamado automaticamente quando um shard é ativado. Por exemplo, se você tiver 100 partições, existem 100 instâncias do carregador, cada um carregando os dados para sua partição. Quando executado de modo síncrono, todos os clientes serão bloqueados até que o pré-carregamento seja concluído.

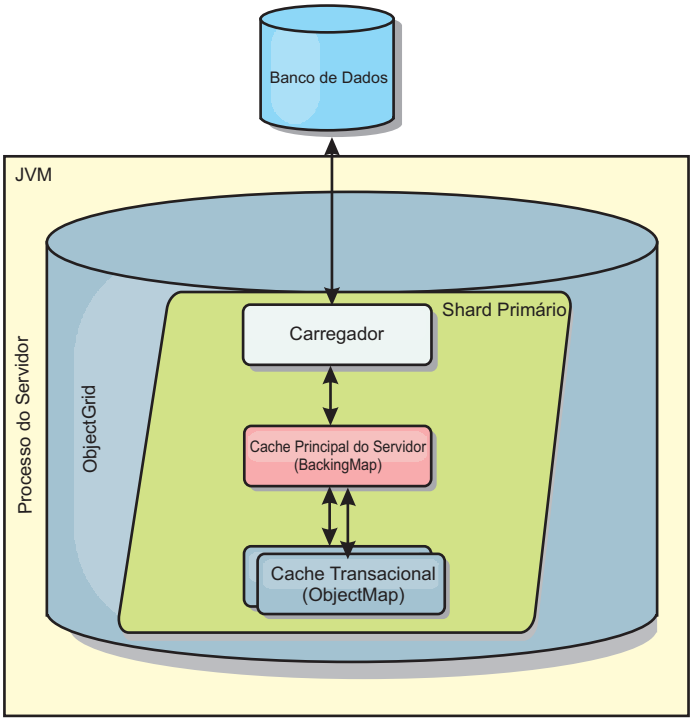


Figura 26. Plug-in do Utilitário de Carga

Utilitário de Carga do Cliente

Um utilitário de carga do cliente é um padrão para uso de um ou mais clientes para carregar a grade com dados. O uso de múltiplos clientes para carregamento de dados da grade pode ser efetivo quando o esquema de partições não está armazenado no banco de dados. É possível chamar os carregadores de cliente manual ou automaticamente quando a grade de dados é iniciada. Os carregadores do cliente podem usar, opcionalmente, o StateManager para configurar o estado da grade de dados no modo de pré-carregamento, para que os clientes não possam acessar a grade enquanto ela estiver pré-carregando os dados. WebSphere eXtreme Scale inclui um carregador baseado em Java Persistence API (JPA)pode ser usado para carregar automaticamente a grade de dados com os provedores JPA OpenJPA ou Hibernate. Para obter mais informações sobre os provedores de cache, consulte “Plug-in do Cache JPA Nível 2 (L2)” na página 26.

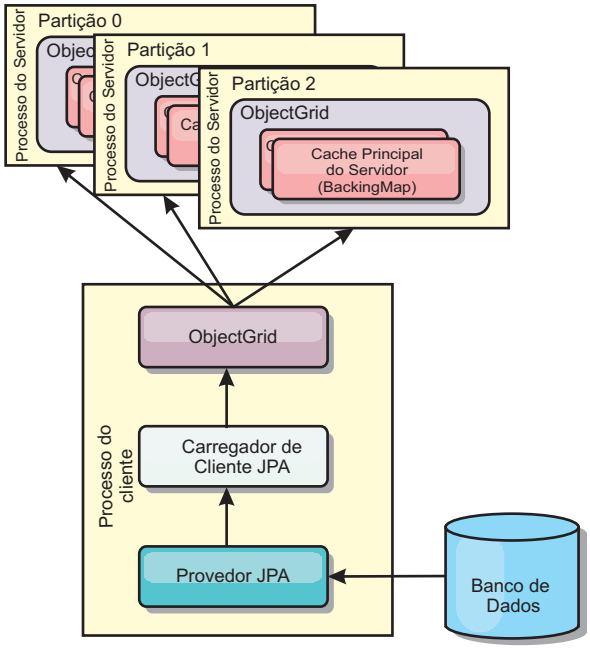


Figura 27. Utilitário de Carga do Cliente

Técnicas de Sincronização de Banco de Dados

Quando o WebSphere eXtreme Scale é utilizado como um cache, os aplicativos devem ser criados para tolerar dados antigos se o banco de dados puder ser atualizado de maneira independente de uma transação do eXtreme Scale. Para atuar como um espaço de processamento de banco de dados de memória sincronizado, o eXtreme Scale fornece várias maneiras de manter o cache atualizado.

Técnicas de Sincronização de Banco de Dados

Atualização periódica

O cache pode ser automaticamente invalidado ou atualizado automaticamente usando o atualizador de banco de dados baseado em tempo JPA (Java Persistence API).O atualizador periodicamente consulta o banco de dados usando um provedor JPA para todas as atualizações ou inserções que ocorreram desde a atualização anterior. Quaisquer alterações identificadas são automaticamente invalidadas ou atualizadas quando utilizadas com um cache disperso. Se utilizadas com um cache completo, as entradas podem ser descobertas e inseridas no cache. As entradas nunca são removidas do cache.

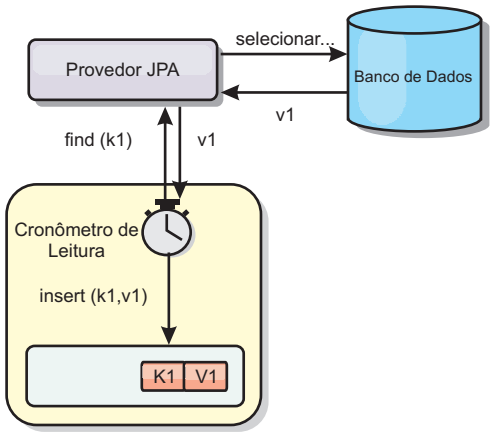


Figura 28. Atualização Periódica

Despejo

Os caches dispersos podem utilizar políticas de despejo para automaticamente remover dados do cache sem afetar o banco de dados. Há três políticas integradas incluídas no eXtreme Scale: time-to-live, least-recently-used e least-frequently-used. Todas as três políticas podem opcionalmente despejar dados mais agressivamente à medida que a memória torna-se restrita ao ativar a opção de despejo baseada em memória.

Invalidação Baseada em Eventos

Caches dispersos e completos podem ser invalidados ou atualizados usando um gerador de eventos como JMS (Java Message Service). A invalidação utilizando JMS pode ser manualmente vinculada a qualquer processo que atualiza o backend utilizando um acionador do banco de dados. Um plug-in ObjectGridEventListener do JMS é fornecido no eXtreme Scale que pode notificar quando o cache do servidor tiver qualquer alteração. Isto pode diminuir a quantidade de tempo que o cliente pode visualizar dados antigos.

Invalidação programática

As APIs do eXtreme Scale permitem interação manual do cache local e do servidor usando os métodos de API Session.beginNoWriteThrough(), ObjectMap.invalidate() e EntityManager.invalidate(). Se um processo do cliente ou servidor não precisar mais de uma parte dos dados, os métodos de invalidação podem ser utilizados para remover dados do cache local ou do servidor. O método beginNoWriteThrough aplica qualquer operação ObjectMap ou EntityManager para o cache local sem chamar o utilitário de carga. Se chamada a partir de um cliente, a operação é aplicável apenas para o cache local (o utilitário de carga remoto não é chamado). Se chamada no servidor, a operação é aplicável apenas ao cache principal do servidor sem chamar o utilitário de carga.

Invalidação de Dados

Para remover os dados de cache de escala, é possível usar um mecanismo de invalidação programático ou baseado em evento.

Invalidação Baseada em Evento

Caches dispersos e completos podem ser invalidados ou atualizados usando um gerador de eventos como JMS (Java Message Service). A invalidação utilizando JMS pode ser manualmente vinculada a qualquer processo que atualiza o backend utilizando um acionador do banco de dados. É fornecido um plug-in JMS ObjectGridEventListener no eXtreme Scale que pode notificar os clientes quando o cache do servidor é alterado. Esse tipo de notificação diminui a quantidade de tempo em que o cliente pode ver dados antigos.

A invalidação baseada em evento normalmente consiste nos três componentes a seguir.

- **Fila de eventos:** Uma fila de eventos armazena os eventos de mudança de dados. Ela pode ser uma fila JMS, um banco de dados, uma fila FIFO em memória ou qualquer tipo de manifesto, contanto que possa gerenciar os eventos de mudança de dados.
- **Publicador de evento:** Um publicador de evento publica os eventos de mudança de dados na fila de eventos. Um publicador de evento geralmente é um aplicativo que você cria ou uma implementação de plug-in do eXtreme Scale. O publicador de evento sabe quando os dados são alterados ou quando ele mesmo altera os dados. Quando uma transação é confirmada, eventos são gerados para os dados alterados e o publicador de eventos publica esses eventos na fila de eventos.
- **Consumidor de evento:** Um consumidor de evento consome eventos de mudança de dados. O consumidor de evento geralmente é um aplicativo para garantir que os dados da grade de destino sejam atualizados com a mudança mais recente das outras grades. Esse consumidor de evento interage com a fila de eventos para obter a mudança de dados mais recente, além de aplicar as mudanças de dados na grade de destino. Os consumidores de evento podem utilizar APIs do eXtreme Scale para invalidar dados antigos ou atualizar a grade com os dados mais recentes.

Por exemplo, JMSObjectGridEventListener tem uma opção para um modelo de cliente/servidor, no qual a fila de eventos é um destino JMS designado. Todos os processos do servidor são publicadores de eventos. Quando uma transação é confirmada, o servidor obtém as mudanças de dados e as publica no destino JMS designado. Todos os processos do cliente são consumidores de evento. Eles recebem as mudanças de dados do destino JMS designado e aplicam as mudanças no cache perto do cliente.

Consulte o tópico sobre a ativação do mecanismo de invalidação de cliente no *Guia de Administração* para obter mais informações.

Invalidação Programática

As APIs do WebSphere eXtreme Scale permitem interação manual do cache local e do servidor usando os métodos de API Session.beginNoWriteThrough(), ObjectMap.invalidate() e EntityManager.invalidate(). Se um processo do cliente ou servidor não precisar mais de uma parte dos dados, os métodos de invalidação podem ser utilizados para remover dados do cache local ou do servidor. O método beginNoWriteThrough aplica qualquer operação ObjectMap ou EntityManager para o cache local sem chamar o utilitário de carga. Se chamada a partir de um cliente, a operação é aplicável apenas para o cache local (o utilitário de carga remoto não é chamado). Se chamada no servidor, a operação é aplicável apenas ao cache principal do servidor sem chamar o utilitário de carga.

É possível utilizar invalidação programática com outras técnicas para determinar quando invalidar os dados. Por exemplo, esse método de invalidação utiliza mecanismos de invalidação baseados em evento para receber eventos de mudança de dados e depois utiliza as APIs para invalidar os dados antigos.

Indexação

Use o plug-in MapIndexPlugin para construir um índice ou vários índices em um BackingMap para suportar acesso a dados sem chave.

Tipos e Configuração de Índice

O recurso de indexação é representado pelo plug-in MapIndexPlugin ou Index, para abreviar. O Índice é um plug-in BackingMap. Um BackingMap pode ter múltiplos plug-ins de Índice configurados, enquanto cada um seguir as regras de configuração de Índice.

O recurso de indexação pode ser usado para construir um ou mais índices em um BackingMap. Um índice é construído a partir de um atributo ou uma lista de atributos de um objeto no BackingMap. Este recurso fornece uma maneira para os aplicativos localizarem determinados objetos mais rapidamente. Com o recurso de indexação, os aplicativos podem localizar objetos com um valor específico ou em um intervalo com os valores de atributos indexados.

Dois tipos de indexação são possíveis: estática e dinâmica. Com a indexação estática, é necessário configurar o plug-in de índice no BackingMap antes de inicializar a instância do ObjectGrid. É possível fazer esta configuração com a configuração XML ou programática do BackingMap. A indexação estática inicia a construção de um índice durante a inicialização do ObjectGrid. O índice é sempre sincronizado com o BackingMap e está pronto para utilização. Depois de o processo de indexação estático iniciar, a manutenção do índice é parte do processo de gerenciamento de transação do eXtreme Scale. Quando as consolidações de transações mudam, estas alterações também atualizam o índice estático, e as alterações de índice são recuperadas se a transação for recuperada.

Com a indexação dinâmica, é possível criar um índice num BackingMap antes ou depois da inicialização da instância do ObjectGrid que o contém. Os aplicativos possuem controle de ciclo de vida sobre o processo de indexação dinâmica para que você possa remover um índice dinâmico quando ele não for mais necessário. Quando um aplicativo cria um índice dinâmico, o índice pode não estar pronto para utilização imediata devido ao tempo gasto na conclusão do processo de construção do índice. Como a quantidade de tempo depende da quantidade de dados indexados, a interface DynamicIndexCallback é fornecido para aplicativos que desejam receber notificações quando ocorrem determinados eventos de indexação. Estes eventos incluem ready, error e destroy. Os aplicativos podem implementar esta interface de retorno de chamada e registrar-se no processo de indexação dinâmica.

Se um BackingMap tiver um plug-in de índice configurado, você pode obter o objeto de proxy do índice do aplicativo a partir do ObjectMap correspondente. Chamar o método getIndex no ObjectMap e passar o nome do plug-in de índice retornam o objeto de proxy do índice. Você deve efetuar o cast do objeto de proxy do índice para uma interface de índice adequada do aplicativo, como MapIndex, MapRangeIndex ou uma interface de índice customizada. Após obter o objeto de proxy do índice, é possível utilizar métodos definidos na interface do índice do aplicativo para localizar objetos armazenados em cache.

As etapas para utilizar a indexação estão resumidas na lista a seguir:

- Incluir plug-ins de indexação, estáticos ou dinâmicos, no BackingMap;
- Obter um objeto de proxy de indexação do aplicativo, emitindo o método getIndex do ObjectMap.
- Direcione o objeto de proxy de índice a uma interface de índice de aplicativo apropriado, como MapIndex, MapRangeIndex, ou uma interface de índice customizada.
- Utilizar os métodos definidos na interface Index do aplicativo para localizar objetos no cache.

A classe HashIndex é a implementação de plug-in de índice integrada que pode suportar ambas as interfaces integradas de índice do aplicativo: MapIndex e MapRangeIndex. Também é possível criar seus próprios índices. É possível incluir o HashIndex como um índice estático ou dinâmico no BackingMap, obter o objeto proxy do índice MapIndex ou MapRangeIndex e usar o objeto proxy do índice para localizar objetos em cache.

Índice Padrão

Se desejar iterar por meio das chaves em um mapa local, o índice padrão poderá ser usado. Este índice não requer nenhuma configuração, porém ele deve ser usado com relação ao shard, usando um agente ou uma instância do ObjectGrid recuperados a partir do método ShardEvents.shardActivated(shard do ObjectGrid).

Consideração sobre a Qualidade dos Dados

Os resultados dos métodos de consulta de índice somente representar uma captura instantânea de dados em um determinado ponto no tempo. Nenhum bloqueio contra as entradas de dados é obtido depois do retorno dos resultados para o aplicativo. O aplicativo deve estar ciente de que podem ocorrer atualizações de dados em um conjunto de dados retornado. Por exemplo, o aplicativo obtém a chave de um objeto armazenado em cache executando o método findAll de MapIndex. Este objeto de chave retornado está associado a uma entrada de dados no cache. O aplicativo deve poder executar o método get no ObjectMap para localizar um objeto fornecendo o objeto chave. Se outra transação remover o objeto de dados do cache imediatamente antes de o método get ser chamado, o resultado retornado será nulo.

Considerações sobre Desempenho de Indexação

Um dos principais objetivos do recurso de indexação é melhorar o desempenho geral do BackingMap. Se a indexação não for utilizada corretamente, o desempenho do aplicativo poderá ficar comprometido. Considere os seguintes fatores antes de usar este retorno.

- **A quantidade de transações de gravação concorrentes:** O processamento de índices pode ocorrer todas as vezes que uma transação gravar dados em um BackingMap. O desempenho será afetado se muitas transações gravarem dados no mapa simultaneamente quando um aplicativo tentar operações de consulta ao índice.
- **O tamanho do conjunto de resultados que é retornado por uma operação de consulta:** Como o tamanho do conjunto de resultados aumenta, o desempenho da consulta cai. O desempenho tente a degradas quando o tamanho do conjunto de resultados é de 15% (ou mais) do BackingMap.

- **A quantidade de índices construídos sobre o mesmo BackingMap:** Cada índice consome os recursos do sistema. Conforme a quantidade dos índices construídos sobre o BackingMap aumenta, o desempenho diminui.

A função de indexação pode aumentar significativamente o desempenho do BackingMap. Os casos ideais ocorrem quando o BackingMap possui a maioria de operações de leitura, o conjunto de resultados da consulta é de uma porcentagem pequena das entradas do BackingMap, e somente poucos índices são construídos sobre o BackingMap.

Carregadores JPA

O Java Persistence API (JPA) é uma especificação que permite o mapeamento de objetos Java para bancos de dados relacionais. O JPA contém uma especificação completa de object-relational mapping (ORM) usando anotações de metadados da linguagem Java, descritores XML, ou ambos para definir o mapeamento entre objetos Java e um banco de dados relacional. Inúmeras implementações comerciais e de software livre estão disponíveis.

É possível utilizar uma implementação de plug-in de utilitário de carga do Java Persistence API (JPA) com eXtreme Scale para interagir com qualquer banco de dados suportado por seu utilitário de carga escolhido. Para usar o JPA, é necessário ter um provedor JPA suportado, como OpenJPA ou Hibernate, arquivos JAR e um arquivo`META-INF/persistence.xml` no seu caminho da classe.

Os plug-ins JPALoader com.ibm.websphere.objectgrid.jpa.JPALoader e JPAEntityLoader com.ibm.websphere.objectgrid.jpa.JPAEntityLoader são dois plug-ins do utilitário de carga do JPA integrados que são usados para sincronizar os mapas do ObjectGrid com um banco de dados. É necessário ter uma implementação do JPA, como Hibernate ou OpenJPA, para usar este recurso. O banco de dados pode ser qualquer back end que seja suportado pelo provedor JPA escolhido.

É possível usar o plug-in do JPALoader ao armazenar dados usando a API ObjectMap. Use o plug-in do JPAEntityLoader ao armazenar dados usando a API EntityManager.

Arquitetura do Utilitário de Carga do JPA

O Utilitário de Carga do JPA é usado para mapas do eXtreme Scale que armazenam objetos Java antigos simples (POJO).

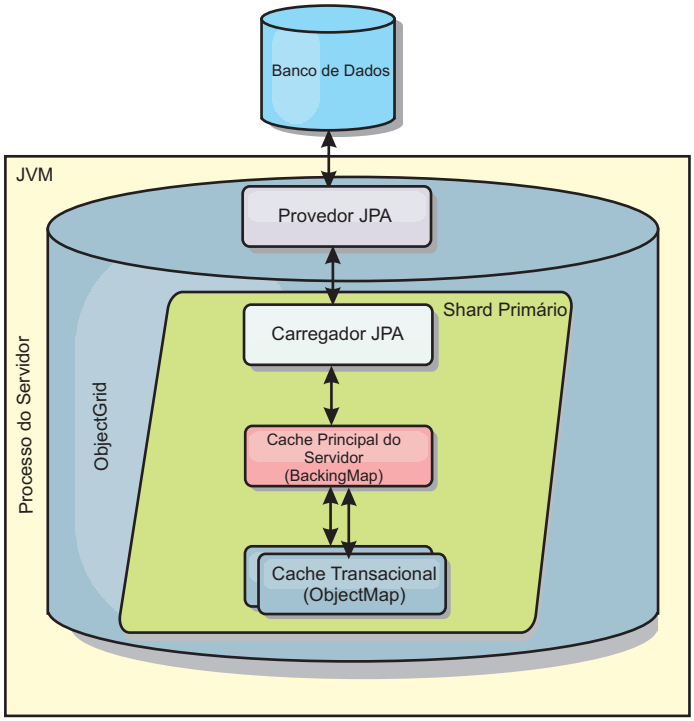


Figura 29. Arquitetura do Utilitário de Carga do JPA

Quando um método ObjectMap.get(Object key) é chamado, o eXtreme Scale executa as primeiras verificações se a entrada está contida na camada do ObjectMap. Se não, o tempo de execução delega a solicitação ao Utilitário de Carga do JPA. Sob solicitação de carregamento da chave, o JPALoader chama o método EntityManager.find(Object key) do JPA para localizar os dados de uma camada do JPA. Se os dados estiverem contidos no gerenciador de entidades JPA, ele serão retornados; caso contrário, o provedor JPA interage com o banco de dados para obter o valor.

Quando uma atualização para o ObjectMap ocorre, por exemplo, usando o método ObjectMap.update(Objectkey, Object value), o tempo de execução do eXtreme Scale cria um LogElement para esta atualização e a envia para o JPALoader. O JPALoader chama o método EntityManager.merge(Object value) do JPA para atualizar o valor no banco de dados.

Para o JPAEntityLoader, as mesmas quatro camadas estão envolvidas. Porém, como o plug-in JPAEntityLoader é usado para mapas que armazenam entidades do eXtreme Scale, as relações entre as entidades poderiam complicar o cenário de uso. Uma entidade do eXtreme Scale é diferenciada de uma entidade do JPA. Para obter mais informações, consulte o plug-in JPAEntityLoader no *Guia de Programação*.

Métodos

Utilitários de Carga Fornecem Três Métodos Principais:

1. get: Retorna uma lista de valores que corresponde à lista de chaves que são passadas por meio da recuperação de dados usando o JPA. O método usa o JPA para localizar as entidades no banco de dados. Para o plug-inJPALoader, a lista retornada contém uma lista de entidades JPA diretamente a partir da operação find. Para o plug-in JPAEntityLoader, a lista retornada contém tuplas do valor da entidade de eXtreme Scale convertidas de entidades do JPA.

2. batchUpdate: grava os dados dos mapas do ObjectGrid para o banco de dados. Dependendo dos diferentes tipos de operação (inserir, atualizar ou excluir), o utilitário de carga usa as operações de persistir, mesclar ou remover para atualizar os dados para o banco de dados. Para o JPALoader, os objetos no mapa são utilizados diretamente como entidades JPA. Para o JPAEntityLoader, as tuplas de entidade no mapa são convertidas nos objetos que são utilizados como entidades JPA.
3. preloadMap: Pré-carrega o mapa usando o método do utilitário de carga do ClientLoader.load. Para mapas particionados, o método preloadMap é chamado apenas em uma partição. A partição é especificada na propriedade preloadPartition da classe JPALoader ou JPAEntityLoader. Se o valor de preloadPartition for configurado para menor que zero ou maior que (*total_number_of_partitions* - 1), o pré-carregamento será desativado.

Ambos os plug-ins JPALoader e JPAEntityLoader funcionam com a classe JPATxCallback para coordenar as transações do eXtreme Scale e as transações do JPA. O JPATxCallback precisa ser configurado na instância do ObjectGrid para utilizar estes dois utilitários de carga.

Configuração e Programação

Se você estiver usando os carregadores JPA em um ambiente multimestre, consulte o "Considerações Sobre o Carregador em uma Topologia Multimestre" na página 168. Para obter mais informações sobre como configurar os carregadores do JPA, consulte informações sobre os carregadores do JPA no *Guia de Administração*. Para obter informações adicionais sobre utilitário de carga do JPA de programação, consulte o *Guia de Programação*.

Visão Geral da Serialização

Os dados são sempre expressos, porém não necessariamente armazenados, como objetos Java na grade de dados. O WebSphere eXtreme Scale usa diversos processos Java para serializar os dados, ao converter as instâncias de objetos Java em bytes e retornar para os objetos novamente, conforme necessário, para mover os dados entre os processos do cliente e do servidor.

Os dados são serializados quando são convertidos em um fluxo de dados para transmissão sobre uma rede, nas situações a seguir:

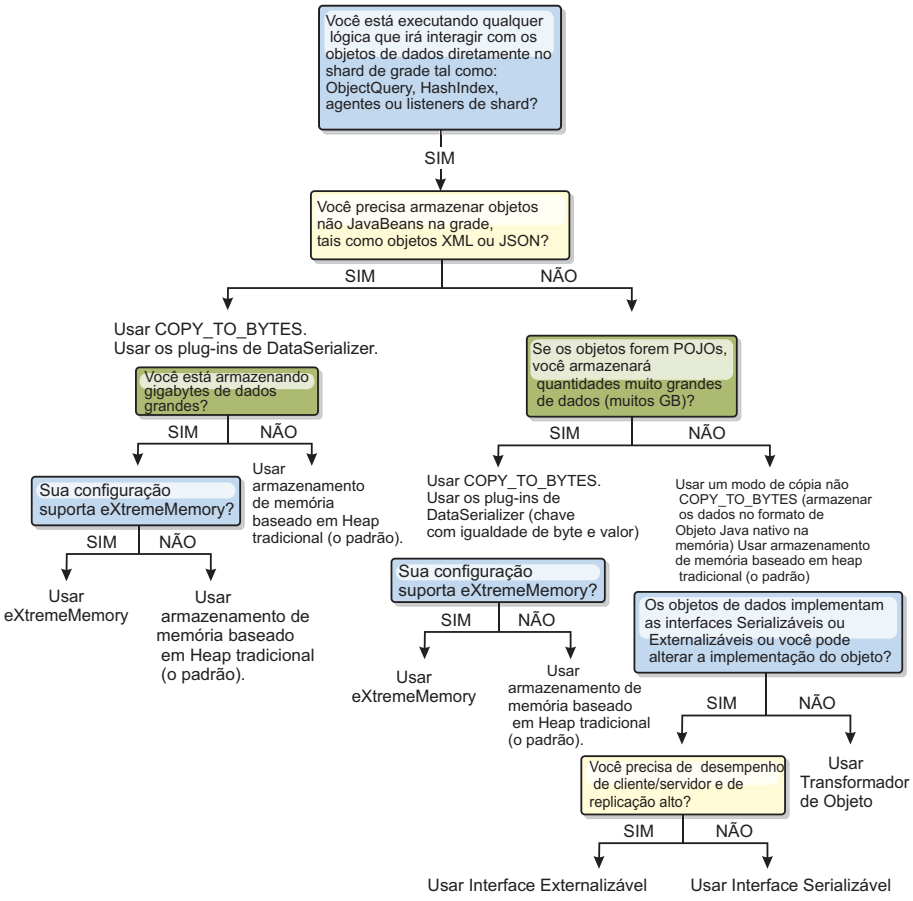
- Quando os clientes se comunicam com os servidores, e esses servidores enviam informações de volta para o cliente
- Quando servidores replicam dados de um servidor para outro

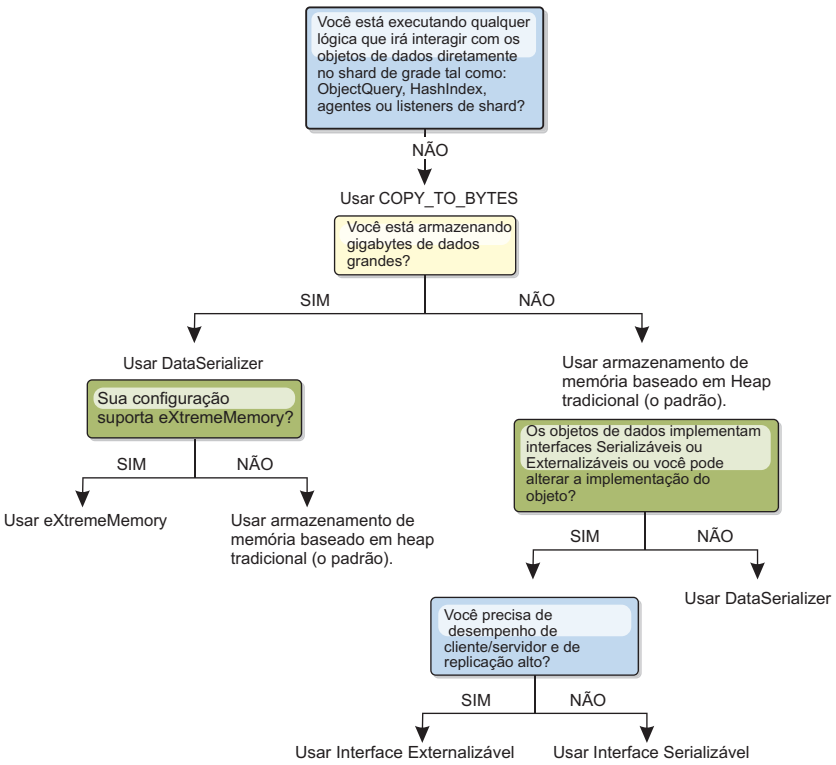
Como alternativa, você pode decidir abrir mão do processo de serialização por meio do WebSphere eXtreme Scale e armazenar dados brutos como matrizes de bytes. As matrizes de bytes são muito mais baratas de armazenar na memória pois a Java Virtual Machine (JVM) possui menos objetos para procurar durante a coleta de lixo e elas podem ser deserializadas somente quando necessário. Use as matrizes de bytes somente se você não precisar acessar os objetos usando consultas ou índices. Como os dados são armazenados como bytes, o eXtreme Scale não possui metadados para descrever atributos para consultar.

Para serializar dados no eXtreme Scale, é possível usar a serialização Java, o plug-in ObjectTransformer ou os plug-ins DataSerializer. Para otimizar a serialização com qualquer uma dessas opções, é possível usar o modo COPY_TO_BYTES para melhorar o desempenho em até 70 por cento porque os

dados são serializados quando transações são confirmadas, significando que a serialização ocorre apenas uma vez. Os dados serializados são enviados sem alteração do cliente para o servidor ou do servidor para servidor replicado. Utilizando o modo COPY_TO_BYTES, é possível reduzir a área de cobertura de memória que um grande gráfico de objetos pode consumir.

Use as figuras a seguir para ajudá-lo a determinar qual tipo de método de serialização é mais apropriado para suas necessidades de desenvolvimento. A primeira figura descreve os métodos de serialização que estão disponíveis quando você está executando a lógica que interage com objetos de dados diretamente no shard de grade. A última figura exibe as opções disponíveis quando você não estão interagindo diretamente com o shard de grade.





Para saber mais sobre as formas de serialização suportadas no produto eXtreme Scale, consulte os seguintes tópicos:

Serialização Usando Java

A serialização Java refere-se à serialização padrão, que usa a interface Serializável, ou à serialização customizada, que usa ambas as interfaces Serializável e Externalizável.

Serialização Padrão

Para usar a serialização padrão, implemente a interface java.io.Serializable, a qual inclui a API que converte objetos em bytes, que são desserializados posteriormente. Use a classe java.io.ObjectOutputStream para persistir o objeto. Em seguida, chame o método ObjectOutputStream.writeObject() para iniciar a serialização e simplificar o objeto Java.

Serialização Customizada

Há alguns casos em que os objetos devem ser modificados para usar a serialização customizada, tais como implementar a interface java.io.Externalizable ou implementar os métodos writeObject e readObject para classes implementando a interface java.io.Serializable. As técnicas de serialização customizadas devem ser empregadas quando os objetos são serializados utilizando mecanismos que não os métodos da API do ObjectGrid ou da API do EntityManager.

Por exemplo, quando objetos ou entidades são armazenados como dados da instância em um agente da API do DataGrid ou o agente retorna objetos ou entidades, tais objetos não são transformados utilizando um ObjectTransformer. O agente, entretanto, utilizará automaticamente o ObjectTransformer ao utilizar a interface `EntityMixin`. Consulte Agentes do DataGrid e Mapas Baseados em Entidade para obter mais detalhes.

Plug-in ObjectTransformer

Com o plug-in ObjectTransformer, é possível serializar, desserializar e copiar objetos no cache para aumentar o desempenho.

 A interface ObjectTransformer foi substituída pelos plug-ins DataSerializer, que podem ser usados para armazenar dados arbitrários eficientemente no WebSphere eXtreme Scale para que as APIs do produto existentes possam ser interagir de modo eficiente com seus dados.

Se você tiver problemas de desempenho com o uso do processador, inclua um plug-in ObjectTransformer em cada mapa. Se um plug-in ObjectTransformer não for fornecido, o processador gastará de 60 a 70% de seu tempo total só serializando e copiando entradas.

Propósito

O plug-in ObjectTransformer permite que os aplicativos forneçam métodos customizados para as seguintes operações:

- Serializar ou desserializar a chave para uma entrada
- Serializar ou desserializar o valor para uma entrada
- Copiar uma chave ou valor para uma entrada

Se nenhum plug-in ObjectTransformer for fornecido, será necessário serializar as chaves e valores, porque o ObjectGrid utiliza a seqüência serializar e desserializar para copiar os objetos. Este método é caro, portanto, utilize um plug-in ObjectTransformer quando o desempenho for importante. A cópia ocorre quando um aplicativo consulta um objeto em uma transação pela primeira vez. É possível evitar essa cópia configurando o modo de cópia como COPY_ON_READ ou reduzir a cópia configurando o modo de cópia como COPY_ON_READ. Otimize a operação de cópia quando requerido pelo aplicativo, fornecendo um método de cópia customizado neste plug-in. Esse plug-in pode reduzir a sobrecarga de cópia de 65 a 70% para 2 a 3% do tempo total do processador.

As implementações dos métodos padrão copyKey e copyValue primeiro tentam utilizar o método clone, se este for fornecido. Se nenhuma implementação do método clone for fornecida, a implementação será padronizada como serialização.

A serialização do objeto também é utilizada diretamente quando o eXtreme Scale estiver em execução no modo distribuído. LogSequence utiliza o plug-in ObjectTransformer para ajudá-lo a serializar chaves e valores antes de transmitir as alterações para os equivalentes no ObjectGrid. Cuidado ao fornecer um método de serialização customizado em vez de utilizar a serialização do Java developer kit integrada. O controle de versões do objeto é um assunto complexo e é possível encontrar problemas com a compatibilidade de versões se você não assegurar que seus métodos customizados foram projetados para controle de versões.

A lista a seguir descreve como o eXtreme Scale tenta serializar chaves e valores:

- Se um plug-in ObjectTransformer customizado for gravado e conectado, o eXtreme Scale chamará os métodos nos métodos na interface ObjectTransformer para serializar chaves e valores e obter cópias de chaves e valores do objeto.
- Se um plug-in ObjectTransformer customizado não for usado, o eXtreme Scale serializa e desserializa os valores de acordo com o padrão. Se o plug-in padrão for utilizado, cada objeto será implementado como externalizável ou implementado como serializável.

- Se o objeto suportar a interface Externalizável, o método writeExternal será chamado. Os objetos que são implementados como externalizáveis geram melhor desempenho.
- Se o objeto não suportar a interface Externalizável e implementar a interface Serializável, o objeto será salvo usando o método ObjectOutputStream.

Utilizando a Interface ObjectTransformer

Um objeto ObjectTransformer precisa implementar a interface ObjectTransformer e seguir as convenções comuns do plug-in ObjectGrid.

Duas abordagens, configuração programática e configuração XML, são utilizadas para incluir um objeto ObjectTransformer na configuração BackingMap da seguinte forma.

Conectando um Objeto ObjectTransformer Programaticamente

O fragmento de código a seguir cria o objeto ObjectTransformer customizado e o inclui em um BackingMap:

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

Abordagem de Configuração XML para Conectar um ObjectTransformer

Suponha que o nome da classe da implementação ObjectTransformer seja a classe com.company.org.MyObjectTransformer. Essa classe implementa a interface ObjectTransformer. Uma implementação ObjectTransformer pode ser configurada usando o seguinte 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="myGrid">
     <backingMap name="myMap" pluginCollectionRef="myMap" />
        </objectGrid>
    </objectGrids>

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

Cenários de Uso do ObjectTransformer

É possível utilizar o plug-in ObjectTransformer nas seguintes situações:

- Objeto não-serializável
- Objeto serializável mas aprimorando o desempenho da serialização
- Cópia de chave ou valor

No exemplo a seguir, o ObjectGrid é utilizado para armazenar a classe Stock:

```
/**
* Objeto Stock para demo do ObjectGrid
*
*
*/
```

```
public class Stock implements Cloneable {
    String ticket;
    double price;
    String company;
    String description;
    int serialNumber;
    long lastTransactionTime;
    /**
    * @return Retorna a descrição.
    */
    public String getDescription()  {
        return description;
    }
    /**
    * @param description A descrição a ser configurada.
    */
    public void setDescription(String description) {
        this.description = description;
    }
    /**
    * @return Retorna lastTransactionTime.
    */
    public long getLastTransactionTime() {
        return lastTransactionTime;
    }
    /**
    * @param lastTransactionTime O último lastTransactionTime a ser configurado.
    */
    public void setLastTransactionTime(long lastTransactionTime) {
        this.lastTransactionTime = lastTransactionTime;
    }
    /**
    * @return Retorna o preço.
    */
    public double getPrice() {
        return price;
    }
    /**
    * @param price O preço a ser configurado.
    */
    public void setPrice(double price) {
        this.price = price;
    }
    /**
    * @return Retorna um serialNumber.
    */
    public int getSerialNumber() {
        return serialNumber;
    }
    /**
    * @param serialNumber O serialNumber a ser configurado.
    */
    public void setSerialNumber(int serialNumber) {
        this.serialNumber = serialNumber;
    }
    /**
    * @return Retorna o registro.
    */
    public String getTicket() {
        return ticket;
    }
    /**
    * @param ticket O registro a ser configurado.
    */
    public void setTicket(String ticket) {
        this.ticket = ticket;
    }
    /**
    * @return Retorna a empresa.
    */
    public String getCompany() {
        return company;
    }
    /**
    * @param company A empresa a ser configurada.
    */
    public void setCompany(String company) {
        this.company = company;
    }
    //clone
```

```
    public Object clone() throws CloneNotSupportedException
    {
        return super.clone();
    }
}
```

É possível gravar uma classe do transformador do objeto customizado para a classe Stock:

```
/**
* Implementação customizada do ObjectTransformer do ObjectGrid para objeto stock
*
*/
public class MyStockObjectTransformer implements ObjectTransformer {
/* (non-Javadoc)
* @see
* com.ibm.websphere.objectgrid.plugins.ObjectTransformer#serializeKey
* (java.lang.Object,
* java.io.ObjectOutputStream)
*/
public void serializeKey(Object key, ObjectOutputStream stream) throws IOException {
    String ticket= (String) key;
    stream.writeUTF(ticket);
}

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

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

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

public Object inflateValue(ObjectInputStream stream) throws IOException, ClassNotFoundException {
    Stock stock=new Stock();
    stock.setTicket(stream.readUTF());
    stock.setCompany(stream.readUTF());
    stock.setDescription(stream.readUTF());
    stock.setPrice(stream.readDouble());
    stock.setLastTransactionTime(stream.readLong());
    stock.setSerialNumber(stream.readInt());
    return stock;
}

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

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

Em seguida, conecte esta classe MyStockObjectTransformer customizada ao BackingMap:

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

Serialização Usando os Plug-ins DataSerializer

Use os plug-ins DataSerializer para armazenar com eficiência dados arbitrários do WebSphere eXtreme Scale para que as APIs do produto existente possam interagir de modo eficiente com seus dados.

Os métodos de serialização, como serialização Java e o plug-in ObjectTransformer, permitem que os dados sejam serializados pela rede. Além disso, quando você usar estas opções de serialização com o modo de cópia COPY_TO_BYTES, a movimentação de dados entre clientes e servidores se torna menos dispendiosa e melhora o desempenho. No entanto, estas opções não resolvem os seguintes problemas que podem existir:

- As chaves não são armazenadas em bytes; elas ainda são objetos Java.
- O código do lado do servidor ainda deve aumentar o objeto, por exemplo, a consulta e o índice ainda usam a reflexão e devem aumentar o objeto. Além disso, agentes, listeners e os plug-ins ainda precisam do formulário do objeto.
- As classes ainda precisam estar no caminho de classe do servidor.
- Os dados ainda estão em formato de serialização Java (ObjectOutputStream).

Os plug-ins do DataSerializer introduzem uma maneira eficiente de resolver estes problemas. Especificamente, o plug-in DataSerializer fornece uma maneira de descrever seu formato de serialização, ou uma matriz de bytes, para o WebSphere eXtreme Scale de modo que o produto possa interrogar a matriz de bytes sem requerer um formato de objeto específico. As classes e interfaces de plug-in DataSerializer públicas estão no pacote com.ibm.websphere.objectgrid.plugins.io. Para obter mais informações, consulte a .

Importante: Os objetos Java de entidade não são armazenados diretamente no BackingMaps quando você usa a API do EntityManager. A API do EntityManager converte o objeto de entidade em objetos de Tupla. Os mapas de entidade são automaticamente associados com um ObjectTransformer altamente otimizado. Sempre que a API do ObjectMap ou a API do EntityManager for utilizada para interagir com mapas de entidade, a entidade ObjectTransformer será chamada. Portanto, quando você usa entidades, nenhum trabalho é necessário para serialização porque o produto automaticamente conclui este processo para você.

Visão Geral de Escalabilidade

O WebSphere eXtreme Scale é escalável por meio do uso de dados particionados, e pode ser escalado para milhares de contêineres se necessário, pois cada contêiner é independente um do outro.

O WebSphere eXtreme Scale divide os conjuntos de dados em partições distintas que podem ser movidas entre os processos ou mesmo entre os servidores físicos no tempo de execução. É possível, por exemplo, iniciar com uma implementação de quatro servidores e, em seguida, expandir para uma implementação com 10 servidores conforme as demandas do cache crescem. Assim como é possível incluir mais servidores físicas e unidades de processamento para escalabilidade vertical, é possível estender o recurso de escalação elástica do horizontalmente com o

particionamento. Escala horizontal é uma grande vantagem de usar o WebSphere eXtreme Scale sobre um banco de dados em memória. Bancos de dados em memória podem ser escalados apenas verticalmente.

Com o WebSphere eXtreme Scale, também é possível usar um conjunto de APIs para obter acesso transacional desses dados particionados e distribuídos. As opções feitas para interagir com o cache são tão significativas quanto as funções para gerenciar o cache para disponibilidade a partir de uma perspectiva de desempenho.

Nota: A escalabilidade não está disponível quando os contêineres se comunicam entre si. O protocolo de gerenciamento de disponibilidade, ou de agrupamento principal, é uma pulsação $O(N^2)$ e um algoritmo de manutenção de visualização, porém é mitigado ao manter o número de membros do grupo principal em 20. Apenas a replicação ponto a ponto existe entre shards.

Clientes Distribuídos

O protocolo do cliente do WebSphere eXtreme Scale suporta quantidades muito grandes de clientes. A estratégia de particionamento oferece assistência assumindo que todos os clientes nem sempre estão interessados em todas as partições porque as conexões podem ser propagadas em vários contêineres. Os clientes se conectam diretamente às partições para que a latência seja limitada a uma conexão transferida.

Partições de Grade de Dados e Shards

Uma grade de dados é dividida em partições. Uma partição retém um subconjunto exclusivo de dados. Uma partição contém um ou mais shards: um shard primário e shards de réplicas. Os shards de réplicas não são necessários em uma partição, porém eles podem ser usados para fornecer alta disponibilidade. Independente se sua implementação for uma grade de dados em memória independente ou um espaço de processamento de banco de dados em memória, o acesso aos dados no eXtreme Scale depende fortemente dos shards.

Os dados para uma partição são armazenados em um conjunto de shards no tempo de execução. Este conjunto de shards inclui um shard principal e, possivelmente, um ou mais shards de réplica. Um shard é a menor unidade que o eXtreme Scale pode incluir ou remover de uma Java Virtual Machine.

Há duas estratégias de posicionamento: posicionamento de partição fixo (padrão) e de posicionamento por contêiner. A seguinte abordagem foca o uso da estratégia de posicionamento de partição fixo.

Número Total de Shards

Se seu ambiente inclui 10 partições que contêm 1 milhão de objetos sem réplicas, existirão 10 shards, cada um armazenando 100.000 objetos. Se você incluir uma réplica neste cenário, então, existe um shard extra em cada partição. Neste caso, há 20 shards: 10 shards primários e 10 shards de réplica. Cada um destes shards armazenam 100.000 objetos. Cada partição consiste de um shard primário e uma ou mais (N) shards de réplica. Determinar a contagem de shards ideal é crítica. Se você configurar um pequeno número de shards, os dados não são distribuídos igualmente entre os shards, resultando em erros de falta de memória e problemas de sobrecarga do processador. Pelo menos 10 shards devem existir para cada JVM à medida que escala. Quando estiver implementando inicialmente uma grade de

dados, muitas partições poderão ser usadas.

Cenários: Número de Shards por JVM

Cenário: pequeno número de shards para cada JVM

Os dados são incluídos e removidos de uma JVM utilizando unidades de shard. Os shards nunca são divididos em partes. Se existirem 10 GB de dados e 20 shards para conterem estes dados, cada shard conterá 500 MB de dados em média. Se nove Java Virtual Machines hospedarem a grade de dados, em média, cada JVM possuirá dois shards. Como 20 não é igualmente divisível por 9, algumas Java Virtual Machines possuem três shards, na seguinte distribuição:

- Sete Java Virtual Machines com dois shards
- Dois Java Virtual Machines com três shards

Como cada shard contém 500 MB de dados, a distribuição de dados é desigual. As sete Java Virtual Machines com dois shards contêm cada uma 1 GB de dados. As duas Java Virtual Machines com três shards possuem 50% mais dados ou 1.5 GB, o que é uma carga de memória muito maior. Como estes dois Java Virtual Machines hospedam três shards, eles também recebem 50% mais solicitações para seus dados. Como resultado, um pequeno número de shards para cada JVM causa desequilíbrio. Para aumentar o desempenho, aumente o número de shards para cada JVM.

Cenário: número aumentado de shards por JVM

Neste cenário, considere um número muito maior de shards. Neste cenário, há 101 shards com nove Java Virtual Machines hospedando 10 GB de dados. Neste caso, cada shard contém 99 MB de dados. A Java Virtual Machines possui a seguinte distribuição de shards:

- Sete Java Virtual Machines com 11 shards
- Dois Java Virtual Machines com 12 shards

As duas Java Virtual Machines com 12 shards agora possuem apenas mais 99 MB de dados do que os outros shards, o que é uma diferença de 9%. Este cenário é distribuído muito mais igualmente do que a diferença de 50% no cenário com um pequeno número de shards. De uma perspectiva de uso do processador, apenas 9% de mais trabalho existe para os dois Java Virtual Machines com os 12 shards comparado aos sete Java Virtual Machines que possuem 11 shards. Ao aumentar o número de shards em cada JVM, o uso dos dados e do processador é distribuído de uma maneira proporcional e equilibrada.

Quando você está criando seu sistema, utilize 10 shards para cada JVM em seu cenário com dimensionamento máximo ou quando o sistema está executando seu número máximo de Java Virtual Machines em seu horizonte de planejamento.

Fatores Adicionais de Posicionamento

O número de partições, a estratégia de posicionamento e o número e o tipo de réplicas são configurados na política de implementação. O número de shards que são colocados depende da política de implementação definida. Os atributos minSyncReplicas, developmentMode, maxSyncReplicas e maxAsyncReplicas afetam onde as partições e as réplicas são colocadas.

Os fatores a seguir afetam quando os shards podem ser colocados:

- Os comandos **xscmd -c suspendBalancing** e **xscmd -c resumeBalancing**.
- 7.1.1+ O arquivo de propriedade de servidor, que possui a propriedade **placementDeferralInterval** que define o número de milissegundos antes de os shards serem colocados nos servidores de contêiner.
- O atributo **numInitialContainers** na política de implementação.

Se o número máximo de réplicas não for colocado durante a primeira inicialização, réplicas adicionais poderão ser colocadas se servidores adicionais forem colocados posteriormente. Ao planejar o número de shards por JVM, o número máximo de shards primário e de réplicas depende da existência de JVMs suficientes iniciadas para suportar o número máximo de réplicas configurado. Uma réplica nunca é colocada no mesmo processo que seu primário. Se um processo for perdido, tanto o primário quanto a réplica são perdidos. Quando o atributo developmentMode é configurado como `false`, o primário e as réplicas não são colocados no mesmo servidor físico.

Particionamento

Use o particionamento para efetuar scale out de um aplicativo. É possível definir o número de partições em sua política de implementação.

Sobre Particionamento

O particionamento não é semelhante à divisão Redundant Array of Independent Disks (RAID), que fatia cada instância em todas as faixas. Cada partição hospeda os dados completos para entradas individuais. O particionamento é um meio mais efetivo para escalar, mas não é aplicável a todos os aplicativos. Os aplicativos que requerem garantias transacionais em grandes conjuntos de dados não são escalados e não podem ser particionados efetivamente. O WebSphere eXtreme Scale não suporta atualmente two-phase commit entre as partições.

Importante: Selecione o número de partições com cuidado. O número de partições definido na política de implementação afeta diretamente o número de servidores de contêiner aos quais o aplicativo pode ser escalado. Cada partição é constituída por um fragmento primário e pelo número configurado de fragmentos de réplica. A fórmula `(Number_Partitions*(1 + Number_Replicas))` é o número de contêineres que pode ser utilizado para executar o scale out de um único aplicativo.

Utilizando Partições

Uma grade de dados pode ter até milhares de partições. Uma grade de dados pode efetuar scale up do produto até o número de partições vezes o número de shards por partição. Por exemplo, se você tiver 16 partições e cada uma delas tiver um shard primário e um shard réplica, ou dois shards, então, será possível ampliar para até 32 Java Virtual Machines. Neste caso, um shard é definido para cada JVM. É necessário escolher um número razoável de partições com base no número esperado de Java Virtual Machines que provavelmente serão utilizadas. Cada shard aumenta o uso de processador e de memória do sistema. O sistema foi desenvolvido para que possa expandir-se e manipular essa sobrecarga de acordo com o número de Java Virtual Machines de servidor disponíveis.

Os aplicativos não devem usar milhares de partições se o aplicativo executar em uma grade de dados de quatro Java Virtual Machines de servidor de conteiner. O aplicativo deve ser configurado para ter um número razoável de shards para cada JVM do servidor de contêiner. Por exemplo, uma configuração inadequada seria 2.000 partições com dois shards em execução em quatro Java Virtual Machines de

contêiner. Essa configuração resultaria em 4.000 shards distribuídos em quatro Java Virtual Machines de contêiner ou em 1.000 shards por JVM de contêiner.

Uma configuração melhor é 10 shards para cada JVM de contêiner esperada. Essa configuração ainda permite expandir elasticamente 10 vezes a configuração inicial enquanto mantém um número adequado de shards por JVM de contêiner.

Considere este exemplo de ajuste de escala: atualmente há seis servidores físicos com dois Java Virtual Machinesde contêiner por servidor físico. Espera-se um crescimento para 20 servidores físicos nos próximos três anos. Com 20 servidores físicos, você tem 40 Java Virtual Machines de servidor de contêiner e escolhe 60 para ser pessimista. Você quer 4 shards por JVM de contêiner. Existem 60 contêineres potenciais, ou um total de 240 shards. Se tiver um primário e uma réplica por partição, terá 120 partições. Esse exemplo proporciona 240 shards dividido por 12 Java Virtual Machines de contêiner ou 20 shards por JVM de contêiner para uma implementação inicial, podendo efetuar scale out para 20 computadores posteriormente.

ObjectMap e Particionamento

Com a estratégia de posicionamento FIXED_PARTITION padrão, os mapas são divididos em partições e chaves hash para diferentes partições. O cliente não precisa saber à qual partição as chaves pertencem. Se um mapSet tiver vários mapas, os mapas deverão ser confirmados em transações separadas.

Entidades e Particionamento

As entidades do Entity Manager têm uma otimização que ajuda os clientes a trabalharem com entidades em um servidor. O esquema de entidade no servidor para conjunto de mapas pode especificar uma única entidade-raiz. O cliente deve acessar todas as entidades através da entidade-raiz. O gerenciador de entidades pode, então, localizar entidades relacionadas a partir dessa raiz na mesma partição, sem exigir que os mapas relacionados tenham uma chave comum. A entidade-raiz estabelece a afinidade com uma única partição. Essa partição será utilizada por todas as buscas de entidades dentro da transação uma vez estabelecida a afinidade. A afinidade pode economizar memória, visto que os mapas relacionados não precisam de uma chave comum. A entidade-raiz deve ser especificada com uma anotação de entidade modificada, conforme mostrado no exemplo a seguir:

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

Utilize a entidade para localizar a raiz do gráfico do objeto. O gráfico de objetos define os relacionamentos entre uma ou mais entidades. Cada entidade vinculada deve resolver para a mesma partição. Todas as entidades filha são assumidas como estando na mesma partição que a raiz. As entidades filhas no gráfico de objetos são acessíveis apenas a partir de um cliente da entidade raiz. Entidades-raiz são sempre necessárias nos ambientes particionados ao usar um cliente do eXtreme Scale para se comunicar com o servidor. Apenas um tipo de entidade-raiz pode ser definido por cliente. As entidades raiz não são necessárias ao usar ObjectGrids do estilo Extreme Transaction Processing (XTP) porque todos a comunicação com a partição é feita por meio de acesso direto e local, e não por meio do mecanismo de cliente e servidor.

Colocação e Partições

Há duas estratégias de posicionamento disponíveis para WebSphere eXtreme Scale: partição fixa e por contêiner. A escolha da estratégia de posicionamento afeta a forma como sua configuração de implementação coloca as partições na grade de dados remota.

Colocação de Partições Fixas

É possível configurar a estratégia de colocação no arquivo XML de política de implementação. A estratégia de disposição padrão é uma colocação de partição fixa, ativada pela configuração `FIXED_PARTITION`. O número de shards primários que são colocados entre os contêineres disponíveis é igual ao número de partições configurado com o atributo numberOfPartitions. Se você tiver configurado réplicas, o número mínimo total de shards é definido pela seguinte fórmula:`((1 shard primário + mínimo de shards síncrono) * partições definidas)`. O número máximo total de shards colocados é definido pela seguinte fórmula: `((1 shard primário + mínimo de shards síncrono + máximo de shards assíncronos) * partições)`. A implementação do WebSphere eXtreme Scale propaga esses shards para os contêineres disponíveis. As chaves de cada mapa são submetidas a hash nas partições designadas com base no total de partições definido. Eles efetuam hash da chave para a mesma partição, mesmo se a partição mover devido ao failover ou alterações do servidor.

Por exemplo, se o valor de numberPartitions for 6 e o valor minSync for 1 para MapSet1, o total de shards para esse conjunto de mapas será 12 porque cada uma das 6 partições requer uma réplica síncrona. Se três contêineres forem iniciados, o WebSphere eXtreme Scale colocará quatro shards por contêiner para o MapSet1.

Colocação por Contêiner

A estratégia de posicionamento alternativa é um posicionamento por contêiner, que é ativado com a configuração `PER_CONTAINER` para o atributo placementStrategy no elemento de conjunto de mapas no arquivo XML de implementação. Com essa estratégia, o número de shards primários colocados em cada novo contêiner é igual ao número de partições *P* configuradas. O ambiente de implementaçãoWebSphere eXtreme Scale coloca*P* réplicas de cada partição para cada contêiner restante. A configuração numInitialContainers é ignorada quando estiver usando uma colocação por contêiner. As partições ficam maiores conforme os contêineres crescem. As chaves para os mapas não são fixas em uma determinada partição nessa estratégia. O cliente é roteado para uma partição e usa um primário aleatório. Se um cliente desejar se reconectar à mesma sessão usada para localizar uma chave mais uma vez, um manipulador de sessão deverá ser usado.

Para obter mais informações, consulte o tópico sobre o uso de uma SessionHandle para roteamento no *Guia de Programação*.

Para servidores com failover ou interrompidos, o ambiente WebSphere eXtreme Scale mudará os shards primários na estratégia de colocação por contêiner se ainda contiverem dados. Se os shards estiverem vazios, eles serão descartados. Na estratégia por contêiner, os shards primários antigos não são mantidos porque os novos shards primários são colocados em cada contêiner.

O WebSphere eXtreme Scale permite posicionamento por contêiner como uma alternativa para o que puder ser chamado de estratégia de posicionamento "típica", uma abordagem de partição fixa com a chave de um Mapa com hash para uma

dessas partições. No caso por contêiner (que é configurado com PER_CONTAINER), sua implementação coloca as partições no conjunto de servidores de contêiner on-line e, automaticamente, efetua scale out ou scale in deles conforme os contêineres são incluídos ou removidos da grade de dados do servidor. Uma grade de dados com a abordagem de partição fixa funciona bem para grades baseadas em chave, na qual o aplicativo usa um objeto chave para localizar dados na grade. A seguir está uma discussão sobre a alternativa.

Exemplo de uma Grade de Dados por Contêiner

As grades de dados PER_CONTAINER são diferentes. Você especifica que a grade de dados usa a estratégia de posicionamento PER_CONTAINER com o atributo placementStrategy em seu arquivo XML de implementação. Em vez de configurar o número total de partições que você deseja na grade de dados, especifique quantas partições deseja por contêiner que for iniciado.

Por exemplo, se configurar cinco partições por contêiner, cinco novas partições anônimas primárias serão criadas quando iniciar esse servidor de contêiner e as réplicas necessárias serão criadas nos outros servidores de contêiner implementados.

A seguir há uma possível sequência em um ambiente por contêiner conforme a grade de dados cresce.

1. Iniciar contêiner C0 hospedando 5 primárias (P0 - P4).
   - C0 hospeda: P0, P1, P2, P3, P4.
2. Iniciar contêiner C1 hospedando mais 5 primárias (P5 - P9). As réplicas são balanceadas nos contêineres.
   - C0 hospeda: P0, P1, P2, P3, P4, R5, R6, R7, R8, R9.
   - C1 hospeda: P5, P6, P7, P8, P9, R0, R1, R2, R3, R4.
3. Iniciar contêiner C2 hospedando mais 5 primárias (P10 - P14). As réplicas são mais balanceadas.
   - C0 hospeda: P0, P1, P2, P3, P4, R7, R8, R9, R10, R11, R12.
   - C1 hospeda: P5, P6, P7, P8, P9, R2, R3, R4, R13, R14.
   - C2 hospeda: P10, P11, P12, P13, P14, R5, R6, R0, R1.

O padrão continua conforme mais contêineres são iniciados, criando cinco novas partições primárias toda vez e reequilibrando as réplicas nos contêineres disponíveis na grade de dados.

Nota: O WebSphere eXtreme Scale não move os shards primários quando usa a estratégia PER_CONTAINER, ele move apenas réplicas.

Lembre-se de que os números de partições são arbitrários e não têm nada a ver com chaves, portanto, não é possível utilizar roteamento baseado em chave. Se um contêiner parar, os IDs de partição criados para esse contêiner não serão mais utilizados, portanto, haverá um intervalo nos IDs de partição. No exemplo, não haveria mais as partições P5 - P9 se o contêiner C2 falhasse, deixando apenas P0 - P4 e P10 - P14, por isso o hash baseado em chave é impossível.

Usar números como cinco, ou mais provavelmente dez, para a quantia de partições por contêiner funcionará melhor se as consequências de uma falha do contêiner forem consideradas. Para propagar uniformemente o carregamento de shards hosting em toda a grade de dados, mais do que apenas uma partição será necessária para cada contêiner. Se você tivesse uma única partição por contêiner,

quando um contêiner falhasse, apenas um contêiner (aquele hospedando o shard de réplica correspondente) deveria suportar o carregamento total da primária perdida. Nesse caso, o carregamento é duplicado imediatamente para o contêiner. No entanto, se tiver cinco partições por contêiner, então cinco contêineres selecionarão o carregamento do contêiner perdido, diminuindo o impacto em oitenta por cento em cada um. O uso de várias partições por contêiner geralmente diminui o possível impacto em cada contêiner substancialmente. Mais diretamente, considere o caso em que um contêiner para inesperadamente - o carregamento da replicação desse contêiner é espalhado sobre os 5 contêineres em vez de apenas um.

Utilizando a Política por Contêiner

Vários cenários tornam a estratégia por contêiner uma configuração ideal, como com a replicação da sessão HTTP ou o estado da sessão de aplicativo. Nesse caso, um roteador HTTP designa uma sessão a um contêiner do servlet. O contêiner do servlet precisa criar uma sessão HTTP e escolher uma das 5 partições locais primárias para a sessão. O "ID" da partição escolhida é armazenado em um cookie. O contêiner do servlet agora tem acesso local ao estado da sessão que significa acesso de latência zero aos dados para esse pedido, contanto que você mantenha a afinidade da sessão. E o eXtreme Scale replica quaisquer mudanças para a partição.

Na prática, lembre-se da repercussão do caso no qual você tem várias partições por contêiner (por exemplo, 5 novamente). Certamente, com cada novo contêiner iniciado, você tem mais 5 partições primárias e mais 5 réplicas. Com o tempo, mais partições devem ser criadas, e elas não devem ser movidas ou destruídas. Mas não é assim que os contêineres se comportam de fato. Quando um contêiner é iniciado, ele hospeda 5 shards primários, que podem ser chamados de primários "iniciais", existentes nos respectivos contêineres que os criaram. Se o contêiner falhar, as réplicas se tornarão primárias e o eXtreme Scale criará mais 5 réplicas para manter a alta disponibilidade (a menos que você desative o reparo automático). Os novos primários estão em um contêiner diferente daquele que os criou, que podem ser chamados de primários "estrangeiros". O aplicativo nunca deve colocar novos estados ou sessões em um primário estrangeiro. Eventualmente, o primário estrangeiro não tem entradas, e o eXtreme Scale o exclui automaticamente, junto com suas réplicas associadas. O propósito dos primários estrangeiros é permitir que sessões existentes continuem disponíveis (mas não as novas sessões).

Um cliente ainda pode interagir com uma grade de dados que não depender de chaves. O cliente apenas inicia uma transação e armazena dados na grade de dados independentes de quaisquer chaves. Ele solicita de Session um objeto SessionHandle, um identificador serializável que permite que o cliente interaja com a mesma partição quando necessário. Para obter mais informações, consulte o tópico sobre o uso de um SessionHandle para roteamento no *Guia de Programação*. O WebSphere eXtreme Scale escolhe uma partição para o cliente da lista de partições primárias iniciais. Ele não retorna uma partição primária estrangeira. O SessionHandle pode ser serializado em um cookie HTTP, por exemplo, e depois converter o cookie novamente em um SessionHandle. Em seguida, as APIs do WebSphere eXtreme Scale podem obter um Session ligado à mesma partição novamente, usando SessionHandle.

Nota: Não é possível usar agentes para interagir com uma grade de dados PER_CONTAINER.

Vantagens

A descrição anterior é diferente de FIXED_PARTITION ou da grade de dados hash normal porque o cliente por contêiner armazena dados em um local na grade, obtém um identificador para eles e usa o identificador para acessá-los novamente. Não existe uma chave fornecida por aplicativo como existe no caso de uma partição fixa.

Sua implementação não cria uma nova partição para cada Session. Portanto, em uma implementação por contêiner, as chaves utilizadas para armazenar dados na partição devem ser exclusivas dentro dessa partição. Por exemplo, é possível fazer o cliente gerar um SessionID exclusivo e depois utilizá-lo como chave para localizar informações em Mapas nessa partição. Várias sessões do cliente interagem com a mesma partição, portanto o aplicativo precisa utilizar chaves exclusivas para armazenar dados da sessão em cada partição fornecida.

Os exemplos anteriores utilizaram 5 partições, mas o parâmetro numberOfPartitions no arquivo XML de objectgrid pode ser utilizado para especificar as partições conforme necessário. Em vez de ser por grade de dados, a configuração é por contêiner. (O número de réplicas é especificado da mesma maneira que com a política de partição fixa.)

A política por contêiner também pode ser utilizada com várias zonas. Se possível, o eXtreme Scale retorna um SessionHandle para uma partição cuja primária está localizada na mesma zona que esse cliente. O cliente pode especificar a zona como um parâmetro para o contêiner ou utilizando uma API. O ID da zona do cliente pode ser configurado por meio de `serverproperties` ou `clientproperties`.

A estratégia PER_CONTAINER para uma grade adéqua os aplicativos que armazenam o estado do tipo de conversação e não os dados orientados por banco de dados. A chave para acessar os dados seria um ID de conversa e não tem relação com um registro do banco de dados específico. Ela fornece melhor desempenho (porque as partições primárias podem ser colocadas junto com os servlets, por exemplo) e configuração mais fácil (sem precisar calcular partições e contêineres).

Transações de Partição Única e entre Grade de Dados

A maior diferença entre as soluções do WebSphere eXtreme Scale e de armazenamento de dados tradicional, como bancos de dados relacionais ou bancos de dados em memória, é o uso do particionamento, que permite que o cache seja escalado de maneira linear. Os tipos de transações importantes a serem considerados são transações de partição única e de cada partição (grade de dados cruzada).

Em geral, as interações com o cache podem ser categorizadas como transações de partição única ou transações de grade de dados cruzada, conforme abordado na seguinte seção.

Transações de Partição Única

As transações de partição única são o método preferido para interagir com os caches que são hospedados pelo WebSphere eXtreme Scale. Quando uma transação é limitada a uma única partição, por padrão, ela é limitada a uma única Java Virtual Machine e, portanto, a um único computador de servidor. Um servidor pode executar *M* número dessas transações por segundo e, se você tiver *N*

computadores, poderá executar M*N transações por segundo. Se os negócios aumentarem e você precisar executar o dobro dessas transações por segundo, poderá dobrar *N* ao adquirir mais computadores. Em seguida, é possível atender as demandas de capacidade sem alterar o aplicativo, fazer upgrade de hardware ou até mesmo usar o aplicativo off-line.

Além de permitir que o cache seja escalado de maneira significativa, as transações de partição única também maximizam a disponibilidade do cache. Cada transação depende apenas de um computador. Qualquer um dos outros (N-1) computadores podem falhar sem afetar o sucesso ou o tempo de resposta da transação. Assim, se você estiver executando 100 computadores e um deles falhar, apenas 1% das transações em andamento no momento em que esse servidor falhou é recuperado. Depois que o servidor falhar, o WebSphere eXtreme Scale relocará as partições que são hospedadas pelo servidor com falha nos outros 99 computadores. Durante esse breve período, antes de concluir a operação, os outros 99 computadores ainda poderão concluir as transações. Apenas as transações que envolveriam as partições que estão sendo relocadas são bloqueadas. Depois que o processo de failover ser concluído, o cache poderá continuar executando, totalmente operacional, a 99% de sua capacidade de rendimento original. Depois que o servidor com falha for substituído e retornado para a grade de dados, o cache voltará para 100% da capacidade de rendimento.

Transações da Grade de Dados Cruzada

Em termos de desempenho, disponibilidade e escalabilidade, as transações de grade de dados cruzada são o oposto de transações de partição única. As transações de grade de dados cruzada acessam cada partição e, portanto, cada computador na configuração. Cada computador na grade de dados é instruído a procurar alguns dados e retornar o resultado. A transação não pode ser concluída até que cada computador tenha respondido e, dessa forma, o rendimento da grade de dados inteira ficará limitado em função do computador mais lento. Incluir computadores não agiliza o computador mais lento e, assim, não melhora o rendimento do cache.

As transações de grade de dados cruzada possuem um efeito semelhante em termos de disponibilidade. Estendendo o exemplo anterior, se você estiver executando 100 servidores e um deles falhar, então, 100% das transações que estão em andamento no momento em que esse servidor falhou será recuperado. Depois que o servidor falhar, o WebSphere eXtreme Scale relocará as partições que são hospedadas por esse servidor nos outros 99 computadores. Durante esse tempo, antes de o processo de failover ser concluído, a grade de dados não pode processar nenhuma dessas transações. Depois que o processo de failover ser concluído, o cache poderá continuar executando, porém com capacidade reduzida. Se cada computador na grade de dados atender 10 partições, então 10 dos 99 computadores restantes receberão, pelo menos, uma partição extra como parte do processo de failover. Incluir uma partição extra aumenta a carga de trabalho desse computador em pelo menos 10%. Como o rendimento da grade de dados é limitado ao rendimento do computador mais lento em uma transação de grade de dados cruzada, em média, o rendimento é reduzido em 10%.

As transações de partição única são preferidas para as transações de grade de dados cruzada para efetuar scale out com um cache de objeto distribuído e altamente disponível, como o WebSphere eXtreme Scale. Aumentar o desempenho desses tipos de sistemas requer o uso de técnicas que são diferentes das metodologias relacionais tradicionais, porém é possível transformar as transações de grade de dados cruzada em transações de partição única escalável.

Boas práticas para criar modelos de dados escaláveis

As boas práticas para construir aplicativos escaláveis com produtos como o WebSphere eXtreme Scale incluem duas categorias: princípios básicos e dicas de implementação. Os princípios básicos são ideias principais que precisam ser capturadas no projeto dos próprios dados. Um aplicativo que não observa esses princípios podem não ser escalados tão bem, mesmo para as transações de linha principal. Por outro lado, as dicas de implementação são aplicadas em transações problemáticas em um aplicativo bem projetado que observa os princípios gerais para modelos de dados escaláveis.

Princípios Básicos

Algumas das maneiras importantes de otimizar a escalabilidade são conceitos ou princípios básicos que devem ser mantidos em mente.

Duplicar em vez de normalizar

O que mais deve-se ter em mente sobre os produtos como o WebSphere eXtreme Scale é que eles são designados para propagar dados entre um grande número de computadores. Se o objetivo é concluir a maioria ou todas as transações em uma única partição, o design do modelo de dados precisa garantir que todos os dados que a transação possa precisar estejam localizados na partição. Na maioria das vezes, a única maneira de fazer isso é duplicar os dados.

Por exemplo, considere um aplicativo, como um quadro de avisos. Duas transações muito importantes para um quadro de mensagens mostram todas as postagens de um determinado usuário e todas as postagens de um determinado tópico. Primeiro considere como essas transações trabalhariam com um modelo de dados normalizado que contenha um registro de usuário, um registro de tópico e um registro de postagem que contenha o texto real. Se as postagens forem particionadas com os registros do usuário, a exibição do tópico torna-se uma transação de grade cruzada e vice-versa. Os tópicos e os usuários não podem ser particionados juntos porque eles possuem um relacionamento muitos-para-muitos.

A melhor maneira de fazer com que esse quadro de avisos seja escalável é duplicar as postagens, armazenar uma cópia com o registro de tópico e uma cópia com o registro do usuário. Em seguida, a exibição das postagens de um usuário é uma transação de partição única, exibir as postagens em um tópico é uma transação de partição única e atualizar ou excluir uma postagem é uma transação de duas partições. Todas essas três transações serão escaladas de maneira linear já que o número de computadores na grade de dados aumenta.

Escalabilidade Em Vez de Recursos

O maior obstáculo a ser superado ao considerar os modelos de dados não-normalizados é o impacto que esse modelos causam nos recursos. Manter duas, três ou mais cópias de alguns dados pode parecer que muitos recursos usados são práticos. Ao se deparar com esse cenário, lembre-se dos seguintes fatos: os recursos de hardware se tornam mais baratos a cada dia. Segundo e o mais importante, o WebSphere eXtreme Scale elimina a maioria dos custos implícitos associados à implementação de mais recursos.

Os recursos devem ser medidos em termos de custo em vez de computador, como megabytes e processadores. Os armazenamentos de

dados que trabalham com dados relacionais normalizados geralmente precisam estar localizados no mesmo computador. Essa colocação necessária significa que um único computador corporativo maior precisa ser adquirido em vez de vários computadores menores. Com o hardware corporativo, um computador que executa um milhão de transações por segundo normalmente é bem mais barato que 10 computadores capazes de executar 100 mil transações por segundo cada um.

Incluir recursos também gera custos de negócios. Um negócio em crescimento normalmente pode ficar sem capacidade. Quando não houver capacidade, é necessário encerrar para mudar para um computador maior e mais rápido ou é necessário criar um segundo ambiente de produção para o qual você possa mudar. De uma das formas, custos adicionais serão acarretados na forma de negócios perdidos ou ao manter quase o dobro da capacidade necessária durante o período de transação.

Com o WebSphere eXtreme Scale, o aplicativo não precisa ser encerrado para incluir capacidade. Se seus projetos de negócios requererem 10% de capacidade a mais para o próximo ano, aumente 10% o número de computadores na grade de dados. É possível aumentar essa porcentagem sem ocorrer tempo de inatividade do aplicativo e sem adquirir capacidade em excesso.

Evitar transformações de dados

Quando estiver usando o WebSphere eXtreme Scale, os dados deverão ser armazenados em um formato que possa ser consumado diretamente pela lógica de negócios. Dividir os dados em um formato mais primitivo gera custos. A transformação precisa ser feita quando os dados forem gravados e lidos. Com os bancos de dados relacionais, essa transformação é feita sem necessidade porque os dados são definitivamente persistidos no disco muito frequentemente, mas com o WebSphere eXtreme Scale, essas transformações não precisam ser executadas. Na maioria das vezes os dados são armazenados na memória e podem, portanto, serem armazenados no formato exato em que o aplicativo precisa.

Observar essa regra de amostra ajuda a desnormalizar os dados de acordo com o primeiro princípio. O tipo mais comum de transformação dos dados de negócios é as operações JOIN que são necessárias para tornar os dados normalizados em um conjunto de resultados que se ajusta às necessidades do aplicativo. Armazenar os dados no formato correto implicitamente evita a execução dessas operações JOIN e produz um modelo de dados não-normalizado.

Eliminar consultas ilimitadas

Independente de como os seus dados são estruturados, as consultas ilimitadas não são bem escaladas. Por exemplo, não tenha uma transação que solicite uma lista de todos os itens classificados por valor. Essa transação pode funcionar inicialmente quando o número total de itens for 1.000, mas quando o número total de itens chegar a 10 milhões, a transação retornará todos os 10 milhões de itens. Se você executar essa transação, os dois resultados mais prováveis são o tempo limite da transação se esgotar ou o cliente receber um erro de falta de memória.

A melhor opção é alterar a lógica de negócios para que apenas os 10 ou 20 itens principais possam ser retornados. Essa mudança de lógica mantém o tamanho da transação gerenciável, independente de quantos itens estão no cache.

Definir esquema

A principal vantagem de normalizar os dados é que o sistema de banco de dados controla a consistência de dados em segundo plano. Quando os dados são desnormalizados para escalabilidade, esse gerenciamento de consistência de dados automático já não existe mais. É necessário implementar um modelo de dados que funcione na camada de aplicativos ou como um plug-in para a grade de dados distribuída para garantir a consistência dos dados.

Considere o exemplo do quadro de mensagens. Se uma transação remover uma postagem de um tópico, a postagem duplicada no registro do usuário precisará ser removida. Sem um modelo de dados, um desenvolvedor pode gravar o código do aplicativo para remover a postagem do tópico e se esquecer de remover a postagem do registro do usuário. Entretanto, se o desenvolvedor estiver usando um modelo de dados em vez de interagir diretamente com o cache, o método removePost no modelo de dados poderá extrair o ID do usuário da postagem, procurar pelo registro do usuário e remover a postagem duplicada em segundo plano.

Como alternativa, é possível implementar um listener que é executado na partição real que detecta a alteração no tópico e ajusta automaticamente o registro do usuário. Um listener pode ser benéfico porque o ajuste do registro do usuário pode ocorrer localmente caso a partição possua o registro do usuário ou, mesmo se o registro do usuário estiver em uma partição diferente, a transação ocorrerá entre os servidores em vez de ocorrer entre o cliente e o servidor. A conexão de rede entre os servidores provavelmente é mais rápida do que a conexão de rede entre o cliente e o servidor.

Evitar contenção

Evite cenários, como ter um contador global. A grade de dados não será escalável se um registro único estiver sendo usado por um número de vezes desproporcional, comparado com o restante dos registros. O desempenho da grade de dados será limitado pelo desempenho do computador que mantém o registro fornecido.

Nessas situações, tente dividir o registro para que ele seja gerenciado por partição. Por exemplo, considere uma transação que retorna o número total de entradas no cache distribuído. Em vez de fazer com que cada operação de inserção e remoção acesse um único registro que incrementa, faça com que o listener em cada partição controle as operações de inserção e remoção. Com o controle desse listener, a inserção e a remoção poderá se transformar nas operações de partição única.

A leitura do contador se transformará em uma operação de grade de dados cruzada, mas para a maior parte, isso já era ineficiente como uma operação de grade de dados cruzada porque o desempenho dependia do desempenho do computador que hospeda o registro.

Dicas de Implementação

Também é possível considerar as seguintes dicas para obter a melhor escalabilidade.

Índices de Procura Reversa

Considere um modelo de dados não-normalizado adequadamente em que os registros são particionados com base no número do ID do cliente. Esse

método de particionamento é a opção lógica porque quase cada operação de negócios executada com o registro do cliente usa o número de ID do cliente. Entretanto, uma transação importante que não usa o número do ID do cliente é a transação de login. É mais comum ter nomes de usuário ou endereços de e-mail para login em vez de números do ID de cliente.

A abordagem simples com o cenário de login é usar uma transação de grade de dados cruzada para localizar o registro do cliente. Conforme explicado anteriormente, essa abordagem não é escalada.

A próxima opção pode ser uma partição no nome do usuário ou e-mail. Essa opção não é viável porque todas as operações baseadas no ID do cliente se transformam em transações de grade de dados cruzada. Além disso, os clientes do seu site podem alterar o nome do usuário ou o endereço de e-mail. Produtos como o WebSphere eXtreme Scale precisam do valor usado para particionar os dados que permanecerem constantes.

A solução correta é usar um índice de consulta reversa. Com o WebSphere eXtreme Scale, um cache pode ser criado na mesma grade distribuída que o cache que mantém todos os registros do usuário. Esse cache é altamente escalável, particionado e escalável. Esse cache pode ser usado para mapear um nome de usuário ou endereço de e-mail para um ID de cliente. Esse cache transforma o login em uma operação de duas partições em vez de uma operação de grade cruzada. Esse cenário não é tão bom quanto uma transação de partição única, mas o rendimento ainda pode ser escalado linearmente conforme o número de computadores aumenta.

Computar no Momento da Gravação

Valores normalmente calculados, como médias ou totais, podem ser despendiosos para serem produzidos porque essas operações normalmente requerem leitura de um grande número de entradas. Como as leituras são mais comuns do que as gravações na maioria dos aplicativos, é eficiente calcular esses valores no momento da gravação e, em seguida, armazenar o resultado no cache. Essa prática torna as operações mais rápidas e mais escaláveis.

Campos Opcionais

Considere um registro de usuário que mantém um número de negócios, doméstico e de telefone. Um usuário pode ter todos, nenhum ou qualquer combinação desses números definida. Se os dados forem normalizados, uma tabela do usuário e um número de telefone existirão. Os números de telefone para um determinado usuário pode ser localizado usando uma operação JOIN entre as duas tabelas.

Desnormalizar esse registro não requer duplicação de dados, porque a maioria dos usuários não compartilha números de telefone. Em vez disso, slots vazios no registro do usuário devem ser permitidos. Em vez de ter uma tabela de número de telefone, inclua três atributos em cada registro do usuário, um para cada tipo de número de telefone. Essa inclusão de atributos elimina a operação JOIN e torna uma procura por número de telefone de um usuário uma operação de partição única.

Colocação de relacionamentos muitos-para-muitos

Considere um aplicativo que controla os produtos e as lojas nas quais os produtos são vendidos. Um único produto é vendido em muitas lojas e uma única loja vende muitos produtos. Suponha que esse aplicativo controla 50 grandes varejistas. Cada produto é vendido em um máximo de 50 lojas, com cada uma vendendo milhares de produtos.

Mantenha uma lista de lojas dentro da entidade do produto (organização A), em vez de manter uma lista de produtos dentro de cada entidade de loja (organização B). Observar algumas das transações que esse aplicativo teria que executar mostra o porquê a organização A é mais escalável.

Primeiro observe as atualizações. Com a organização A, remover um produto do inventário de uma loja bloqueia a entidade do produto. Se a grade de dados mantiver 10.000 produtos, apenas 1/10.000 da grade de dados precisará ser bloqueada para executar a atualização. Com a organização B, a grade de dados contém apenas 50 lojas, de modo que 1/50 da grade deverá ser bloqueada para concluir a atualização. Portanto, embora os dois possam ser considerados operações de partição única, a organização A é escalada mais eficientemente.

Agora, considerando as leituras na organização A, observar as lojas nas quais um produto é vendido é uma transação de partição única que é escalada e é rápido porque a transação transmite apenas uma pequena quantia de dados. Com a organização B, essa transação se torna uma transação de grade de dados cruzada porque cada entidade de loja deve ser acessada para ver se o produto é vendido nessa loja, que revela uma enorme vantagem de desempenho para a organização A.

Escalando com Dados Normalizados

Um uso legítimo de transações de grade de dados cruzada é escalar o processamento de dados. Se uma grade de dados tiver 5 computadores e uma transação de grade de dados cruzada for despachada, que classifica cerca de 100.000 registros em cada computador, essa transação classificará entre 500.000 registros. Se o computador mais lento na grade de dados puder executar 10 dessas transações por segundo, a grade de dados poderá classificar cerca de 5.000.000 de registros por segundo. Se os dados da grade dobrarem, cada computador deverá classificar cerca de 200.000 registros e cada transação classificará cerca de 1.000.000 de registros. Esse aumento de dados diminui o rendimento do computador mais lento para 5 transações por segundo, reduzindo, assim, o rendimento da grade de dados para 5 transações por segundo. Além disso, a grade de dados classifica entre 5.000.000 de registros por segundo.

Neste cenário, dobrar o número de computadores permite que cada computador volte para o carregamento anterior de classificação de 100.000 registros, permitindo que o computador mais lento processe 10 dessas transações por segundo. O rendimento da grade de dados permanece o mesmo nas 10 solicitações por segundo, mas agora cada transação processa 1.000.000 de registros dobrando, assim, a capacidade da grade em processar 10.000.000 de registros por segundo.

Aplicativos, como um mecanismo de procura, que precisam ser escalados tanto em termos de processamento de dados para acomodar o tamanho crescente da Internet e de rendimento para acomodar o crescimento de usuários, requer a criação de diversas grades de dados, com um round robin das solicitações entre as grades. Se você precisar efetuar scale up do rendimento, inclua computadores e outra grade de dados para solicitações de serviço. Se for necessário efetuar scale up do processamento de dados, inclua mais computadores e mantenha o número de grades de dados constante.

Ampliação de Unidades ou Pods

Embora seja possível implementar uma grade de dados sobre milhares de Java virtual machines, pode-se considerar a divisão a grade de dados em unidades ou pods para aumentar a confiabilidade e facilidade de teste de sua configuração. Um pod é um grupo de servidores que está executando o mesmo conjunto de aplicativos.

Implementando uma Grade de Dados Única Grande

Testes verificaram que o eXtreme Scale pode adicionar para mais de 1000 JVMs. Tais testes incentivam a construção de aplicativos para implementar grades de dados únicas em um grande número de caixas. Isso pode ser feito, porém não é recomendado por várias razões:

1. **Questões de orçamento:** Na realidade, seu ambiente não pode testar uma grade de dados de 1000 servidores. Entretanto, ele pode testar uma grade muito menor, por motivos de orçamento, de modo que não é necessário comprar duas vezes o hardware, principalmente para um número tão grande de servidores.
2. **Versões de aplicativo diferentes:** Exibir um grande número de caixas para cada encadeamento de teste não é considerado prático. O risco é que você não esteja testando os mesmos fatores que estaria em um ambiente de produção.
3. **Perda de dados:** A execução de um banco de dados em um único disco rígido não é confiável. Qualquer problema com o disco rígido faz você perder dados. A execução de um aplicativo em crescimento em uma grade de dados única é semelhante. Você provavelmente terá erros em seu ambiente e em seus aplicativos. Portanto, colocar todos os dados em um único sistema grande muitas vezes levará à perda de grandes quantidades de dados.

Dividindo a Grade de Dados

A divisão da grade de dados do aplicativo em pods (unidades) é uma opção mais confiável. Um pod é um grupo de servidores que executa uma pilha de aplicativos homogêneos. Os pods podem ser de qualquer tamanho, mas o ideal é que eles consistam em cerca de 20 servidores físicos. Em vez de ter 500 servidores físicos em uma grade de dados única, é possível ter 25 pods de 20 servidores físicos. Uma única versão de uma pilha de aplicativos deve ser executada em um determinado pod, mas diferentes pods podem ter suas próprias versões de uma pilha de aplicativos.

Geralmente, uma pilha de aplicativos considera os níveis dos seguintes componentes.

- Sistema Operacional
- Hardware
- JVM
- WebSphere eXtreme Scale versão
- Aplicativo
- Outros componentes necessários

Um pod é uma unidade de implementação de tamanho conveniente para testes. Em vez de ter centenas de servidores para testes, é mais prático ter 20 servidores. Nesse caso, você ainda está testando a mesma configuração que teria em uma produção. A produção utiliza grades com um tamanho máximo de 20 servidores, constituindo um pod. É possível fazer testes de tensão de um único pod e determinar sua capacidade, número de usuários, quantidade de dados e

rendimento da transação. Isso facilita o planejamento e segue o padrão de escala previsível a um custo previsível.

Configurando um Ambiente Baseado em Pod

Em casos diferentes, o pod não precisa ter necessariamente 20 servidores. O propósito do tamanho do pod são os testes práticos. Um tamanho de pod deve ser pequeno o bastante para que se encontrar problemas na produção, a fração de transações afetadas seja tolerável.

Idealmente, qualquer erro causa impacto em um único pod. Um erro teria impacto em apenas quatro por cento das transações do aplicativo em vez de 100 por cento. Além disso, os upgrades são mais fáceis porque podem ser lançados por um pod por vez. Como resultado, se um upgrade para um pod criar problemas, o usuário poderá alternar esse pod de volta para o nível anterior. Os upgrades incluem quaisquer mudanças no aplicativo, na pilha de aplicativos ou nas atualizações do sistema. Enquanto possível, os upgrades só devem alterar um único elemento da pilha por vez para tornar o diagnóstico do problema mais preciso.

Para implementar um ambiente com pods, você precisa de uma camada de roteamento acima dos pods compatíveis com versões anteriores e posteriores, caso os pods sofram upgrades de software. Além disso, você deve criar um diretório que inclua informações sobre qual pod contém dados. É possível usar outra grade de dados do eXtreme Scale para isso com um banco de dados por trás dele (de preferência usando o cenário write-behind). Isso gera uma solução de duas camadas. A camada 1 é o diretório e é utilizado para localizar qual pod trata uma transação específica. A camada 2 é composta pelos próprios pods. Quando a camada 1 identifica um pod, a configuração roteia cada transação para o servidor correto no pod, que geralmente é o servidor contendo a partição para os dados utilizados pela transação. Opcionalmente, você também pode utilizar um cache local na camada 1 para diminuir o impacto associado à consulta do pod correto.

O uso dos pods é um pouco mais complexo do que ter uma grade de dados única, mas as melhorias operacionais, de teste e de confiabilidade tornam esse uso uma parte crucial dos testes de escalabilidade.

Visão Geral de Disponibilidade

Alta Disponibilidade

Com alta disponibilidade, o WebSphere eXtreme Scale fornece redundância de dados confiáveis e detecção de falhas.

O WebSphere eXtreme Scale organiza automaticamente as grades de dados do Java Virtual Machines em uma árvore vagamente federada. O serviço de catálogo na raiz e os grupos principais que mantêm os contêineres estão nas folhas da árvore. Consulte “Arquitetura de Armazenamento em Cache: Mapas, Contêineres, Clientes e Catálogos” na página 11 para obter mais informações.

Cada grupo principal é automaticamente criado pelo serviço de catálogo em grupos de cerca de 20 servidores. Os membros do grupo principal fornecem monitoramento de funcionamento para outros membros do grupo. Também, cada grupo principal elege um membro para ser o líder para comunicar as informações do grupo para o serviço de catálogo. Limitar o tamanho do grupo principal permite o bom monitoramento de funcionamento e um ambiente altamente escalável.

Nota: Em um ambiente do WebSphere Application Server, no qual o tamanho do grupo principal pode ser alterado, oeXtreme Scale não suporta mais de 50 membros por grupo principal.

Pulsação

1. Os soquetes são mantidos abertos entre o Java Virtual Machines e se um soquete for fechado inesperadamente, esse fechamento inesperado será detectado como uma falha do peer Java Virtual Machine. Essa detecção captura casos de falha, como a Java Virtual Machine saindo rapidamente. Tal detecção permite a recuperação desses tipos de falhas normalmente em menos de um segundo.
2. Outros tipos de falhas incluem: um pânico no sistema operacional, uma falha física do servidor ou falha de rede. Essas falhas são descobertas por meio de pulsação.

Pulsações são enviadas periodicamente entre pares de processos: Quando um número fixo de pulsações está ausente, é assumida uma falha. Essa abordagem detecta falhas em N*M segundos, em que N é o número de pulsações ausentes e M é o intervalo de pulsações. Especificar diretamente M e N não é suportado. Um mecanismo de régua de controle é usado para permitir que um intervalo de combinações de M e N testadas seja usado.

Falhas

Um processo pode falhar de várias maneiras. o processo poderia falhar porque algum limite de recurso foi atingido, tal como o tamanho máximo do heap ou alguma lógica de controle de processo terminou um processo. O sistema operacional poderia falhar, fazendo com que todos os processos em execução no mesmo sistema fossem perdidos. O hardware poderia falhar, embora com menos freqüência, como a NIC (placa da interface de rede), fazendo com que o sistema operacional fosse desconectado da rede. Muitos outros pontos de falha podem ocorrer, fazendo com que o processo se torne indisponível. Neste contexto, todas estas falhas podem ser categorizadas em um dos dois tipos: falha de processo e perda de conectividade.

Falha de Processo

O WebSphere eXtreme Scale reage às falhas de processo rapidamente. Quando um processo falha, o sistema operacional é responsável pela limpeza de qualquer recurso pendente que o processo estava utilizando. Essa limpeza inclui a alocação e conectividade da porta. Quando um processo falha, um sinal é enviado para as conexões que estavam sendo utilizadas por esse processo para fechar cada conexão. Com estes sinais, uma falha de processo pode ser detectada instantaneamente por qualquer outro processo que está conectado ao processo falho.

Perda de Conectividade

A perda de conectividade ocorre quando o sistema operacional se desconecta. Como resultado, o sistema operacional não pode enviar sinais a outros processos. Há diversos motivos pelos quais pode ocorrer uma perda de conectividade, mas eles podem ser divididos em duas categorias: falha de host e insulamento.

Falha de Host

Se a máquina for desconectada da tomada de energia, ela desligará instantaneamente.

Insulamento

Este cenário apresenta a condição de falha mais complicada para o software tratar corretamente porque o processo é presumido como indisponível, embora não esteja. Essencialmente, um servidor ou outro processo aparece para o sistema com falho enquanto, de fato, está executando adequadamente.

Falhas do Contêiner

Falhas contêiner geralmente são descobertas por contêineres peer através do mecanismo de grupo principal. Quando um contêiner ou conjunto de contêineres falha, o serviço de catálogo migra os fragmentos que foram hospedados nesse contêiner ou contêineres. O serviço de catálogo procura uma réplica síncrona primeiro, antes de migrar para uma réplica assíncrona. Depois da migração dos fragmentos primários para os novos contêineres de host, o serviço de catálogo procura novos contêineres de host para as réplicas que agora estão faltando.

Nota: Ilhamento de contêiner - O serviço de catálogo migrará os shards a partir dos contêineres quando o contêiner for detectado como indisponível. Se esses contêineres se tornarem disponíveis, o serviço de catálogo considerará os contêineres elegíveis para disposição como no fluxo de inicialização normal.

Latência de detecção de falha do contêiner

As falhas podem ser categorizadas em falhas brandas e falhas graves. Falhas brandas normalmente são causadas quando um processo falha. Tais falhas são detectadas rapidamente pelo sistema operacional, que pode recuperar recursos usados, como soquetes de rede. A detecção de falha típica para falhas brandas é de menos de um segundo. A falhas graves podem demorar até 200 segundos para serem detectadas com o ajuste de pulsação padrão. Tais falhas incluem: travamentos da máquina física, desconexões de cabo de rede ou falhas do sistema operacional. O tempo de execução depende da pulsação para detectar falhas graves que podem ser configuradas.

Falha de Serviço de Catálogo

Porque a grade de serviço do catálogo é uma grade do eXtreme Scale, ela também usa o mecanismo de agrupamento principal da mesma forma que o processo de falha de contêiner. A principal diferença é que o domínio do serviço de catálogo usa um processo de eleição de peer para definir o shard principal em vez do algoritmo do serviço de catálogo usado para os contêineres.

O serviço de posicionamento e o serviço de agrupamento principal são Um dos serviços N. Um serviço Um de N é executado em um membro do grupo de alta disponibilidade. O serviço de local e a administração são executados em todos os membros do grupo de alta disponibilidade. O serviço de disposição e o serviço de agrupamento principal são singletons porque são responsáveis pela configuração do sistema. O serviço de local e a administração são serviços somente leitura e existe em qualquer lugar para fornecer escalabilidade.

O serviço de catálogo usa a replicação para tornar-se tolerante a falhas. Se um processo de serviço de catálogo falhar, o serviço será reiniciado para restaurar o sistema para o nível desejado de disponibilidade. Se todos os processos que

estiverem hospedando o serviço de catálogo falharem, a grade de dados perderá dados críticos. Essa falha resulta em um reinício necessário de todos os servidores de contêiner. Como o serviço de catálogo pode ser executado em muitos processos, essa falha é um evento improvável. Entretanto, se você estiver executando todos os processos em uma única caixa, dentro de um único chassi de blade, ou de um único comutador de rede, será mais provável que uma falha ocorra. Tente remover os modos de falha comuns das caixas que estão hospedando o serviço de catálogo para reduzir a possibilidade de falha.

Várias Falhas de Contêiner

Uma réplica nunca é colocada no mesmo processo que seu primário porque, se o processo for perdido, isso resultará na perda do primário e da réplica. Em um ambiente de implantação em uma única máquina, talvez você queira ter dois contêineres e replicar entre eles. É possível definir o atributo de modo de desenvolvimento na política de desenvolvimento para configurar uma réplica a ser colocada na mesma máquina que o primário. Entretanto, na produção, o uso de uma única máquina não é suficiente porque a perda de tal host resulta na perda de ambos os servidores de contêiner. Para alterar entre o modo de desenvolvimento em uma máquina única e um modo de produção com várias máquinas, desative o modo de desenvolvimento no arquivo de configuração da política de implementação.

Tabela 4. Resumo de Descoberta de Falha e Recuperação

Tipo de perda	Mecanismo de descoberta (detecção)	Método de recuperação
Perda de processo	E/S	Reiniciar
Perda de servidor	Pulsação	Reiniciar
Interrupção da rede	Pulsação	Reestabelecer rede e conexão
Interrupção do lado do servidor	Pulsação	Parar e reiniciar servidor
Servidor ocupado	Pulsação	Aguardar até servidor estar disponível

Replicação para Disponibilidade

A replicação fornece tolerância a falhas e aumenta o desempenho para uma topologia eXtreme Scale distribuída. A replicação é ativada associando mapas de apoio a um conjunto de mapas.

Sobre Conjuntos de Mapas

Um conjunto de mapas é uma coleção de mapas que são categorizados por uma chave da partição. Esta chave da partição é derivada da chave no mapa individual obtendo o número de partições de seu modulo de hash. Se um grupo de mapas dentro do conjunto de mapas tiver a chave da partição X, esses mapas serão armazenados em uma partição X correspondente na grade de dados. Se um outro grupo possuir a chave da partição Y, todos os mapas serão armazenados na partição Y e assim por diante. Os dados dentro dos mapas são replicados com base na política definida no conjunto de mapas. A replicação ocorre nas topologias distribuídas.

Os conjuntos de mapas são designados ao número de partições e a uma política de replicação. A configuração da replicação do conjunto de mapas identifica o número de shards de réplica síncrona e assíncrona para o conjunto de mapas, além do shard primário. Por exemplo, se uma réplica síncrona e uma assíncrona existirem, todos os BackingMaps que foram designados ao conjunto de mapas terão um shard de réplica cada distribuído automaticamente dentro do conjunto de servidores de contêineres disponíveis para a grade de dados. A configuração de replicação também deve ativar clientes para ler dados a partir de servidores replicados de maneira síncrona. Isto pode propagar o carregamento para pedidos de leitura sobre servidores adicionais no eXtreme Scale. A replicação tem um impacto no modelo de programação somente ao pré-carregar os mapas de apoio.

Pré-carregamento de Mapas

Mapas podem ser associados aos utilitários de carga. Um utilitário de carga é utilizado para buscar objetos quando eles não podem ser localizados no mapa (uma ocorrência de cache) e também para gravar alterações em um backend quando ocorre o commit de uma transação. Os Utilitários de Carga também podem ser utilizados para pré-carregar dados para um mapa. O método preloadMap da interface Loader é chamado em cada mapa quando sua partição correspondente no conjunto de mapas se torna um primário. O método preloadMap não é chamado nas réplicas. Ele tenta carregar todos os dados referenciados destinados a partir do backend no mapa utilizando a sessão fornecida. O mapa relevante é identificado pelo argumento BackingMap que é transmitido ao método preloadMap.

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

Pré-carregando no Conjunto de Mapas Particionados

Os mapas podem ser particionados em N partições. Portanto, os mapas podem ser divididos em vários servidores, com cada entrada identifica por uma chave que é armazenada apenas em um destes servidores. Mapas muitos grandes podem ser mantidos em um eXtreme Scale porque o aplicativo não está mais limitado pelo tamanho de heap de uma JVM única para conter todas as entradas de um Mapa. Aplicativos que desejam pré-carregar com o método preloadMap da interface do Utilitário de Carga deve identificar o subconjunto de dados que ele pré-carrega. Sempre existe um número fixo de partições. É possível determinar este número utilizando o seguinte exemplo de código:

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

Este exemplo de código mostra que um aplicativo pode identificar o subconjunto dos dados a serem pré-carregados a partir do banco de dados. Os aplicativos sempre devem utilizar estes métodos mesmo quando o mapa não é inicialmente particionado. Estes métodos permitem flexibilidade: Se o mapa for posteriormente particionado pelos administradores, então, o utilitário de carga continua a funcionar corretamente.

O aplicativo deve emitir consultas para recuperar o subconjunto *myPartition* a partir do backend. Se um banco de dados for utilizado, então, pode ser mais fácil ter uma coluna com o identificador de partições para um determinado registro, a menos que haja alguma consulta natural que permita que os dados na tabela sejam particionados facilmente.

Desempenho

A implementação de pré-carregamento copia dados do backend para o mapa, armazenando vários objetos no mapa em uma única transação. O número ideal de registros a serem armazenados por transação depende de vários fatores, incluindo complexidade e tamanho. Por exemplo, após a transação incluir blocos de mais de 100 entradas, o benefício do desempenho diminui conforme você aumenta o número de entradas. Para determinar o número ideal, comece com 100 entradas e, em seguida, aumente o número até que não sejam mais percebidos ganhos de desempenho. Transações maiores resultam em melhor desempenho de replicação. Lembre-se, apenas o primário executa o código de pré-carregamento. Os dados pré-carregados são replicados do primário para quaisquer réplicas que estão on-line.

Pré-carregando Conjuntos de Mapas

Se o aplicativo usar um conjunto de mapas com diversos mapas, cada mapa terá seu próprio carregador. Cada utilitário de carga possui um método preload. Cada mapa é carregado serialmente pelo eXtreme Scale. Pode ser mais eficiente pré-carregar todos os mapas, projetando um único mapa como o mapa de pré-carregamento. Esse processo é uma convenção do aplicativo. Por exemplo, dois mapas, department e employee, podem utilizar o Utilitário de Carga de department para pré-carregar os mapas department e employee. Isto assegura que, transacionalmente, se um aplicativo desejar um departamento, os funcionários desse departamento estarão no cache. Quando o Utilitário de Carga do departamento pré-carregar um departamento do backend, ele também buscará os funcionários para esse departamento. O objeto department e seus objetos employee associados são, então, incluídos no mapa utilizando uma transação única.

Pré-carregamento Recuperável

Alguns clientes têm conjuntos de dados muito grandes que precisam ser armazenados em cache. O pré-carregamento de dados pode consumir muito tempo. Às vezes, o pré-carregamento deve ser concluído antes de o aplicativo ficar on-line. É possível beneficiar-se ao tornar o pré-carregamento recuperável. Supunha que haja um milhão de registros para pré-carregar. O primário está pré-carregando-os e falha no registro de número 800.000. Normalmente, a réplica escolhida para ser o novo primário limpa qualquer estado replicado e começa do início. O eXtreme Scale pode utilizar uma interface ReplicaPreloadController. O utilitário de carga para o aplicativo também precisa implementar a interface ReplicaPreloadController. Este exemplo inclui um método único no Utilitário de Carga: `Status checkPreloadStatus(Session session, BackingMap bmap);`. Este método é chamado pelo tempo de execução do eXtreme Scale antes do método preload da interface do Utilitário de Carga ser chamada normalmente. O eXtreme Scale testa o resultado deste método (Status) para determinar seu comportamento sempre que uma réplica é promovida para um primário.

Tabela 5. Valor de Status e Resposta

Valor do Status Retornado	Resposta do eXtreme Scale
Status.PRELOADED_ALREADY	O eXtreme Scale não chama o método preload porque este valor do status indica que o mapa foi totalmente pré-carregado.
Status.FULL_PRELOAD_NEEDED	O eXtreme Scale limpa o mapa e chama o método preload normalmente.
Status.PARTIAL_PRELOAD_NEEDED	O eXtreme Scale deixa o mapa no estado em que se encontra e chama o pré-carregamento. Essa estratégia permite que o Utilitário de Carga do aplicativo continue o pré-carregamento desse ponto em diante.

Claramente, enquanto um primário está pré-carregando o mapa, ele deve deixar algum estado em um mapa no conjunto de mapas que está sendo replicado para que a réplica determine qual status retornar. É possível utilizar um mapa extra denominado, por exemplo, RecoveryMap. Este RecoveryMap deve fazer parte do mesmo conjunto de mapas que está sendo pré-carregado para assegurar que o mapa seja replicado de forma consistente com os dados que estão sendo pré-carregados. A seguir, está uma implementação sugerida.

À medida que ocorre o commit de cada bloco de registros, o processo também atualiza um contador ou valor no RecoveryMap como parte de tal transação. Os dados pré-carregados e os dados de RecoveryMap são replicados atomicamente para as réplicas. Quando a réplica é promovida para o primário, ela pode verificar o RecoveryMap para saber o que aconteceu.

O RecoveryMap pode conter uma única entrada com a chave de estado. Se nenhum objeto existir para esta chave, será necessário um pré-carregamento (checkPreloadStatus returns FULL_PRELOAD_NEEDED) integral. Se um objeto existir para esta chave de estado e o valor for COMPLETE, o pré-carregamento é concluído e o método checkPreloadStatus retorna PRELOADED_ALREADY. Caso contrário, o objeto de valor indica onde o pré-carregamento reinicia e o método checkPreloadStatus retorna: `PARTIAL_PRELOAD_NEEDED`. O utilitário de carga pode armazenar o ponto de recuperação em uma variável de instância para o utilitário de carga para que, quando o pré-carregamento for chamado, ele saiba o ponto de partida. O RecoveryMap também pode conter uma entrada por mapa se cada mapa for pré-carregado de maneira independente.

Manipulando a recuperação no modo de replicação síncrono com um Utilitário de Carga

O tempo de execução doeXtreme Scale é projetado para não perder dados com commit quando o primário falha. A seção a seguir mostra os algoritmos utilizados. Estes algoritmos se aplicam apenas quando um grupo de replicação utiliza a replicação síncrona. Um utilitário de carga é opcional.

O tempo de execução do eXtreme Scale pode ser configurado para replicar todas as alterações a partir de um primário para as réplicas de maneira síncrona. Quando uma réplica síncrona é posicionada ela recebe uma cópia dos dados existentes no shard primário. Durante este tempo, o primário continua recebendo transações e as copia na réplica assincronamente. A réplica não é considerada como estando on-line neste período.

Depois de a réplica capturar o primário, ela entre no modo peer e começa a replicação síncrona. Cada transação consolidada no primário é enviada às réplicas síncronas e o primário aguarda por uma resposta de cada réplica. Uma sequência de consolidação síncrona com um utilitário de carga no primário se parece com o conjunto e etapas a seguir:

Tabela 6. Sequência de Commit no Primário

Etapa com o Utilitário de Carga	**Etapa sem o Utilitário de Carga**
Obter bloqueios para entradas	igual
Limpar alterações no utilitário de carga	no-op
Salvar alterações no cache	igual
Enviar mudanças para réplicas e aguardar confirmação	igual

Tabela 6. Sequência de Commit no Primário (continuação)

Etapa com o Utilitário de Carga	Etapa sem o Utilitário de Carga
Confirmar para o utilitário de carga por meio do Plug-in TransactionCallback	commit do Plug-in chamado, mas não faz nada
Liberar bloqueios para entradas	igual

Observe que as alterações são enviadas para a réplica antes de serem confirmadas para o utilitário de carga. Para determinar quando ocorre o commit das alterações na réplica, revise esta sequência: No momento da inicialização, inicialize as listas tx no primário, conforme abaixo.

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

Durante o processamento de confirmação síncrona, utilize a seguinte seqüência:

Tabela 7. Processamento de Commit Síncrono

Etapa com o Utilitário de Carga	Etapa sem o Utilitário de Carga
Obter bloqueios para entradas	igual
Limpar alterações no utilitário de carga	no-op
Salvar alterações no cache	igual
Enviar mudanças com uma transação confirmada, retroceder transação para réplica e aguardar confirmação	igual
Limpar lista de transações confirmadas e de transações que receberam rollback	igual
Confirmar o utilitário de carga por meio do plug-in TransactionCallBack	A confirmação do plug-in TransactionCallBack ainda é chamada mas, geralmente, não faz nada
Se a confirmação for bem-sucedida, inclua a transação nas transações confirmadas; caso contrário, inclua nas transações que receberam rollback	no-op
Liberar bloqueios para entradas	igual

Para processamento de réplica, utilize a seguinte seqüência:

1. Receber alterações
2. Confirmar todas as transações recebidas na lista de transações confirmadas
3. Efetuar rollback de todas as transações recebidas na lista de transações que receberam rollback
4. Iniciar uma transação ou sessão
5. Aplicar alterações à transação ou sessão
6. Salvar a transação ou sessão na lista pendente
7. Retornar resposta

Observe que, na réplica, não existem interações do Utilitário de Carga enquanto ele está no modo de réplica. O primário deve enviar todas as alterações por meio do Utilitário de Carga. A réplica não faz nenhuma mudança. Um efeito secundário deste algoritmo é que a réplica sempre tem as transações, mas elas não são confirmadas, até que a próxima transação primária envie o status de confirmação destas transações. Elas são então confirmadas ou recebem rollback na réplica. Mas,

até então, as transações não são confirmadas. É possível incluir um cronômetro no primário que envia o resultado da transação após um pequeno período (alguns segundos). Esse cronômetro limita, mas não elimina, qualquer deterioração desse espaço de tempo. Este staleness é um problemas apenas ao utilizar o modo de leitura de réplica. Do contrário, a deterioração não tem impacto sobre o aplicativo.

Quando o primário falha, é provável que poucos commits ou rollback tenham ocorrido nas transações no primário, mas a mensagem nunca fez isto para a réplica com estas saídas. Quando uma réplica for promovida para o novo primário, uma de suas primeiras ações será manipular esta condição. Cada transação pendente é processada novamente junto ao novo conjunto de mapas do primário. Se houver um Utilitário de Carga, então, cada transação é fornecida para o Utilitário de Carga. Estas transações são aplicadas na ordem FIFO (primeiro a entrar, primeiro a sair) estrita. Se uma transação falhar, ela será ignorada. Se três transações estiverem pendentes, A, B e C, A poderá ser confirmada, B poderá ser retrocedida e C também poderá ser confirmada. Nenhuma transação tem impacto sobre as outras. Suponha que elas sejam independentes.

Um utilitário de carga talvez queira utilizar uma lógica um pouco diferente quando no modo recuperação de failover versus modo normal. O utilitário de carga pode saber facilmente quando está em modo de recuperação de failover, implementando a interface ReplicaPreloadController. O método checkPreloadStatus é chamado apenas quando a recuperação de failover é concluída. Portanto, se o método apply da interface do Utilitário de Carga for chamado antes do checkPreloadStatus, ele será uma transação de recuperação. Depois que o método checkPreloadStatus for chamado, a recuperação de failover estará concluída.

Equilíbrio de Carga entre Réplicas

O eXtreme Scale, a menos que configurado de outra forma, envia todos os pedidos de leitura e gravação para o servidor primário para um determinado grupo de replicação. O primário deve atender todos os pedidos de clientes. É possível permitir que pedidos de leitura sejam enviados para réplicas do primário. Enviar pedidos de leitura para as réplicas permite que o carregamento dos pedidos de leitura seja compartilhado por várias Java Virtual Machines (JVM). No entanto, utilizar réplicas para pedidos de leitura pode resultar em respostas inconsistentes.

Geralmente, o balanceamento de carga por meio de réplicas é utilizado apenas quando clientes estão armazenando dados em cache que estão sendo alterados sempre ou quando os clientes estão utilizando bloqueio pessimista.

Se os dados estiverem sendo alterados continuamente e sendo invalidados no cliente, caches locais e o primário deverão ver uma taxa relativamente alta de pedidos get de clientes como resultado. Da mesma forma, no modo de bloqueio pessimista, não existe cache local, portanto, todos os pedidos são enviados para o primário.

Se os dados forem relativamente estáticos ou se o modo pessimista não for usado, o envio de solicitações de leitura à réplica não terá um grande impacto no desempenho. A freqüência de pedidos get dos clientes com caches ativos não será alta.

Quando um cliente é iniciado pela primeira vez, seu near cache está vazio. Os pedidos de cache para esse cache vazio são redirecionados para o primário. O cache cliente obtém dados com o tempo, causando a eliminação deste carregamento de pedido. Se muitos clientes iniciarem simultaneamente, o

carregamento poderá ser significativo e a leitura da réplica poderá ser uma opção de desempenho apropriada.

Replicação do Lado do Cliente

Com o eXtreme Scale, é possível replicar um mapa de servidor em um ou mais clientes utilizando a replicação assíncrona. Um cliente pode solicitar uma cópia local somente leitura de um mapa do lado do servidor usando o método ClientReplicableMap.enableClientReplication.

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

O primeiro parâmetro é o modo de replicação. Esse modo pode ser uma replicação contínua ou uma replicação de captura instantânea. O segundo parâmetro é uma matriz de IDs de partição que representa as partições a partir das quais replicar os dados. Se o valor for nulo ou uma matriz vazia, os dados são replicados a partir de todas as partições. O último parâmetro é um listener para receber eventos de replicação de cliente. Consulte ClientReplicableMap e ReplicationMapListener na documentação da API para obter detalhes.

Depois de ativada a replicação, então o servidor começa a replicar o mapa para o cliente. O cliente eventualmente está apenas algumas transações atrás do servidor em questão de tempo.

Serviço de Catálogo de Alta Disponibilidade

Um domínio de serviço de catálogo é a grade de dados de servidores de catálogos que está usando, que retém as informações de topologia para todos os contêineres no seu ambiente do eXtreme Scale. O serviço de catálogo controla o equilíbrio e o roteamento de todos os clientes. Para implementar o eXtreme Scale como um espaço de processamento de banco de dados em memória, você deve armazenar em cluster o serviço de catálogo em um domínio do serviço de catálogo para alta disponibilidade.

Componentes do Domínio do Serviço de Catálogo

Quando múltiplos servidores de catálogo iniciam, um dos servidores é eleito como o servidor de catálogo principal que aceita pulsações do Internet Inter-ORB Protocol (IIOP) e manipula as alterações dos dados do sistema em resposta a qualquer serviço de catálogo ou as alterações de contêiner.

Quando os clientes entram em contato com qualquer um dos servidores de catálogos, a tabela de roteamento para o domínio do serviço de catálogo é propagada para os clientes por meio do contexto de serviço de Common Object Request Broker Architecture (CORBA).

Configure pelo menos três servidores de catálogos. Os servidores de catálogos devem ser instalados em nós ou em imagens de instalação separados a partir de seus servidores de contêiner para assegurar que seja possível fazer upgrade facilmente de seus servidores em uma data posterior. Se a sua configuração tem zonas, é possível configurar um servidor de catálogos por zona.

Quando um servidor e contêiner do eXtreme Scale entram em contato com um dos servidores de catálogos, a tabela de roteamento para o domínio do serviço de catálogo também é propagada para o servidor e contêiner do eXtreme Scale por meio do contexto de serviço de CORBA. Além disso, se o servidor de catálogos contactado atualmente não for o servidor de catálogos principal, o pedido é

automaticamente roteado para o servidor de catálogos principal atual e a tabela de roteamento para o servidor de catálogos é atualizada.

Nota: Um domínio do serviço de catálogo e a grade de dados do servidor de contêiner são muito diferentes. O domínio do serviço de catálogo é para alta disponibilidade dos dados do sistema. A grade de dados do servidor de contêiner é para a alta disponibilidade de dados, escalabilidade e gerenciamento de carga. Portanto, existem duas tabelas de roteamento diferentes: a tabela de roteamento para o domínio de serviço de catálogo e a tabela de roteamento para os shards da grade de dados do servidor de contêiner.

As responsabilidades do domínio de serviço de catálogo são divididas em uma série de serviços:

Gerenciador de grupo principal

O serviço de catálogo usa o gerenciador de alta disponibilidade (gerenciador HA) para agrupar processos para o monitoramento de disponibilidade. Cada agrupamento dos processos é um grupo principal. Com o eXtreme Scale, o gerenciador do grupo principal agrupa dinamicamente o processo. Estes processos são mantidos pequenos para permitir a escalabilidade. Cada grupo principal elege um líder que possui a responsabilidade adicional de enviar o status para o gerenciador do grupo principal quando membros individuais falham. O mesmo mecanismo de status é usado para descobrir quando todos os membros de um grupo falham, o que causa a falha da comunicação com o líder.

O gerenciador do grupo principal é um serviço completamente automático responsável pela organização de contêineres em pequenos grupos de servidores que são então automaticamente associados de maneira livre para criar um ObjectGrid. Quando um contêiner contata pela primeira vez o serviço de catálogo, ele aguarda para ser designado a um grupo novo ou existente. Uma implementação do eXtreme Scale consiste em vários desses grupos, e esse agrupamento é um importante ativador de escalabilidade. Cada grupo é um grupo do Java Virtual Machines que usa a pulsação para monitorar a disponibilidade dos outros grupos. Um destes membros do grupo é eleito o líder e possui uma responsabilidade adicional de retransmitir informações de disponibilidade para o serviço de catálogo para permitir reação através de realocação e encaminhamento de rotas.

Serviço de disposição

O serviço de catálogo gerencia o posicionamento de shards em todo o conjunto de servidores de contêiner disponíveis. O serviço de posicionamento é responsável pela manutenção de um equilíbrio entre os recursos físicos. O serviço de disposição é responsável pela alocação de shards individuais em seu contêiner de host. O serviço de posicionamento é executado como um serviço eleito Um de N na grade de dados, portanto, exatamente uma instância do serviço está em execução. Se tal instância falhar, outro processo é então eleito e assume. Para redundância, o estado do serviço de catálogo é replicado entre todos os servidores que estão hospedando o serviço de catálogo.

Administração

O serviço de catálogo também é o ponto de entrada lógico para administração do sistema. O serviço de catálogo hospeda um Managed Bean (MBean) e fornece URLs para o Java Management Extensions (JMX) para qualquer um dos servidores que o serviço de catálogo gerencia.

Serviço de local

O serviço de local atua como o ponto de contato para ambos os clientes que estão procurando por contêineres que hospedam o aplicativo que procuram, bem como os servidores de contêiner que registram aplicativos hospedados com o serviço de posicionamento. O serviço de local é executado em todos os membros da grade de dados para efetuar scale out desta função.

Implementação do Domínio do Serviço de Catálogo

O serviço de catálogo hospeda lógica que é tipicamente inativa durante estados estáveis. Como resultado, o serviço de catálogo influencia a escalabilidade minimamente. O serviço é baseado em centenas de serviços de contêineres que se tornam disponíveis simultaneamente. Para fins de disponibilidade, configure o serviço de catálogo em uma grade de dados.

Planejamento

Depois que um domínio do serviço de catálogo for iniciado, os membros da grade de dados se conectarão. Planeje com cuidado sua topologia de domínio do serviço de catálogo, pois não é possível modificar a configuração do domínio do serviço de catálogo no tempo de execução. Distribua a grade de dados o mais diversamente possível para evitar erros.

Iniciando um domínio do serviço de catálogo

Para obter mais informações sobre como criar um domínio de serviço de catálogo, consulte o detalhes sobre como iniciar um domínio de serviço de catálogo no *Guia de Administração*.

Conectando contêineres do eXtreme Scale integrados no WebSphere Application Server a um domínio do serviço de catálogo independente

É possível configurar contêineres do eXtreme Scale integrados em um ambiente do WebSphere Application Server para conectar a um domínio do serviço de catálogo independente.

- Os domínios de serviço de catálogo também podem ser criados no console administrativo do WebSphere Application Server. Consulte o Criando Domínios do Serviço de Catálogo no WebSphere Application Serverdetalhes sobre como criar domínios de serviço de catálogo no *Guia de Administração* para obter mais informações.
-  (reprovado) Em releases anteriores, você conectou os serviços de catálogo a um domínio do serviço de catálogo criando uma propriedade customizada. Esta propriedade ainda pode ser usada, mas é reprovada. Para obter mais informações sobre essa propriedade customizada, consulte o Iniciando e Parando Servidores em um Ambiente do WebSphere Application Server informações sobre como iniciar o processo do serviço de catálogo em um WebSphere Application Server no *Guia de Administração*..

  Nota: Conflito de nome do servidor: Como esta propriedade é usada para iniciar o servidor de catálogos do eXtreme Scale, assim como para se conectar a ele, os servidores de catálogo não devem ter o mesmo nome de nenhum servidor do WebSphere Application Server.

Consulte o “Quorums de Servidores de Catálogo” para obter mais informações.

Quorums de Servidores de Catálogo

Quando o mecanismo quorum é ativado, todos os servidores de catálogo no quorum deve estar disponíveis para que as operações de posicionamento ocorram na grade de dados.

Termos Importantes

- **Pulsação**: Um sinal que é enviado entre os servidores para informar que ele estão sendo executados.
- **Quorum**: Um grupo de servidores de catálogos que se comunicam e conduzem as operações de posicionamento na grade de dados. Esse grupo consiste de todos os servidores de catálogos na grade de dados, a menos que o mecanismo quorum seja substituído manualmente pelas ações administrativas.
- **Indisponibilidade de energia**: Uma perda temporária de conectividade entre um ou mais servidores.
- **Blecaute**: Uma perda permanente de conectividade entre um ou mais servidores.
- **Datacenter**: Um grupo de servidores localizados geograficamente que normalmente se conectam a uma rede local (LAN).
- **Zona**: Uma zona é uma opção de configuração usada para agrupar servidores que compartilham alguma característica física. Alguns exemplos de zonas para um grupo de servidores incluem: um datacenter, uma rede local, uma construção ou um piso de uma construção.

Pulsações e Detecção de Falha

Servidores de contêiner e grupos principais

O serviço de catálogo coloca servidores de contêiner em grupos principais de tamanho limitado. Um grupo principal tenta detectar a falha de seus membros. Um único membro de um grupo principal é escolhido para ser o líder do grupo principal. O líder do grupo principal informa periodicamente ao serviço de catálogo que o grupo principal está ativo e relata quaisquer alterações de associação no serviço de catálogo. Uma mudança de associação pode ser uma falha da JVM ou uma JVM recém-incluída que une o grupo principal.

Se um soquete de JVM for fechado, essa JVM será considerada não mais disponível. Cada membro do grupo principal também efetua pulsação sobre esses soquetes a uma taxa determinada pela configuração. Se uma JVM não responder a essas pulsações dentro de um período de tempo máximo configurado, a JVM será considerada como não mais disponível, acionando uma detecção de falha.

Se o serviço de catálogo marcar uma JVM de contêiner como com falha e o servidor de contêiner for posteriormente relatado como disponível, o contêiner JVM será informado a encerrar os servidores de contêiner do WebSphere eXtreme Scale. Uma JVM neste estado não é visível nas consultas do comando do utilitário

xscmd. As mensagens nos logs da JVM de contêiner indicam que a JVM de contêiner falhou. É necessário reiniciar manualmente estas JVMs.

Se o líder do grupo principal não puder entrar em contato com nenhum membro, ele continuará tentando novamente o contato com o membro.

Também é possível uma falha completa de todos os membros de um grupo principal. Se o grupo principal inteiro falhar, o serviço de catálogo é responsável por detectar essa perda.

Pulsação do domínio de serviço de catálogo

O domínio do serviço de catálogo é semelhante a um grupo principal privado com uma associação estática e um mecanismo de quorum. Ele detecta falhas da mesma forma que um grupo principal normal. Porém, o comportamento é modificado para incluir a lógica de quorum. O serviço de catálogo também usa uma configuração de pulsação menos rigorosa.

Detecção de Falha

O WebSphere eXtreme Scale detecta o fim dos processos por meio de eventos de encerramento do soquete anormais. O serviço de catálogo é informado imediatamente quando um processo é finalizado.

Para obter mais informações sobre como configurar a pulsação, consulte Ajustando a Configuração do Intervalo de Pulsação para Detecção de Failoveras informações sobre como configurar a detecção de failover no *Guia de Administração*.

Comportamento do Quorum

Normalmente, os membros do serviço de catálogo possuem conectividade completa. O domínio do serviço de catálogo é um conjunto estático de JVMs. O WebSphere eXtreme Scale espera que todos os membros do serviço de catálogo estejam on-line. Quando todos os membros estiverem on-line, o serviço de catálogo possuirá quorum. O serviço de catálogo responde aos eventos do contêiner apenas enquanto o serviço de catálogo possuir quorum.

Razões para perda de quorum

O WebSphere eXtreme Scale espera perder o quorum nos seguintes cenários:

- Falha do membro da JVM de serviço de catálogo
- Indisponibilidade de energia da rede
- Perda do datacenter

O WebSphere eXtreme Scale não perde o quorum no seguinte cenário:

- Parar uma instância do servidor de catálogos com o comando **stopOgServer** ou quaisquer outras ações administrativas. O sistema sabe que a instância do servidor parou, o que é diferente de uma falha ou indisponibilidade de energia da JVM.

Se o serviço de catálogo perder um quorum, ele espera que o quorum seja restabelecido. Enquanto o serviço de catálogo não possui um quorum, ele ignora os eventos dos servidores de contêiner. Os servidores de contêiner continuam tentando quaisquer solicitações que forem rejeitadas pelo servidor de catálogos durante esse tempo. A pulsação é suspensa até um quorum ser restabelecido.

Perda de quorum a partir de uma falha de JVM

Uma falha de servidor de catálogos causa perda de quorum. Se uma JVM falhar, o quorum deverá ser substituído o mais rápido possível. O serviço de catálogo com falha não pode unir novamente a grade de dados até que o quorum seja substituído.

Perda de quorum a partir de indisponibilidade de energia da rede

O WebSphere eXtreme Scale foi projetado para esperar a possibilidade de indisponibilidades de energia. Uma indisponibilidade de energia é quando ocorre uma perda temporária de conectividade entre os datacenters. Essas indisponibilidades de energia geralmente são temporárias e finalizadas em segundos ou minutos. Embora o WebSphere eXtreme Scale tenta manter a operação normal durante o período de indisponibilidade de energia, essa indisponibilidade é considerada um evento de falha único. A falha deve ser corrigida e, em seguida,a operação normal deve ser continuada sem ações necessárias.

Uma queda de energia de longa duração pode ser classificada como um blacaute apenas sob intervenção do usuário. Substituir o quorum em um lado da indisponibilidade de energia é necessário para que o evento seja classificado como um blecaute.

Ciclo JVM de serviço de catálogo

Se um servidor de catálogos for parado ao usar o comando **stopOgServer**, o quorum diminuirá para um servidor a menos. Os servidores restantes ainda possuirão um quorum. Reiniciar o servidor do catálogos configura o quorum de volta para o número anterior.

Consequências de perda de quorum

Se uma JVM de contêiner falhar enquanto o quorum for perdido, a recuperação não ocorrerá até que a indisponibilidade de energia seja recuperada. Em um cenário de blecaute, a recuperação não ocorre até que o comando de quorum seja executado. A perda de quorum e uma falha do contêiner são consideradas como uma falha dupla, o que é um evento raro. Devido a essa falha dupla, os aplicativos podem perder o acesso de gravação dos dados que foram armazenados na JVM com falha. Quando o quorum é restaurado, a recuperação normal ocorrerá.

Da mesma forma, se você tentar iniciar um contêiner durante um evento de perda de quorum, o contêiner não iniciará.

A conectividade completa do cliente é permitida durante a perda de quorum. Se não ocorrer nenhuma falha de contêiner ou problemas de conectividade durante o evento de perda de quorum, os clientes ainda poderão interagir completamente com os servidores de contêineres.

Se ocorrer uma indisponibilidade de energia, alguns clientes poderão não ter acesso às cópias principais ou réplicas dos dados até voltar a energia.

Novos clientes poderão iniciar porque uma JVM de serviço de catálogo deve existir em cada datacenter. Portanto, pelo menos um servidor de catálogos pode ser acessado por um cliente mesmo durante um evento de indisponibilidade de energia.

Recuperação de quorum

Se o quorum for perdido por algum motivo, quando o quorum for restabelecido, um protocolo de recuperação será executado. Quando ocorrer um evento de perda de quorum, toda a verificação de atividade dos grupos principais será suspensa e os relatórios de falha também serão ignorados. Quando o quorum retornar, o serviço de catálogo verificará todos os grupos principais para determinar imediatamente a associação. Quaisquer shards anteriormente hospedados nas JVMs de contêiner relatados como falha serão recuperados. Se os shards primários forem perdidos, as réplicas sobreviventes serão promovidas para serem os shards primários. Se os shards de réplicas foram perdidos, shards de réplicas adicionais serão criados nos que sobreviveram.

Substituindo Quorum

Substitua o quorum apenas quando ocorrer uma falha do datacenter. A perda de quorum é causada por uma falha de JVM de serviço de catálogo ou por uma indisponibilidade de energia de rede, o que deverá ser recuperado automaticamente quando a JVM de serviço de catálogo for reiniciada ou a energia de rede for restabelecida.

Os administradores são os únicos que reconhecem uma falha do datacenter. O WebSphere eXtreme Scale trata uma indisponibilidade de energia e um blecaute de forma semelhante. Você deve informar o ambiente do WebSphere eXtreme Scale de tais falhas com o comando **xscmd -c overrideQuorum**. Este comando informa ao serviço de catálogo a assumir que o quorum foi obtido com a associação atual e que uma recuperação completa ocorrerá. Ao emitir um comando de substituição de quorum, você está assegurando que as JVMs no datacenter com falha realmente falharam e que não há nenhuma chance de recuperação.

A lista a seguir considera alguns cenários para substituição de quorum. Nesse cenário, há três servidores de catálogos: A, B e C.

- **Indisponibilidade de energia:** O servidor de catálogos C é isolado temporariamente. O serviço de catálogo perde o quorum e aguarda a indisponibilidade de energia acabar. Quando a indisponibilidade de energia acabar, o servidor de catálogo C une novamente o domínio do serviço de catálogo e o quorum é restabelecido. Seu aplicativo não terá problemas durante esse tempo.
- **Falha temporária:** Durante uma falha temporária, o servidor de catálogos C falha e o serviço de catálogo perde o quorum. O quorum deve ser substituído. Depois que o quorum for restabelecido, é possível reiniciar o servidor de catálogos C. O servidor de catálogos C une o domínio do serviço de catálogo novamente quando ele for reiniciado. Seu aplicativo não terá problemas durante esse tempo.
- **Falha do datacenter:** Verifique se o datacenter falhou e se ele foi isolado na rede. Em seguida, emita o comando **xscmd -c overrideQuorum**. Os dois datacenters sobreviventes executam uma recuperação completa ao substituir os shards que estavam hospedados no datacenter com falha. O serviço de catálogo agora está em execução com um quorum completo dos servidores de catálogos A e B. O aplicativo pode ver atrasos ou exceções durante o intervalo entre o início do blecaute e quando o quorum é substituído. Depois que o quorum for substituído, a grade de dados será recuperada e a operação normal é continuada.
- **Recuperação do datacenter:** Os datacenters sobreviventes já estão em execução com o quorum substituído. Quando o datacenter que contém o servidor de

catálogos C é reiniciado, todas as JVMs no datacenter deverão ser reiniciadas. Em seguida, o servidor de catálogos C unirá novamente o domínio do serviço de catálogo existente e a configuração do quorum será revertida para a situação normal sem nenhuma intervenção do usuário.

- **Falha e indisponibilidade de energia do datacenter:** O datacenter que contém o servidor de catálogos C falha. O quorum é substituído e recuperado nos datacenters restantes. Se ocorrer uma indisponibilidade de energia entre os servidores de catálogos A e B, as regras de recuperação normal dessa indisponibilidade serão aplicadas. Depois que a indisponibilidade de energia acabar, o quorum será restabelecido e a recuperação necessária da perda do quorum ocorrerá.

Comportamento do Contêiner Durante a perda de Quorum

Os contêineres hospedam um ou mais shards. Os shards são primários ou réplicas de uma partição específica. O serviço de catálogo designa shards para um contêiner e o servidor de contêiner usa essa designação até que as novas instruções cheguem a partir do serviço de catálogo. Por exemplo, um shard primário continua tentando se comunicar com seus shards de réplica durante as indisponibilidades de energia de rede até o serviço de catálogo fornecer instruções adicionais para o shard primário.

Comportamento de réplica síncrona

O shard primário pode aceitar novas transações enquanto a conexão estiver interrompida e se o número de réplicas on-line estiver pelo menos dentro do valor de propriedade **minsync** para o conjunto de mapas. Se alguma das novas transações for processada no shard primário enquanto o link para a réplica síncrona estiver quebrado, a réplica será ressincronizada com o estado atual do shard primário quando o link for restabelecido.

Não configure a replicação síncrona entre os datacenters ou em um link do tipo WAN.

Comportamento de réplica assíncrona

Enquanto a conexão estiver interrompida, o shard primário pode aceitar novas transações. O shard primário armazena em buffer as mudanças até um limite. Se a conexão com a réplica for restabelecida antes de atingir esse limite, a réplica será atualizada com as alterações em buffer. Se esse limite for atingido, o shard primário destruirá a lista armazenada em buffer e, quando a réplica for reconectada, ela será limpa e ressincronizada.

Comportamento do Cliente Durante a Perda de Quorum

Os clientes sempre podem se conectar com o servidor de catálogos para realizar uma autoinicialização da grade de dados independente se o domínio de serviço de catálogo tiver quorum ou não. O cliente tenta se conectar com qualquer instância do servidor de catálogos para obter uma tabela de rota e, em seguida, interage com a grade de dados. A conectividade de rede pode impedir que o cliente interaja com algumas partições devido à configuração da rede. O cliente pode se conectar com réplicas locais para dados remotos se ele tiver sido configurado para tal. Os clientes não poderão atualizar os dados se a partição primária para esses dados não estiver disponível.

Réplicas e Shards

Com o eXtreme Scale, um banco de dados de memória ou shard pode ser replicado a partir de uma Java Virtual Machine (JVM) para outra. Uma parte representa uma partição que é colocada em um contêiner. Várias partes que representam diferentes partições podem existir em um único contêiner. Cada partição tem uma instância que é um shard primário e um número configurável de shards de réplica. Os shards de réplica são síncronos ou assíncronos. Os tipos e a disposição dos shards de réplica são determinados pelo eXtreme Scale utilizando uma política de implementação, que especifica o número mínimo e máximo de shards síncronos e assíncronos.

Tipos de Shard

A replicação usa três tipos de shards:

- Principal
- Réplica síncrona
- Réplica assíncrona

A parte primária recebe todas as opções insert, update e remove. A parte primária inclui e remove réplicas, replica dados para as réplicas e gerencia confirmações e recuperações de transações.

As réplicas síncronas mantêm o mesmo estado que o primário. Quando um primário replica dados para uma réplica síncrona, a transação não é confirmada, até que seja confirmada na réplica síncrona.

As réplicas assíncronas podem ou não ter o mesmo estado que o primário. Quando um primário replica dados para uma réplica assíncrona, o primário não aguarda a confirmação da réplica assíncrona.

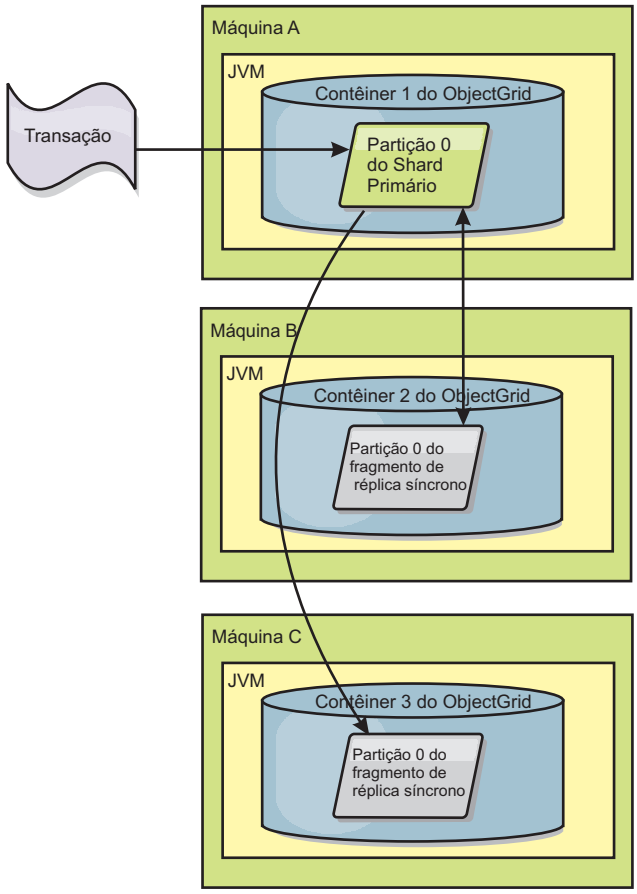


Figura 30. Caminho de Comunicação Entre um Shard Primário e Shards de Réplica

Mínimo de Shards de Réplica Síncrona

Quando um primário prepara para consolidar dados, ele verifica quantas partes de réplicas síncronas foram aprovadas para confirmar a transação. Se a transação processar normalmente na réplica, ela aprovará a confirmação. Se aconteceu algo errado na réplica síncrona, ela não aprovará a confirmação. Antes de um principal confirmar, o número de shards de réplica síncronas que são aprovadas para confirmação devem corresponder à configuração minSyncReplica da política de implementação. Quando o número de partes de réplicas síncronas sendo aprovado para confirmação for muito baixo, o primário não confirma a transação e resulta em um erro. Essa ação assegura que o número requerido de réplicas síncronas está disponível com os dados corretos. As réplicas síncronas que encontraram erros são novamente registradas para corrigir seu estado. Para obter mais informações sobre novo registro, consulte Recuperação de shard de réplica.

O primário lança um erro ReplicationVotedToRollbackTransactionException, se poucas réplicas síncronas foram aprovadas para confirmação.

Replicação e Utilitários de Carga

Normalmente, uma parte primária grava as alterações de forma síncrona por meio do Utilitário de Carga em um banco de dados. O Utilitário de Carga e o banco de dados sempre estão em sync. Quando o primário falha sobre uma parte da réplica, o banco de dados e o Utilitário de Carga podem não estar em synch. Por exemplo:

- O primário pode enviar a transação para a réplica e, em seguida, falhar antes de confirmar com o banco de dados.
- O primário pode confirmar com o banco de dados e, em seguida, falhar antes de enviar para a réplica.

A abordagem é conduzida para ou a réplica está sendo uma transação em frente de ou atrás do banco de dados. Essa situação não é aceitável. OeXtreme Scale utiliza um protocolo especial e um contrato com a implementação do Utilitário de Carga para resolver este problema sem two phase commit. O protocolo é mostrado a seguir:

Lado primário

- Envie a transação juntamente com os resultados da transação anterior.
- Grave no banco de dados e tente confirmar a transação.
- Se o banco de dados for confirmado, então confirme no eXtreme Scale. Se ele não for confirmado, recupere a transação.
- Registre a saída.

Lado da réplica

- Receba uma transação e armazene-a.
- Para todos os resultados, envie com a transação, confirme todas as transações armazenadas em buffer e descarte todas as transações recuperadas.

Lado da réplica em failover

- Para todas as transações armazenadas em buffer, forneça as transações ao Utilitário de Carga e o Utilitário de Carga tenta confirmá-las.
- O Utilitário de Carga precisa ser gravado para tornar cada transação idempotente.
- Se a transação já estiver no banco de dados, o Utilitário de Carga não realizará nenhuma operação.
- Se a transação não estiver no banco de dados, o Utilitário de Carga aplica a transação.
- Após todas as transações serem processadas, o novo primário pode começar a receber os pedidos.

Esse protocolo assegura que o banco de dados está no mesmo nível que o novo estado primário.

Posicionamento de Shard

O serviço de catálogos é responsável pela disposição dos shards. Cada ObjectGrid tem inúmeras partições, sendo que cada uma tem um shard principal e um conjunto opcional de shards de réplica. O serviço de catálogo aloca os shards equilibrando-os para que eles sejam distribuídos uniformemente nos servidores de contêiner disponíveis. Os shards de réplica e primário da mesma partição nunca são colocados no mesmo servidor de contêiner ou no mesmo endereço IP, a menos que a configuração esteja no modo de desenvolvimento.

Se um novo servidor de contêiner iniciar, o eXtreme Scale recuperará os shards a partir dos servidores de contêiner relativamente sobrecarregados para o novo servidor de contêiner vazio. Esse movimento de shards permite escala horizontal.

Efetuando Scaling Out

Efetuar scale out significa que quando servidores de contêiner extras são incluídos em uma grade de dados, o eXtreme Scale tenta mover os primários ou de réplicas existentes do conjunto de servidores de contêiner antigo para o novo conjunto. Este movimento expande a grade de dados para obter vantagem do processador, da rede e de memória dos servidores de contêiner recém-incluídos. O movimento também equilibra a grade de dados e tenta garantir que cada JVM na grade de dados hospede a mesma quantia de dados. À medida que a grade de dados se expande, cada servidor hospeda um subconjunto menor da grade total. O eXtreme Scale assume que os dados são distribuídos igualmente entre as partições. Esta expansão possibilita o scaling out.

Efetuando Scaling In

Efetuar scale in significa que se um JVM falhar, o eXtreme Scale tenta reparar o dano. Se a JVM falha tinha uma réplica, então, o eXtreme Scale substitui a réplica perdida criando uma nova réplica em uma JVM sobrevivente. Se a JVM falha tinha um principal, então o eXtreme Scale localiza a melhor réplica nos sobreviventes e promove a réplica para ser o novo principal. O eXtreme Scale, então, substitui a réplica promovida por uma nova réplica que é criada no servidores restantes. Para manter a escalabilidade, o eXtreme Scale preserva a contagem de réplica para partições à medida que os servidores falham.

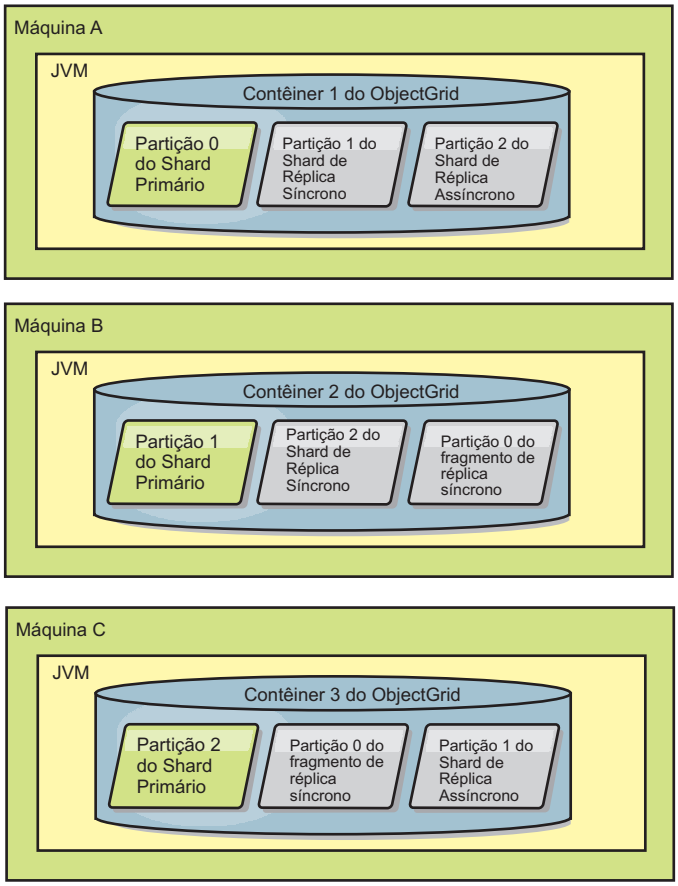


Figura 31. Posicionamento de um Conjunto de Mapas do ObjectGrid com uma Política de Implementação de 3 Partições com um Valor minSyncReplicas de 1, um Valor maxSyncReplicas de 1 e um Valor maxAsyncReplicas de 1

Lendo a partir de Réplicas

É possível configurar conjuntos de mapas a partir dos quais um cliente pode ler a partir de uma réplica em vez de limitar-se a apenas aos shards primários.

Isso normalmente pode ser vantajoso para permitir que as réplicas atendam mais que apenas shards primários potenciais no caso de falhas. Por exemplo, os conjuntos de mapas podem ser configurados para permitir que as operações de leitura sejam roteadas para réplicas ao configurar a opção replicaReadEnabled no MapSet para true. A configuração padrão é false.

Para obter mais informações sobre o elemento MapSet, consulte o tópico no arquivo XML descritor de política de implementação no *Guia de Administração*.

Ativar a leitura de réplicas pode melhorar o desempenho ao propagar os pedidos de leitura para mais Java™ virtual machines. Se a opção não estiver ativada, todos os pedidos de leitura, tais como os métodos ObjectMap.get ou Query.getResultIterator são roteados para o primário. Quando replicaReadEnabled for configurado para true, alguns pedidos poderão retornar dados obsoletos, então, um aplicativo que usa essa opção deve poder tolerar essa possibilidade. Entretanto, não ocorrerá uma perda de cache. Se os dados não estiverem na réplica, o pedido get será redirecionado para o shard primário e tentado novamente.

A opção replicaReadEnabled pode ser usada com ambas as replicações síncrona e assíncrona.

Equilíbrio de Carga entre Réplicas

Geralmente, o balanceamento de carga por meio de réplicas é utilizado apenas quando clientes estão armazenando dados em cache que estão sendo alterados sempre ou quando os clientes estão utilizando bloqueio pessimista.

O eXtreme Scale, a menos que configurado de outra forma, envia todos os pedidos de leitura e gravação para o servidor primário para um determinado grupo de replicação. O primário deve atender todos os pedidos de clientes. É possível permitir que pedidos de leitura sejam enviados para réplicas do primário. Enviar pedidos de leitura para as réplicas permite que o carregamento dos pedidos de leitura seja compartilhado por várias Java Virtual Machines (JVM). No entanto, utilizar réplicas para pedidos de leitura pode resultar em respostas inconsistentes.

Geralmente, o balanceamento de carga por meio de réplicas é utilizado apenas quando clientes estão armazenando dados em cache que estão sendo alterados sempre ou quando os clientes estão utilizando bloqueio pessimista.

Se os dados estiverem sendo alterados continuamente e sendo invalidados no cliente, caches locais e o primário deverão ver uma taxa relativamente alta de pedidos get de clientes como resultado. Da mesma forma, no modo de bloqueio pessimista, não existe cache local, portanto, todos os pedidos são enviados para o primário.

Se os dados forem relativamente estáticos ou o modo pessimista não for utilizado, o envio de pedidos de leitura de réplicas não terá um grande impacto no desempenho. A freqüência de pedidos get dos clientes com caches ativos não será alta.

Quando um cliente é iniciado pela primeira vez, seu near cache está vazio. Os pedidos de cache para esse cache vazio são redirecionados para o primário. O cache cliente obtém dados com o tempo, causando a eliminação deste

carregamento de pedido. Se houver um grande número de clientes iniciados simultaneamente, este carregamento poderá ser significativo e a leitura de réplicas poderá ser uma opção de desempenho apropriada.

Ciclos de Vida de Shard

As partes passam por diferentes estados e eventos para suportarem a replicação. O ciclo de vida de um shard inclui ficar on-line, tempo de execução, encerramento, failover e manipulação de erro. Os shards podem ser promovidos de um shard réplica para um shard primário para manipular alterações do estado do servidor.

Eventos de Ciclo de Vida

Quando as partes primárias e de réplicas são colocadas e iniciadas, elas passam por uma série de eventos para torná-las on-line e deixá-las no modo de escuta.

shard primário

O serviço de catálogo coloca uma parte primária para uma partição. O serviço de catálogo também realiza o trabalho de equilíbrio de locais de partes primárias e o início de failover para partes primárias.

Quando uma parte torna-se uma parte primária, ela recebe uma lista de réplicas do serviço de catálogo. A nova parte primária cria um grupo de réplicas e registra todas as réplicas.

Quando o principal está pronto, uma abertura para a mensagem de negócios é exibida no arquivo `SystemOut.log` para o contêiner no qual está em execução. A mensagem aberta, ou a mensagem CWOBJ1511I, lista o nome do mapa, nome do conjunto do mapa e número da partição do shard primário que iniciou.

```
CWOBJ1511I: mapName:mapSetName:partitionNumber (primário) é aberto para negócios.
```

Consulte “Posicionamento de Shard” na página 110 para obter informações adicionais sobre como o serviço de catálogo coloca shards.

Shard de Réplica

As partes da réplica são principalmente controladas pela parte primária, a não ser que a réplica detecte um problema. Durante um ciclo de vida normal, o shard primário posiciona, registra e cancela o registro de um shard de réplica.

Quando a parte primária inicializa uma parte da réplica, uma mensagem exibe o log que descreve onde a réplica é executada para indicar que está disponível a parte da réplica. A mensagem aberta, ou a mensagem CWOBJ1511I, lista o nome do mapa, nome do conjunto do mapa e número da partição do shard de réplica. Essa mensagem é mostrada a seguir:

```
CWOBJ1511I: mapName:mapSetName:partitionNumber (réplica síncrona) é aberta para negócios.
```

ou

```
CWOBJ1511I: mapName:mapSetName:partitionNumber (réplica assíncrona) é aberta para negócios.
```

Shard de réplica assíncrona: Um shard de réplica assíncrona sonda o primário para dados. A réplica automaticamente ajustará a sincronização da sondagem se não receber dados do primário, o que indica que ela é capturada com o primário. Ela também ajustará se receber um erro que possa indicar que o primário falhou ou se houver um problema de rede.

Quando a réplica assíncrona inicia a replicação, ela imprime a seguinte mensagem para o arquivo SystemOut.log para a réplica. Esta mensagem pode ser impressa mais de uma vez por mensagem CWOBJ1511. Ela será impressa novamente se a réplica conectar a um primário diferente ou se os mapas do modelo forem incluídos.

```
CWOBJ1543I: A réplica assíncrona objectGridName:mapsetName:partitionNumber iniciou ou
continuou a replicação a partir do primário. Replicando para mapas: [mapName]
```

Shard de réplica síncrona: Quando o shard de réplica síncrona é iniciado pela primeira vez, ele ainda não está no modo peer. Quando uma parte da réplica está no modo de mesmo nível, ela recebe dados da primária conforme eles chegam na primária. Antes de digitar o modo de mesmo nível, a parte da réplica precisa de uma cópia de todos os dados existentes na parte primária.

A réplica síncrona copia dados do shard primário semelhante a uma réplica assíncrona sondando os dados. Quando copia os dados existentes do primário, ela alterna para o modo peer e começa a receber dados como o primário recebe os dados.

Quando um shard de réplica alcança o modo peer, ele imprime uma mensagem no arquivo SystemOut.log para a réplica. O tempo refere-se ao período de tempo que o shard de réplica leva para obter todos os seus dados iniciais do shard primário. O tempo pode ser exibido como zero ou mínimo, se a parte primária não tiver nenhum dado existente para replicar. Esta mensagem pode ser impressa mais de uma vez por mensagem CWOBJ1511. Ela será impressa novamente se a réplica conectar a um primário diferente ou se os mapas do modelo forem incluídos.

```
CWOBJ1526I: Réplica objectGridName:mapsetName:partitionNumber:mapName entrando no modo peer
após X segundos.
```

Quando o shard de réplica síncrona está no modo peer, o shard primário deve replicar transações para todas as réplicas síncronas do modo peer. Os dados da parte da réplica síncrona permanecem no mesmo nível que os dados da parte primária. Se um número mínimo de réplicas síncronas ou minSync for configurado na política de implementação, esse número de réplicas síncronas deve ser votado para confirmação antes que a transação possa ser confirmada com êxito no primário.

Eventos de Recuperação

A replicação é projetada para recuperar a partir da falha e de eventos de erro. Se uma parte primária falhar, outra réplica será transferida. Se os erros estiverem nas partes da réplica, ela tentará a recuperação. O serviço de catálogo controla a disposição e as transações de novos shards primários ou novos shards de réplica.

O shard de réplica torna-se um shard principal

Uma parte da réplica torna-se uma parte primária por dois motivos. A parte primária parou ou falhou, ou uma decisão de equilíbrio foi tomada para mover a parte primária anterior para um novo local.

O serviço de catálogo seleciona uma nova parte primária a partir das partes de réplicas síncronas existentes. Se um movimento primário tiver que ocorrer e não houver nenhuma réplica, uma réplica temporária será colocada para concluir a transição. A nova parte primária registra todas as réplicas existentes e aceita as transações como a nova parte primária. Se as partes da réplica existentes tiverem o nível correto de dados, os dados atuais serão preservados conforme as partes de réplicas são registradas com a nova parte primária. As réplicas assíncronas serão

sondadas em relação ao novo primário.

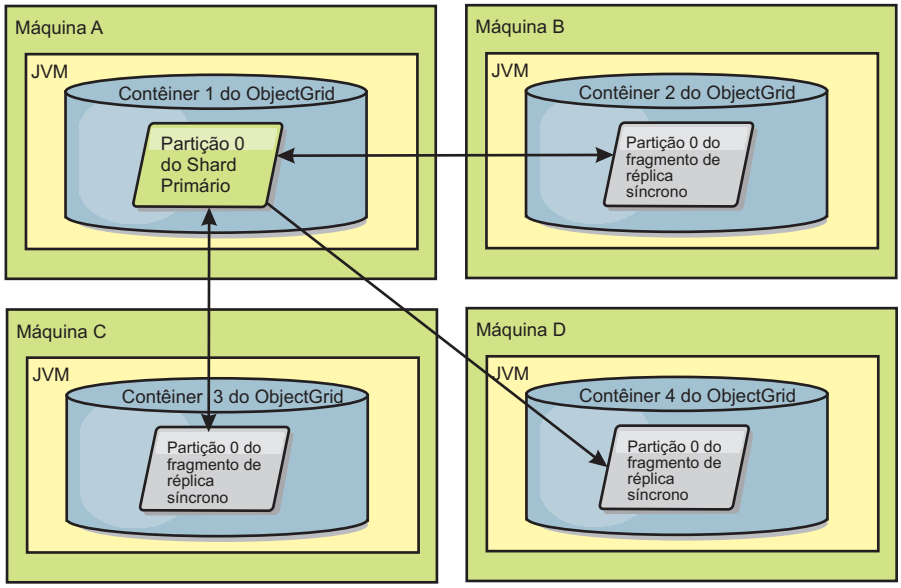


Figura 32. Posicionamento de exemplo de um conjunto de mapas do ObjectGrid para a partição partition0. A política de implementação tem um valor de minSyncReplicas de 1, um valor de maxSyncReplicas de 2, e um valor de maxAsyncReplicas de 1.

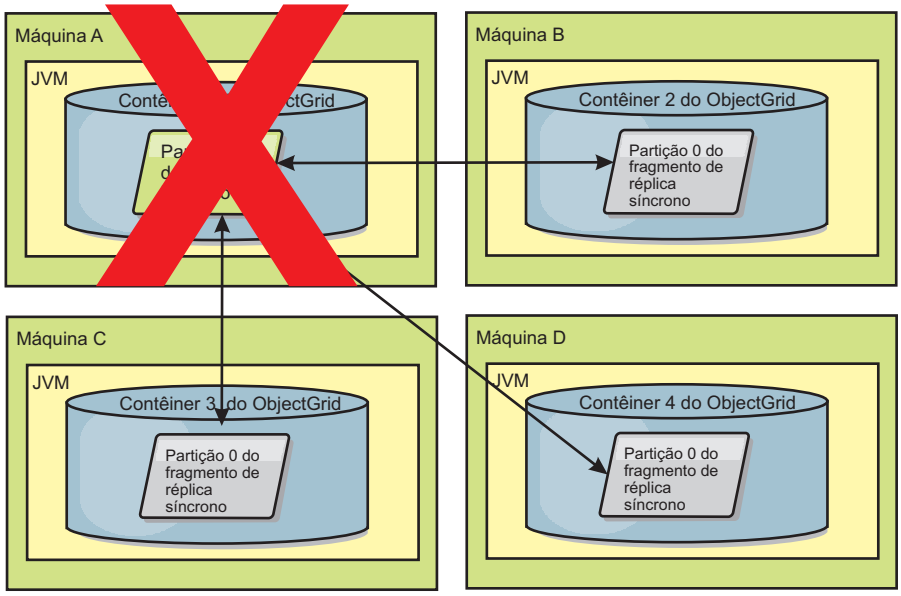


Figura 33. O contêiner para o shard primário falha

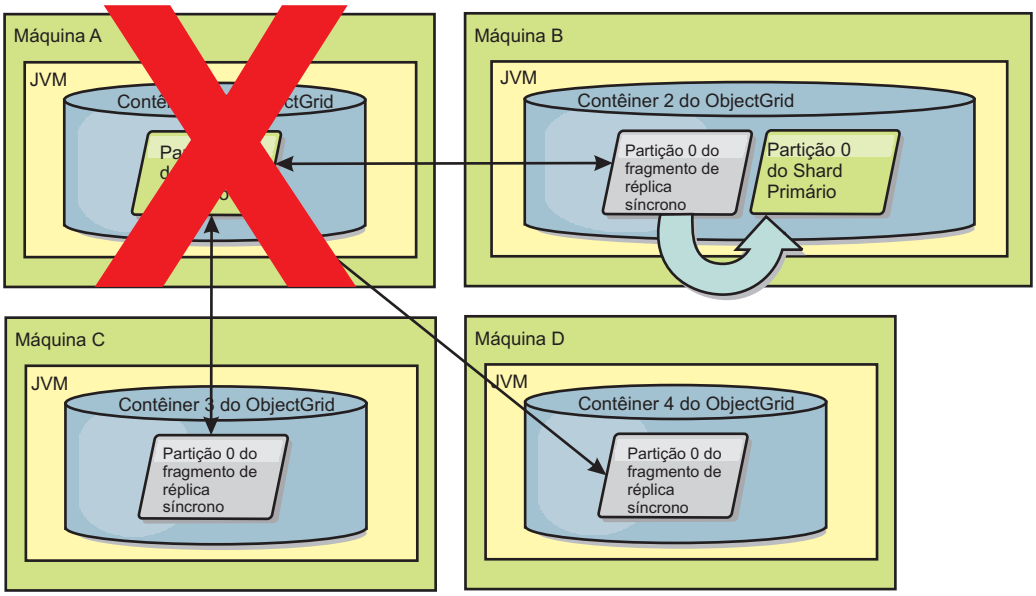


Figura 34. O Shard de Réplica Síncrona no Contêiner 2 do ObjectGrid se Torna o Shard Primário

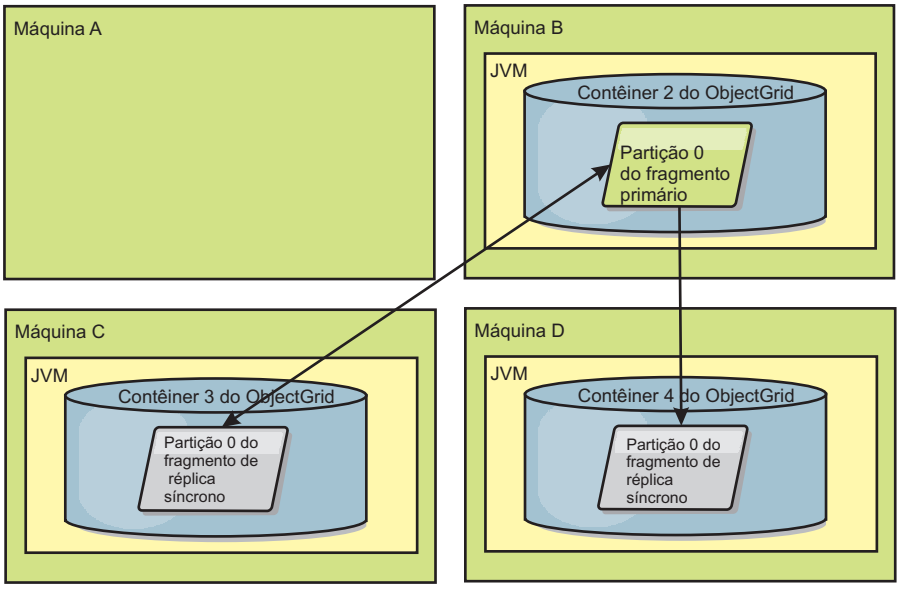


Figura 35. A máquina B contém o shard primário. Dependendo de como o modo de reparo automático é configurado e da disponibilidade dos contêineres, um novo shard de réplica síncrona pode ou não ser posicionado em uma máquina.

Recuperação de shard de réplica

Um shard de réplica síncrona é controlado pelo shard primário. No entanto, se uma réplica detectar um problema, ela poderá acionar um evento novamente registrado para corrigir o estado dos dados. A réplica remove os dados atuais e obtém uma cópia atual da primária.

Quando uma parte da réplica inicia um evento novamente registrado, a réplica copia uma mensagem de log.

```
CWOBJ1524I: Listener de réplica
objectGridName:mapSetName:partition deve registrar novamente com o shard primário.
Razão: Exceção listada
```

Se uma transação provocar um erro em uma parte da réplica durante o processamento, então a parte da réplica estará em um estado desconhecido. A transação foi processada com êxito na parte primária, mas aconteceu algo errado na réplica. Para corrigir essa situação, a réplica inicia um evento novamente registrado. Com uma nova cópia de dados da primária, a parte da réplica pode continuar. Se ocorrer novamente o mesmo problema, a parte da réplica não será novamente registrada de forma contínua. Consulte “Eventos de Falha” para obter detalhes adicionais.

Eventos de Falha

Uma réplica pode parar a replicação de dados se ela encontrar situações de erro nas quais ela não pode ser recuperada.

Muitas tentativas de registro

Se uma réplica acionar várias vezes um novo registro sem confirmar dados com êxito, ela irá parar. A parada evita que uma réplica entre em um loop de novo registro sem fim. Por padrão, uma parte da réplica tenta registrar novamente três vezes seguidas antes de parar.

Se uma parte da réplica registrar novamente muitas vezes, ela copiará a mensagem a seguir no log.

```
CWOBJ1537E: objectGridName:mapSetName:partition excedeu o número máximo
de vezes para registrar novamente (timesAllowed) sem transações com êxito.
```

Se a réplica não conseguir ser recuperada pelo novo registro, poderá existir um problema predominante com as transações relativas à parte da réplica. Um possível problema poderia ser recursos ausentes no caminho de classe, se ocorrer um erro ao aumentar as chaves ou valores da transação.

Falha ao entrar no modo peer

Se uma réplica tentar digitar o modo de mesmo nível e tiver um erro no processamento de dados existentes em massa a partir do primário (os dados do ponto de verificação), a réplica será encerrada. O encerramento evita que uma réplica seja iniciada com dados iniciais incorretos. Como ela recebe os mesmos dados do shard primário, caso seja novamente registrada, a réplica não tentará novamente.

Se uma parte da réplica falhar ao entrar no modo de mesmo nível, ela copiará a mensagem a seguir no log:

```
CWOBJ1527W A réplica objectGridName:mapSetName:partition:mapName ao entrar no modo de mesmo nível depois de numSeconds segundos.
```

Uma mensagem adicional é exibida no log que explica o motivo pelo qual a réplica falhou ao entrar no modo de mesmo nível.

Recuperação após falha ao registrar novamente ou do modo de mesmo nível

Se uma réplica falhar ao registrar novamente ou ao entrar no modo de mesmo nível, a réplica estará em um estado inativo até um novo evento de posicionamento ocorrer. Um novo evento de posicionamento pode ser um novo servidor que está iniciando ou parando. Também é possível iniciar um evento de

posicionamento usando o método triggerPlacement no Mbean PlacementServiceMBean.

Conjuntos de Mapas para Replicação

A replicação é ativada ao associar os BackingMaps a um conjunto de mapas.

Um conjunto de mapas é uma coleção de mapas que são categorizados pela partição-chave. Esta partição-chave é derivada da chave do mapa individual pegando seu do módulo de hash a quantidade de partições. Se um grupo de mapas no conjunto de mapas possuir a partição-chave X chave, estes mapas serão armazenados em uma partição X correspondente na grade de dados. Se outro grupo possuir a partição-chave Y chave, todos os mapas serão armazenados na partição Y, e assim por diante. Além disso, os dados nos mapas são replicados com base na política definida no conjunto de mapas, que é usado apenas para topologias distribuídas do eXtreme Scale (desnecessário para instâncias locais).

Consulte “Particionamento” na página 78 para obter detalhes adicionais.

Os conjuntos de mapa são designados ao número de partições que eles receberão e a uma política de replicação. A configuração da replicação do conjunto de mapas apenas identifica o número de shards de réplicas síncronas e assíncronas que um conjunto de mapas deve ter além do shard primário. Por exemplo, se for necessário possuir 1 réplica síncrona e 1 réplica assíncrona, todos os BackingMaps designados para o conjunto de mapas terão, cada um, um shard de réplica distribuído automaticamente no conjunto de contêineres disponíveis para o eXtreme Scale. A configuração de replicação também deve ativar clientes para ler dados a partir de servidores replicados de maneira síncrona. Isto pode propagar o carregamento para pedidos de leitura sobre servidores adicionais no eXtreme Scale. A replicação possui apenas um impacto no modelo de programação ao pré-carregar os BackingMaps.

Visão Geral do Processamento de Transações

O WebSphere eXtreme Scale usa transações de acordo com seu mecanismo de interação com os dados.

Para interagir com os dados, o encadeamento em seu aplicativo precisa de sua própria sessão. Quando o aplicativo desejar usar o ObjectGrid em um encadeamento, chame um dos métodos ObjectGrid.getSession para obter um encadeamento. Com a sessão, o aplicativo pode trabalhar com dados que são armazenados nos mapas ObjectGrid.

Quando um aplicativo usa um objeto de Sessão, a sessão deve estar no contexto de uma transação. Uma transação inicia e é consolidada ou inicia e é recuperada usando os métodos begin, commit e rollback no objeto de Sessão. Os aplicativos também podem trabalhar em modo de auto-consolidação, no qual a Sessão inicia e consolida automaticamente uma transação sempre que uma operação é executada no mapa. O modo de auto-confirmação não pode agrupar várias operações em uma única transação, assim, ele é a opção mais lenta se você estiver criando um lote de várias operações em uma única transação. Porém, para transações que contêm uma operação, a auto-consolidação é a opção mais rápida.

Transações

As transações possuem muitas vantagens para o armazenamento e a manipulação de dados. É possível usar as transações para proteger a grade de dados contra mudanças simultâneas, aplicar várias mudanças como uma unidade simultânea, replicar dados e implementar um ciclo de vida para bloqueios nas mudanças.

Quando uma transação inicia, o WebSphere eXtreme Scale aloca um mapa de diferença especial para conter as alterações atuais ou cópias dos pares chave e valor que a transação utiliza. Normalmente, quando um par de chave e valor é acessado, o valor é copiado antes de o aplicativo receber o valor. O mapa de diferenças controla todas as alterações de operações, como inserir, atualizar, obter, remover e assim por diante. As chaves não são copiadas porque elas são assumidas como imutáveis. Se um objeto ObjectTransformer for especificado, então, ele será utilizado para copiar o valor. Se a transação estiver utilizando o bloqueio optimistic, as imagens anteriores dos valores também serão rastreadas para comparação quando a transação for confirmada.

Se uma transação for recuperada, as informações do mapa de diferenças serão descartadas e os bloqueios nas entradas serão liberados. Quando uma transação é consolidada, as alterações são aplicadas nos mapas e os bloqueios são liberados. Se o bloqueio otimista estiver sendo utilizado, o eXtreme Scale compara as versões de imagens anteriores dos valores com os valores que estão no mapa. Esses valores devem corresponder para que a transação seja confirmada. Essa comparação permite um esquema de bloqueio de várias versões, mas a um custo de duas cópias sendo feitas quando a transação acessa a entrada. Todos os valores são copiados novamente e a nova cópia é armazenada no mapa. O WebSphere eXtreme Scale executa esta cópia para se proteger do aplicativo alterando a referência do aplicativo para o valor após um commit.

É possível evitar o uso de diversas cópias das informações. O aplicativo pode salvar uma cópia, utilizando o bloqueio pessimistic em vez do bloqueio optimistic como o custo da limitação da simultaneidade. A cópia do valor no momento da confirmação também pode ser evitada se o aplicativo concordar em não alterar um valor após uma confirmação.

Vantagens das Transações

Utilize as transações pelos seguintes motivos:

Usando as transações, você pode:

- Recuperar alterações se ocorrer uma exceção ou a lógica de negócios precisar desfazer mudanças de estado.
- Para aplicar várias alterações como uma unidade atômica no momento commit.
- Mantém e libera bloqueios em dados para aplicar múltiplas alterações como uma unidade atômica no momento da consolidação.
- Protege um encadeamento de alterações concorrentes.
- Implementa um ciclo de vida para bloqueios nas alterações.
- Produz uma unidade atômica de replicação.

Tamanho da Transação

Transações maiores são mais eficientes, especificamente para replicação. No entanto, as transações maiores podem causar impacto adverso na simultaneidade porque os bloqueios nas entradas são retidos por um período maior de tempo. Se você usar transações maiores, é possível aumentar o desempenho de replicação. Este aumento de desempenho é importante quando você estiver pré-carregando um Mapa. Experimente diferentes tipos de batch para determinar qual funciona melhor para o seu cenário.

Transações maiores também ajudam com os utilitários de carga. Se estiver sendo usado um utilitário de carga que possa executar SQL em lote, então ganhos consideráveis no desempenho são possíveis dependendo da transação e de reduções significativas de carga no lado do banco de dados. Esse ganho no desempenho depende da implementação do Carregador.

Modo de Commit Automático

Se nenhuma transação for ativamente iniciada, então quando um aplicativo interage com um objeto ObjectMap, uma operação automática é iniciada e uma consolidação é executada em nome do aplicativo. Esta operação automática de início e consolidação funciona, mas evita que a recuperação e o bloqueio funcionem efetivamente. A velocidade de replicação síncrona sofre um impacto devido ao tamanho de transação muito reduzido. Se estiver usando um aplicativo gerenciador de entidades, então não use o modo de consolidação automática pois os objetos que estiverem bloqueados com o método EntityManager.find se tornarão imediatamente não gerenciados no retorno do método e inutilizáveis.

Coordenadores de Transação Externos

Normalmente, as transações iniciam com o método session.begin e terminam com o método session.commit. Porém, quando o eXtreme Scale está incorporado, as transações podem ser iniciadas e encerradas por um coordenador externo de transações. Se você estiver usando um coordenador de transação externo, não é necessário chamar o método session.begin e terminar com o método session.commit. Se você estiver usando o WebSphere Application Server, é possível usar o plug-in WebSphereTranscationCallback.

Atributo CopyMode

É possível ajustar o número de cópias ao definir o atributo CopyMode do BackingMap ou objetos ObjectMap no arquivo XML do descritor do ObjectGrid.

É possível ajustar o número de cópias definindo o atributo CopyMode do BackingMap ou objetos ObjectMap. O modo de cópia possui os seguintes valores:

- COPY_ON_READ_AND_COMMIT
- COPY_ON_READ
- NO_COPY
- COPY_ON_WRITE
- COPY_TO_BYTES
- COPY_TO_BYTES_RAW

O valor COPY_ON_READ_AND_COMMIT é o padrão. O valor COPY_ON_READ copia os dados iniciais recuperados, mas não copia no momento da consolidação. Este modo é seguro se o aplicativo não modificar um valor depois de consolidar uma transação. O valor NO_COPY não copia os dados, que são seguros apenas para dados de leitura. Se ele nunca for alterado, não será necessário copiá-lo por motivos de isolamento.

Seja cauteloso ao usar o valor do atributo NO_COPY com mapas que possam ser atualizados. O WebSphere eXtreme Scale utiliza a cópia no primeiro acesso para permitir o retrocesso da transação. O aplicativo alterou apenas a cópia e, como resultado, o eXtreme Scale descarta a cópia. Se o valor de atributo NO_COPY for utilizado e o aplicativo modificar o valor confirmado, não será possível concluir a recuperação. Modificar o valor confirmado conduz a problemas nos índices, replicação e assim por diante porque os índices e as réplicas são atualizadas

quando a transação é confirmada. Se você modificar os dados confirmados e, em seguida, recuperar a transação, o que na realidade não é recuperada, os índices não serão atualizados e a replicação não ocorrerá. Os outros encadeamentos podem ver as alterações não confirmadas imediatamente, mesmo se tiverem bloqueios. Utilize o valor de atributo NO_COPY apenas para mapas somente leitura ou para aplicativos que concluem a cópia apropriada antes de modificar o valor. Se você utilizar o valor de atributo NO_COPY e chamar o suporte IBM com um problema de integridade de dados, será necessário reproduzir o problema com o modo de cópia definido como COPY_ON_READ_AND_COMMIT.

O valor COPY_TO_BYTES armazena os valores no mapa de maneira serializada. No tempo de leitura, o eXtreme Scale aumenta o valor de um formato serializado e, no tempo de consolidação, ele armazena o valor em um formato serializado. Com esse método, uma cópia é feita no tempo de leitura e de consolidação.

O modo de cópia padrão para um mapa pode ser configurado no objeto BackingMap. Também é possível alterar o modo de cópia antes de iniciar uma transação usando o método ObjectMap.setCopyMode.

A seguir há um exemplo de um fragmento de mapa de apoio de um arquivo `objectgrid.xml` que mostra como configurar o modo de cópia para um determinado mapa de apoio. Este exemplo assume que você esteja utilizando `cc` como o espaço de nomes `objectgrid/config`.

```
<cc:backingMap name="RuntimeLifespan" copyMode="NO_COPY"/>
```

Gerenciador de Bloqueio

Quando configurar uma estratégia de bloqueio, um gerenciador de bloqueio é criado para o mapa de apoio para manter a consistência da entrada de cache.

Configuração do Gerenciador de Bloqueios

Quando a estratégia de bloqueio pessimista ou otimista for utilizada, será criado um gerenciador de bloqueios para o BackingMap. O gerenciador de bloqueios utiliza um mapa hash para controlar entradas bloqueadas por uma ou mais transações. Se existirem muitas entradas de mapa no mapa hash, mais depósitos de bloqueio podem resultar em melhor desempenho. O risco de colisões de sincronização Java é inferior conforme a quantidade de depósitos aumenta. Mais depósitos de bloqueios também resultam em maior simultaneidade. Os exemplos anteriores mostram como um aplicativo pode configurar o número de depósitos de bloqueio para utilizar para uma determinada instância de BackingMap.

Para evitar uma exceção java.lang.IllegalStateException, o método setNumberOfLockBuckets deve ser chamado antes de chamar os métodos initialize ou getSession na instância do ObjectGrid. O parâmetro do método setNumberOfLockBuckets é um inteiro primitivo Java que especifica a quantidade de depósitos de bloqueio para uso. Utilizar um número primo permite uma distribuição uniforme de entradas do mapa sobre os depósitos de bloqueios. Um bom ponto de partida para obter melhor desempenho é configurar o número de depósitos de bloqueios para aproximadamente dez por cento do número esperado de entradas do BackingMap.

Estratégias de Bloqueio

As estratégias de bloqueio incluem pessimista, otimista e nenhum. Para escolher uma estratégia de bloqueio, é necessário considerar questões como a porcentagem de cada tipo de operações que você tem, se você utiliza um utilitário de carga, entre outras.

Os bloqueios são limitados pelas transações. É possível especificar as seguintes configurações de bloqueio:

- **Ausência de bloqueio**: Executar sem a configuração de bloqueio é a opção mais rápida. Se estiver utilizando dados de leitura, você talvez não precise do bloqueio.
- **Bloqueio pessimistic**: Adquire bloqueios em entrada e, em seguida, contém o bloqueio até o momento do commit. Essa estratégia de bloqueio fornece boa consistência à custa do rendimento.
- **Bloqueio optimistic**: Obtém uma imagem anterior de cada registro que a transação acessa e compara a imagem com os valores de entrada atuais quando ocorre o commit da transação. Se os valores de entrada forem alterados, a transação será recuperada. Nenhum bloqueio será retido até o momento da confirmação. Esta estratégia de bloqueio fornece melhor simultaneidade do que a estratégia pessimista, no risco da transação sendo recuperada e no custo da memória de criar a cópia extra da entrada.

Configure a estratégia de bloqueio no BackingMap. Não é possível alterar a estratégia de bloqueio para cada transação. A seguir há um exemplo de fragmento XML que mostra como configurar o modo de bloqueio em um mapa utilizando o arquivo XML, assumindo que `cc` é o espaço de nomes para o espaço de nomes `objectgrid/config`.

```
<cc:backingMap name="RuntimeLifespan" lockStrategy="PESSIMISTIC" />
```

Bloqueio Pessimista

Use a estratégia de bloqueio pessimista para ler e gravar mapas quando outras estratégias de bloqueio não foram possíveis. Quando um mapa do ObjectGrid map é configurado para utilizar a estratégia de bloqueio pessimista, um bloqueio de transação pessimista para uma entrada de mapa é obtido quando uma transação primeiro obtém a entrada do BackingMap. O bloqueio pessimistic fica retido até que o aplicativo conclua a transação. Geralmente, a estratégia de bloqueio pessimistic é utilizada nas seguintes situações:

- Quando o BackingMap é configurado com ou sem um utilitário de carga e as informações de controle de versões não estão disponíveis.
- Quando o BackingMap é utilizado diretamente por um aplicativo que precisa de ajuda do eXtreme Scale para controle de simultaneidade.
- Quando as informações de controle de versões estão disponíveis, mas as transações de atualização colidem freqüentemente nas entradas de suporte, resultando em falhas de atualização otimistas.

Como a estratégia de bloqueio pessimista tem o maior impacto no desempenho e escalabilidade, esta estratégia deve ser utilizada apenas para ler e gravar mapas quando outras estratégias de bloqueio não são viáveis. Por exemplo, essas situações incluem quando ocorrem falhas de atualização otimista com frequência ou quando a recuperação da falha otimista é difícil para um aplicativo manipular.

Bloqueio Otimista

A estratégia de bloqueio otimista assume que nenhuma transação em execução simultânea pode tentar atualizar a mesma entrada de mapa. Devido a esta convicção, o modo de bloqueio não precisa ser retido pelo ciclo de vida da transação porque é improvável que mais de uma transação possa atualizar a entrada de mapa simultaneamente. A estratégia de bloqueio optimistic geralmente é utilizada nas seguintes situações:

- Quando um BackingMap é configurado com ou sem um utilitário de carga e as informações de controle de versões estão disponíveis.
- Quando um BackingMap possui em sua maior parte, transações que executam operações de leitura. As operações insert, update ou remove nas entradas de mapa não ocorrem com frequência no BackingMap.
- Quando um BackingMap é inserido, atualizado ou removido mais freqüentemente do que é lido, mas as transações raramente colidem na mesma entrada do mapa.

Como a estratégia de bloqueio pessimistic, o métodos na interface ObjectMap determinam como o eXtreme Scale automaticamente tenta adquirir um modo de bloqueio para a entrada de mapa que está sendo acessada. Entretanto, existem as seguintes diferenças entre as estratégias pessimistic e optimistic:

- Como a estratégia de bloqueio pessimistic, um modo de bloqueio S é adquirido pelos métodos get e getAll quando o método é chamado. No entanto, com o bloqueio optimistic, o modo de bloqueio S não fica retido até que a transação seja concluída. Em vez disso, o modo de bloqueio S é liberado antes de o método retornar ao aplicativo. O propósito de adquirir o modo de bloqueio é para que o eXtreme Scale possa garantir que apenas dados com commit de outras transações fiquem visíveis para a transação atual. Após o eXtreme Scale ter verificado que ocorreu commit nos dados, o modo de bloqueio S é liberado. No momento do commit, uma verificação de versão optimistic é executada para garantir que nenhuma outra transação tenha alterado a entrada do mapa após a transação atual ter liberado seu modo de bloqueio S. Se uma entrada não for procurada a partir do mapa antes de ser atualizada, invalidada ou excluída, o tempo de execução do eXtreme Scale implicitamente procura a entrada a partir do mapa. Esta operação get implícita é desempenhada para obter o valor atual no momento em que foi solicitada a modificação da entrada.
- Diferente da estratégia de bloqueio pessimista, os métodos getForUpdate e getAllForUpdate são tratados exatamente como os métodos get e getAll quando a estratégia de bloqueio otimista é utilizada. Ou seja, um modo de bloqueio S é adquirido no início do método e o modo de bloqueio S é liberado antes de retornar para o aplicativo.

Todos os outros métodos ObjectMap são tratados exatamente como são tratados para a estratégia de bloqueio pessimistic. Ou seja, quando o método commit é chamado, um modo de bloqueio X é obtido para qualquer entrada do mapa que tenha sido inserida, atualizada, removida, tocada ou invalidada e o modo de bloqueio X é retido até que a transação tenha concluído o processamento de consolidação.

A estratégia de bloqueio optimistic assume que nenhuma transação em execução simultânea tenta atualizar a mesma entrada de mapa. Devido a esta suposição, o modo de bloqueio não precisa ser mantido pela duração da transação porque é improvável que mais de uma transação possa atualizar a entrada de mapa simultaneamente. Entretanto, como um modo de bloqueio não foi mantido, outra transação simultânea poderia potencialmente atualizar a entrada do mapa após a transação atual ter liberado seu modo de bloqueio S.

Para tratar esta possibilidade, o eXtreme Scale obtém um bloqueio X no momento do commit e executa uma verificação de versão optimistic para verificar se nenhuma outra transação alterou a entrada do mapa após a transação atual ter lido a entrada do mapa a partir do BackingMap. Se outra transação alterar a entrada do mapa, a verificação de versão falhará e ocorrerá uma exceção OptimisticCollisionException. Esta exceção força a transação atual a ser retrocedida

e o aplicativo deve tentar novamente a transação inteira. A estratégia de bloqueio optimistic é muito útil quando um mapa é lido em sua maior parte e é improvável que ocorram atualizações na mesma entrada do mapa.

Ausência de Bloqueio

Quando um BackingMap é configurado para usar nenhuma estratégia de bloqueio, nenhum bloqueio de transação para uma entrada de mapa é obtido.

Não utilizar uma estratégia de bloqueio é útil quando um aplicativo é um gerenciador de persistência, como um contêiner Enterprise JavaBeans (EJB) ou quando um aplicativo utiliza Hibernate para obter dados persistentes. Neste cenário, o BackingMap é configurado sem um utilitário de carga e o gerenciador de persistência utiliza o BackingMap como um cache de dados. Neste cenário, o gerenciador de persistência fornece controle de simultaneidade entre transações que estão acessando as mesmas entradas de Mapa.

O WebSphere eXtreme Scale não precisa obter nenhum bloqueio de transação para o propósito de controle de simultaneidade. Essa situação presume que o gerenciador de persistência não libera os bloqueios da transação antes de atualizar o mapa ObjectGrid com as alterações confirmadas. Se o gerenciador de persistência libera seus bloqueios, então uma estratégia de bloqueio pessimistic ou optimistic deve ser utilizada. Por exemplo, suponha que o gerenciador de persistência de um contêiner EJB esteja atualizando o mapa do ObjectGrid com dados que foram confirmados na transação gerenciada por contêiner de EJB. Se a atualização do mapa do ObjectGrid ocorrer antes dos bloqueios de transação do gerenciador de persistência serem liberados, então é possível não utilizar nenhuma estratégia de bloqueio. Se o mapa do ObjectGrid ocorrer após os bloqueios de transação do gerenciador de persistência serem liberados, será necessário utilizar a estratégia de bloqueio otimista ou pessimista.

Outro cenário onde a ausência de estratégia de bloqueio pode ser utilizada é quando o aplicativo utiliza um BackingMap diretamente e um Utilitário de Carga é configurado para o mapa. Neste cenário, o utilitário de carga utiliza o suporte de controle de simultaneidade que é fornecido por um Relational Database Management System (RDBMS) utilizando Java Database Connectivity (JDBC) ou Hibernate para acessar dados em um banco de dados relacional. A implementação do utilitário de carga pode utilizar uma abordagem optimistic ou pessimistic. Um utilitário de carga que utiliza um bloqueio optimistic ou uma abordagem de controle de versões ajuda a obter a maior quantidade de simultaneidade e desempenho. Para obter mais informações sobre como implementar uma abordagem de bloqueio otimista, consulte a seção OptimisticCallback em informações sobre as considerações do carregador no *Guia de Administração*. Se estiver usando um utilitário de carga que usa suporte de bloqueio pessimistic de um backend subjacente, é possível querer usar o parâmetro forUpdate que é transmitido no método get da interface do utilitário de carga. Configure este parâmetro como true se o método getForUpdate da interface ObjectMap foi utilizado pelo aplicativo para obter os dados. O utilitário de carga pode utilizar esse parâmetro para determinar se solicitará um bloqueio atualizável na linha que está sendo lida. Por exemplo, o DB2 obtém um bloqueio atualizável quando uma instrução select SQL contém uma cláusula `FOR UPDATE`. Esta abordagem oferece a mesma prevenção de conflito que está descrita em “Bloqueio Pessimista” na página 122.

Para obter mais informações, consulte o tópico sobre manipulação de bloqueios no *Guia de Programação* ou bloqueio de entrada de mapa no *Guia de Administração*.

Distribuindo Transações

Use Java Message Service (JMS) para mudanças de transação distribuída entre diferentes camadas ou em ambientes em plataformas mistas.

O JMS é um protocolo ideal para alterações distribuídas entre diferentes camadas ou em ambientes em plataformas mistas. Por exemplo, alguns aplicativos que usam o eXtreme Scale podem ser implementados noIBM WebSphere Application Server Community Edition, Apache Geronimo ou Apache Tomcat, considerando que outros aplicativos podem executar no WebSphere Application Server Versão 6.x. O JMS é ideal para alterações distribuídas entre peers do eXtreme Scale nesses diferentes ambientes. O transporte de mensagens do gerenciador de alta disponibilidade é muito rápido, mas pode apenas distribuir alterações para as Java Virtual Machines que estão em um grupo principal único. O JMS é mais lento, mas permite que conjuntos maiores e mais diversos de aplicativos clientes compartilhem um ObjectGrid. O JMS é ideal no compartilhamento de dados em um ObjectGrid entre um cliente Swing rápido e um aplicativo implementado no WebSphere Extended Deployment.

O Mecanismo de Invalidação do Cliente e a Replicação Ponto a Ponto incorporados são exemplos da distribuição de alterações transacionais com base no JMS. Consulte o informações sobre como configurar a replicação ponto a ponto com o JMS no *Guia de Administração* para obter mais informações.

Implementando o JMS

O JMS é implementado para distribuir alterações de transação usando um objeto Java que se comporta como um ObjectGridEventListener. Este objeto pode propagar o estado nas quatro maneiras a seguir:

1. Invalidar: Qualquer entrada que é despejada, atualizada ou excluída é removida em todas as Java Virtual Machines peer quando elas recebem a mensagem.
2. Invalidar condicional: A entrada é despejada somente se a versão local for a mesma ou mais antiga que a versão no publicador.
3. Push: Qualquer entrada que foi despejada, atualizada, excluída ou inserida é incluída ou sobrescrita em todas as Java Virtual Machines peer quando elas recebem a mensagem JMS.
4. Push condicional: A entrada é atualizada ou incluída no lado de recebimento apenas se a entrada local for menos recente que a versão que está sendo publicada.

Atender Alterações de Publicação

O plug-in implementa a interface ObjectGridEventListener para interceptar o evento transactionEnd. Quando o eXtreme Scale chama este método, o plug-in tenta converter a lista LogSequence list para cada mapa que é acessado pela transação em uma mensagem JMS e, então, a publica. O plug-in pode ser configurado para publicar alterações para todos os mapas ou um subconjunto de mapas. Os objetos LogSequence são processados para os mapas com a publicação ativada. A classe LogSequenceTransformer do ObjectGrid serializa um LogSequence filtrado para cada mapa em um fluxo. Após todas as LogSequences serem serializadas para o fluxo, então, um ObjectMessage JMS é criado e publicado em um tópico bem conhecido.

Atender Mensagens JMS e Aplicá-las ao ObjectGrid Local

O mesmo plug-in também inicia um encadeamento que gira em um loop, recebendo todas as mensagens publicadas no tópico bem conhecido. Quando chega uma mensagem, ele transmite o conteúdo da mensagem para a classe LogSequenceTransformer para convertê-lo em um conjunto de objetos LogSequence. Em seguida, uma transação não-write-through é iniciada. Cada objeto LogSequence é fornecido ao método Session.processLogSequence, que atualiza os Mapas locais com as alterações. O método processLogSequence entende o modo de distribuição. A transação é confirmada e o cache local agora reflete as alterações. Para obter mais informações sobre como usar o JMS para distribuir as mudanças transacionais, consulte o informações sobre como distribuir as mudanças entre as Java Virtual Machines equivalentes no *Guia de Administração*.

Transações de Partição Única e entre Grade de Dados

A maior diferença entre as soluções do WebSphere eXtreme Scale e de armazenamento de dados tradicional, como bancos de dados relacionais ou bancos de dados em memória, é o uso do particionamento, que permite que o cache seja escalado de maneira linear. Os tipos de transações importantes a serem considerados são transações de partição única e de cada partição (grade de dados cruzada).

Em geral, as interações com o cache podem ser categorizadas como transações de partição única ou transações de grade de dados cruzada, conforme abordado na seguinte seção.

Transações de Partição Única

As transações de partição única são o método preferido para interagir com os caches que são hospedados pelo WebSphere eXtreme Scale. Quando uma transação é limitada a uma única partição, por padrão, ela é limitada a uma única Java Virtual Machine e, portanto, a um único computador de servidor. Um servidor pode executar *M* número dessas transações por segundo e, se você tiver *N* computadores, poderá executar `M*N` transações por segundo. Se os negócios aumentarem e você precisar executar o dobro dessas transações por segundo, poderá dobrar *N* ao adquirir mais computadores. Em seguida, é possível atender as demandas de capacidade sem alterar o aplicativo, fazer upgrade de hardware ou até mesmo usar o aplicativo off-line.

Além de permitir que o cache seja escalado de maneira significativa, as transações de partição única também maximizam a disponibilidade do cache. Cada transação depende apenas de um computador. Qualquer um dos outros `(N-1)` computadores podem falhar sem afetar o sucesso ou o tempo de resposta da transação. Assim, se você estiver executando 100 computadores e um deles falhar, apenas 1% das transações em andamento no momento em que esse servidor falhou é recuperado. Depois que o servidor falhar, o WebSphere eXtreme Scale relocará as partições que são hospedadas pelo servidor com falha nos outros 99 computadores. Durante esse breve período, antes de concluir a operação, os outros 99 computadores ainda poderão concluir as transações. Apenas as transações que envolveriam as partições que estão sendo relocadas são bloqueadas. Depois que o processo de failover ser concluído, o cache poderá continuar executando, totalmente operacional, a 99% de sua capacidade de rendimento original. Depois que o servidor com falha for substituído e retornado para a grade de dados, o cache voltará para 100% da capacidade de rendimento.

Transações da Grade de Dados Cruzada

Em termos de desempenho, disponibilidade e escalabilidade, as transações de grade de dados cruzada são o oposto de transações de partição única. As transações de grade de dados cruzada acessam cada partição e, portanto, cada computador na configuração. Cada computador na grade de dados é instruído a procurar alguns dados e retornar o resultado. A transação não pode ser concluída até que cada computador tenha respondido e, dessa forma, o rendimento da grade de dados inteira ficará limitado em função do computador mais lento. Incluir computadores não agiliza o computador mais lento e, assim, não melhora o rendimento do cache.

As transações de grade de dados cruzada possuem um efeito semelhante em termos de disponibilidade. Estendendo o exemplo anterior, se você estiver executando 100 servidores e um deles falhar, então, 100% das transações que estão em andamento no momento em que esse servidor falhou será recuperado. Depois que o servidor falhar, o WebSphere eXtreme Scale relocará as partições que são hospedadas por esse servidor nos outros 99 computadores. Durante esse tempo, antes de o processo de failover ser concluído, a grade de dados não pode processar nenhuma dessas transações. Depois que o processo de failover ser concluído, o cache poderá continuar executando, porém com capacidade reduzida. Se cada computador na grade de dados atender 10 partições, então 10 dos 99 computadores restantes receberão, pelo menos, uma partição extra como parte do processo de failover. Incluir uma partição extra aumenta a carga de trabalho desse computador em pelo menos 10%. Como o rendimento da grade de dados é limitado ao rendimento do computador mais lento em uma transação de grade de dados cruzada, em média, o rendimento é reduzido em 10%.

As transações de partição única são preferidas para as transações de grade de dados cruzada para efetuar scale out com um cache de objeto distribuído e altamente disponível, como o WebSphere eXtreme Scale. Aumentar o desempenho desses tipos de sistemas requer o uso de técnicas que são diferentes das metodologias relacionais tradicionais, porém é possível transformar as transações de grade de dados cruzada em transações de partição única escalável.

Boas práticas para criar modelos de dados escaláveis

As boas práticas para construir aplicativos escaláveis com produtos como o WebSphere eXtreme Scale incluem duas categorias: princípios básicos e dicas de implementação. Os princípios básicos são ideias principais que precisam ser capturadas no projeto dos próprios dados. Um aplicativo que não observa esses princípios podem não ser escalados tão bem, mesmo para as transações de linha principal. Por outro lado, as dicas de implementação são aplicadas em transações problemáticas em um aplicativo bem projetado que observa os princípios gerais para modelos de dados escaláveis.

Princípios Básicos

Algumas das maneiras importantes de otimizar a escalabilidade são conceitos ou princípios básicos que devem ser mantidos em mente.

Duplicar em vez de normalizar

> O que mais deve-se ter em mente sobre os produtos como o WebSphere eXtreme Scale é que eles são designados para propagar dados entre um grande número de computadores. Se o objetivo é concluir a maioria ou todas as transações em uma única partição, o design do modelo de dados

precisa garantir que todos os dados que a transação possa precisar estejam localizados na partição. Na maioria das vezes, a única maneira de fazer isso é duplicar os dados.

Por exemplo, considere um aplicativo, como um quadro de avisos. Duas transações muito importantes para um quadro de mensagens mostram todas as postagens de um determinado usuário e todas as postagens de um determinado tópico. Primeiro considere como essas transações trabalhariam com um modelo de dados normalizado que contenha um registro de usuário, um registro de tópico e um registro de postagem que contenha o texto real. Se as postagens forem particionadas com os registros do usuário, a exibição do tópico torna-se uma transação de grade cruzada e vice-versa. Os tópicos e os usuários não podem ser particionados juntos porque eles possuem um relacionamento muitos-para-muitos.

A melhor maneira de fazer com que esse quadro de avisos seja escalável é duplicar as postagens, armazenar uma cópia com o registro de tópico e uma cópia com o registro do usuário. Em seguida, a exibição das postagens de um usuário é uma transação de partição única, exibir as postagens em um tópico é uma transação de partição única e atualizar ou excluir uma postagem é uma transação de duas partições. Todas essas três transações serão escaladas de maneira linear já que o número de computadores na grade de dados aumenta.

Escalabilidade Em Vez de Recursos

O maior obstáculo a ser superado ao considerar os modelos de dados não-normalizados é o impacto que esse modelos causam nos recursos. Manter duas, três ou mais cópias de alguns dados pode parecer que muitos recursos usados são práticos. Ao se deparar com esse cenário, lembre-se dos seguintes fatos: os recursos de hardware se tornam mais baratos a cada dia. Segundo e o mais importante, o WebSphere eXtreme Scale elimina a maioria dos custos implícitos associados à implementação de mais recursos.

Os recursos devem ser medidos em termos de custo em vez de computador, como megabytes e processadores. Os armazenamentos de dados que trabalham com dados relacionais normalizados geralmente precisam estar localizados no mesmo computador. Essa colocação necessária significa que um único computador corporativo maior precisa ser adquirido em vez de vários computadores menores. Com o hardware corporativo, um computador que executa um milhão de transações por segundo normalmente é bem mais barato que 10 computadores capazes de executar 100 mil transações por segundo cada um.

Incluir recursos também gera custos de negócios. Um negócio em crescimento normalmente pode ficar sem capacidade. Quando não houver capacidade, é necessário encerrar para mudar para um computador maior e mais rápido ou é necessário criar um segundo ambiente de produção para o qual você possa mudar. De uma das formas, custos adicionais serão acarretados na forma de negócios perdidos ou ao manter quase o dobro da capacidade necessária durante o período de transação.

Com o WebSphere eXtreme Scale, o aplicativo não precisa ser encerrado para incluir capacidade. Se seus projetos de negócios requererem 10% de capacidade a mais para o próximo ano, aumente 10% o número de computadores na grade de dados. É possível aumentar essa porcentagem sem ocorrer tempo de inatividade do aplicativo e sem adquirir capacidade em excesso.

Evitar transformações de dados

Quando estiver usando o WebSphere eXtreme Scale, os dados deverão ser armazenados em um formato que possa ser consumado diretamente pela lógica de negócios. Dividir os dados em um formato mais primitivo gera custos. A transformação precisa ser feita quando os dados forem gravados e lidos. Com os bancos de dados relacionais, essa transformação é feita sem necessidade porque os dados são definitivamente persistidos no disco muito frequentemente, mas com o WebSphere eXtreme Scale, essas transformações não precisam ser executadas. Na maioria das vezes os dados são armazenados na memória e podem, portanto, serem armazenados no formato exato em que o aplicativo precisa.

Observar essa regra de amostra ajuda a desnormalizar os dados de acordo com o primeiro princípio. O tipo mais comum de transformação dos dados de negócios é as operações JOIN que são necessárias para tornar os dados normalizados em um conjunto de resultados que se ajusta às necessidades do aplicativo. Armazenar os dados no formato correto implicitamente evita a execução dessas operações JOIN e produz um modelo de dados não-normalizado.

Eliminar consultas ilimitadas

Independente de como os seus dados são estruturados, as consultas ilimitadas não são bem escaladas. Por exemplo, não tenha uma transação que solicite uma lista de todos os itens classificados por valor. Essa transação pode funcionar inicialmente quando o número total de itens for 1.000, mas quando o número total de itens chegar a 10 milhões, a transação retornará todos os 10 milhões de itens. Se você executar essa transação, os dois resultados mais prováveis são o tempo limite da transação se esgotar ou o cliente receber um erro de falta de memória.

A melhor opção é alterar a lógica de negócios para que apenas os 10 ou 20 itens principais possam ser retornados. Essa mudança de lógica mantém o tamanho da transação gerenciável, independente de quantos itens estão no cache.

Definir esquema

A principal vantagem de normalizar os dados é que o sistema de banco de dados controla a consistência de dados em segundo plano. Quando os dados são desnormalizados para escalabilidade, esse gerenciamento de consistência de dados automático já não existe mais. É necessário implementar um modelo de dados que funcione na camada de aplicativos ou como um plug-in para a grade de dados distribuída para garantir a consistência dos dados.

Considere o exemplo do quadro de mensagens. Se uma transação remover uma postagem de um tópico, a postagem duplicada no registro do usuário precisará ser removida. Sem um modelo de dados, um desenvolvedor pode gravar o código do aplicativo para remover a postagem do tópico e se esquecer de remover a postagem do registro do usuário. Entretanto, se o desenvolvedor estiver usando um modelo de dados em vez de interagir diretamente com o cache, o método removePost no modelo de dados poderá extrair o ID do usuário da postagem, procurar pelo registro do usuário e remover a postagem duplicada em segundo plano.

Como alternativa, é possível implementar um listener que é executado na partição real que detecta a alteração no tópico e ajusta automaticamente o registro do usuário. Um listener pode ser benéfico porque o ajuste do

registro do usuário pode ocorrer localmente caso a partição possua o registro do usuário ou, mesmo se o registro do usuário estiver em uma partição diferente, a transação ocorrerá entre os servidores em vez de ocorrer entre o cliente e o servidor. A conexão de rede entre os servidores provavelmente é mais rápida do que a conexão de rede entre o cliente e o servidor.

Evitar contenção

Evite cenários, como ter um contador global. A grade de dados não será escalável se um registro único estiver sendo usado por um número de vezes desproporcional, comparado com o restante dos registros. O desempenho da grade de dados será limitado pelo desempenho do computador que mantém o registro fornecido.

Nessas situações, tente dividir o registro para que ele seja gerenciado por partição. Por exemplo, considere uma transação que retorna o número total de entradas no cache distribuído. Em vez de fazer com que cada operação de inserção e remoção acesse um único registro que incrementa, faça com que o listener em cada partição controle as operações de inserção e remoção. Com o controle desse listener, a inserção e a remoção poderá se transformar nas operações de partição única.

A leitura do contador se transformará em uma operação de grade de dados cruzada, mas para a maior parte, isso já era ineficiente como uma operação de grade de dados cruzada porque o desempenho dependia do desempenho do computador que hospeda o registro.

Dicas de Implementação

Também é possível considerar as seguintes dicas para obter a melhor escalabilidade.

Índices de Procura Reversa

Considere um modelo de dados não-normalizado adequadamente em que os registros são particionados com base no número do ID do cliente. Esse método de particionamento é a opção lógica porque quase cada operação de negócios executada com o registro do cliente usa o número de ID do cliente. Entretanto, uma transação importante que não usa o número do ID do cliente é a transação de login. É mais comum ter nomes de usuário ou endereços de e-mail para login em vez de números do ID de cliente.

A abordagem simples com o cenário de login é usar uma transação de grade de dados cruzada para localizar o registro do cliente. Conforme explicado anteriormente, essa abordagem não é escalada.

A próxima opção pode ser uma partição no nome do usuário ou e-mail. Essa opção não é viável porque todas as operações baseadas no ID do cliente se transformam em transações de grade de dados cruzada. Além disso, os clientes do seu site podem alterar o nome do usuário ou o endereço de e-mail. Produtos como o WebSphere eXtreme Scale precisam do valor usado para particionar os dados que permanecerem constantes.

A solução correta é usar um índice de consulta reversa. Com o WebSphere eXtreme Scale, um cache pode ser criado na mesma grade distribuída que o cache que mantém todos os registros do usuário. Esse cache é altamente escalável, particionado e escalável. Esse cache pode ser usado para mapear um nome de usuário ou endereço de e-mail para um ID de cliente. Esse cache transforma o login em uma operação de duas partições em vez de

uma operação de grade cruzada. Esse cenário não é tão bom quanto uma transação de partição única, mas o rendimento ainda pode ser escalado linearmente conforme o número de computadores aumenta.

Computar no Momento da Gravação

Valores normalmente calculados, como médias ou totais, podem ser despendiosos para serem produzidos porque essas operações normalmente requerem leitura de um grande número de entradas. Como as leituras são mais comuns do que as gravações na maioria dos aplicativos, é eficiente calcular esses valores no momento da gravação e, em seguida, armazenar o resultado no cache. Essa prática torna as operações mais rápidas e mais escaláveis.

Campos Opcionais

Considere um registro de usuário que mantém um número de negócios, doméstico e de telefone. Um usuário pode ter todos, nenhum ou qualquer combinação desses números definida. Se os dados forem normalizados, uma tabela do usuário e um número de telefone existirão. Os números de telefone para um determinado usuário pode ser localizado usando uma operação JOIN entre as duas tabelas.

Desnormalizar esse registro não requer duplicação de dados, porque a maioria dos usuários não compartilha números de telefone. Em vez disso, slots vazios no registro do usuário devem ser permitidos. Em vez de ter uma tabela de número de telefone, inclua três atributos em cada registro do usuário, um para cada tipo de número de telefone. Essa inclusão de atributos elimina a operação JOIN e torna uma procura por número de telefone de um usuário uma operação de partição única.

Colocação de relacionamentos muitos-para-muitos

Considere um aplicativo que controla os produtos e as lojas nas quais os produtos são vendidos. Um único produto é vendido em muitas lojas e uma única loja vende muitos produtos. Suponha que esse aplicativo controla 50 grandes varejistas. Cada produto é vendido em um máximo de 50 lojas, com cada uma vendendo milhares de produtos.

Mantenha uma lista de lojas dentro da entidade do produto (organização A), em vez de manter uma lista de produtos dentro de cada entidade de loja (organização B). Observar algumas das transações que esse aplicativo teria que executar mostra o porquê a organização A é mais escalável.

Primeiro observe as atualizações. Com a organização A, remover um produto do inventário de uma loja bloqueia a entidade do produto. Se a grade de dados mantiver 10.000 produtos, apenas 1/10.000 da grade de dados precisará ser bloqueada para executar a atualização. Com a organização B, a grade de dados contém apenas 50 lojas, de modo que 1/50 da grade deverá ser bloqueada para concluir a atualização. Portanto, embora os dois possam ser considerados operações de partição única, a organização A é escalada mais eficientemente.

Agora, considerando as leituras na organização A, observar as lojas nas quais um produto é vendido é uma transação de partição única que é escalada e é rápido porque a transação transmite apenas uma pequena quantia de dados. Com a organização B, essa transação se torna uma transação de grade de dados cruzada porque cada entidade de loja deve ser acessada para ver se o produto é vendido nessa loja, que revela uma enorme vantagem de desempenho para a organização A.

Escalando com Dados Normalizados

Um uso legítimo de transações de grade de dados cruzada é escalar o processamento de dados. Se uma grade de dados tiver 5 computadores e uma transação de grade de dados cruzada for despachada, que classifica cerca de 100.000 registros em cada computador, essa transação classificará entre 500.000 registros. Se o computador mais lento na grade de dados puder executar 10 dessas transações por segundo, a grade de dados poderá classificar cerca de 5.000.000 de registros por segundo. Se os dados da grade dobrarem, cada computador deverá classificar cerca de 200.000 registros e cada transação classificará cerca de 1.000.000 de registros. Esse aumento de dados diminui o rendimento do computador mais lento para 5 transações por segundo, reduzindo, assim, o rendimento da grade de dados para 5 transações por segundo. Além disso, a grade de dados classifica entre 5.000.000 de registros por segundo.

Neste cenário, dobrar o número de computadores permite que cada computador volte para o carregamento anterior de classificação de 100.000 registros, permitindo que o computador mais lento processe 10 dessas transações por segundo. O rendimento da grade de dados permanece o mesmo nas 10 solicitações por segundo, mas agora cada transação processa 1.000.000 de registros dobrando, assim, a capacidade da grade em processar 10.000.000 de registros por segundo.

Aplicativos, como um mecanismo de procura, que precisam ser escalados tanto em termos de processamento de dados para acomodar o tamanho crescente da Internet e de rendimento para acomodar o crescimento de usuários, requer a criação de diversas grades de dados, com um round robin das solicitações entre as grades. Se você precisar efetuar scale up do rendimento, inclua computadores e outra grade de dados para solicitações de serviço. Se for necessário efetuar scale up do processamento de dados, inclua mais computadores e mantenha o número de grades de dados constante.

Visão Geral de Segurança

WebSphere eXtreme Scale pode proteger o acesso a dados, incluindo permissão para integração com provedores de segurança externos.

Nota: Em um armazenamento de dados fora do cache existente, como um banco de dados, provavelmente é necessário ter recursos de segurança integrados que podem não ser necessários para configuração ou ativação de modo ativo. Entretanto, após ter armazenado seus dados em cache com o eXtreme Scale, você deve considerar a importante situação resultante de que seus recursos de segurança de backend não estão mais em vigor. É possível configurar a segurança do eXtreme Scale no níveis necessários para que a nova arquitetura armazenada em cache para os seus dados também fique segura.
A seguir é apresentado um breve resumo sobre os recursos de segurança do eXtreme Scale. Para obter mais informações detalhadas sobre como configurar a segurança, consulte o *Guia de Administração* e o *Guia de Programação*.

Fundamentos sobre Segurança Distribuída

A segurança distribuída do eXtreme Scale é baseada em três conceitos fundamentais:

Autenticação confiável
A habilidade de determinar a identidade do solicitante. O WebSphere eXtreme Scale suporta autenticação cliente-para-servidor e servidor-para-servidor.

Autorização
A habilidade de dar permissões para conceder direitos de acesso ao solicitante. O WebSphere eXtreme Scale suporta diferentes autorizações para várias operações.

Transporte Seguro
A transmissão segura dos dados sobre uma rede. O WebSphere eXtreme Scale suporta os protocolos TLS/SSL (Transport Layer Security/Secure Sockets Layer).

Autenticação

O WebSphere eXtreme Scale suporta uma estrutura de cliente e servidor distribuída. Uma infraestrutura de segurança de cliente e servidor está estabelecida para proteger o acesso aos servidores eXtreme Scale. Por exemplo, quando a autenticação é necessária pelo servidor do eXtreme Scale, um cliente do eXtreme Scale deve fornecer credenciais para se autenticar no servidor. Essas credenciais podem ser um par de nome de usuário e senha, um certificado cliente, um ticket Kerberos ou dados que são apresentados em um formato acordado entre o cliente e o servidor.

Autorização

As autorizações do WebSphere eXtreme Scale são baseadas em assuntos e permissões. É possível utilizar o Java Authentication and Authorization Services (JAAS) para autorizar o acesso ou é possível conectar uma abordagem customizada, tal como Tivoli Access Manager (TAM), para tratar as autorizações. As seguintes autorizações podem ser fornecidas a um cliente ou grupo:

Autorização de mapa
Execute operações insert, read, update, evict ou delete nos Mapas.

Autorização do ObjectGrid
Execute consultas em objetos ou entidades e consultas em fluxos nos objetos do ObjectGrid.

Autorização do agente do DataGrid
Permita que os agentes do DataGrid sejam implementados em um ObjectGrid.

Autorização do mapa do lado do servidor
Replique um mapa de servidor para o lado do cliente ou crie um índice dinâmico para o mapa do servidor.

Autorização de administração
Execute tarefas de administração.

Segurança do Transporte

Para garantir a segurança da comunicação entre cliente e o servidor, o WebSphere eXtreme Scale suporta TLS/SSL. Estes protocolos fornecem segurança da camada de transporte com autenticidade, integridade e confidencialidade para uma conexão segura entre um cliente e um servidor do eXtreme Scale.

Segurança da Grade

Em um ambiente seguro, um servidor deve poder verificar a autenticidade de outro servidor. O WebSphere eXtreme Scale utiliza um mecanismo de cadeia de chave secreta compartilhado para este propósito. Este mecanismo de chave secreta é semelhante a uma senha compartilhada. Todos os servidores eXtreme Scale aceitam uma cadeia secreta compartilhada. Quando um servidor se junta à grade de dados, o servidor é desafiado a apresentar a sequência secreta. Se a cadeia secreta do servidor que está se juntando corresponder a uma cadeia no servidor principal, então o servidor que está se juntando pode ser unir á grade. Caso contrário, o pedido de junção será rejeitado.

Não é seguro enviar um segredo em texto não-criptografado. A infraestrutura de segurança do eXtreme Scale fornece um plug-in SecureTokenManager para possibilitar que o servidor faça a segurança deste segredo antes de enviá-lo. É possível escolher como implementar a operação segura. O WebSphere eXtreme Scale fornece uma implementação, na qual a operação segura é implementada para criptografar e assinar o segredo.

Segurança Java Management Extensions (JMX) em uma Topologia de Implementação Dinâmica

A segurança JMX MBean é suportada em todas as versões do eXtreme Scale. Clientes dos MBeans do servidor de catálogos e MBeans do servidor de contêineres podem ser autenticados e o acesso às operações do MBean podem ser impostos.

Segurança Local do eXtreme Scale

A segurança local do eXtreme Scale é diferente do modelo distribuído do eXtreme Scale porque o aplicativo instancia diretamente e utiliza uma instância do ObjectGrid. Seu aplicativo e as instâncias do eXtreme Scale estão na mesma Java virtual machine (JVM). Como não há nenhum conceito de cliente/servidor neste modelo, a autenticação não é suportada. Seu aplicativo deve gerenciar sua própria autenticação e, então, passar o objeto Subject autenticado para o eXtreme Scale. Porém, o mecanismo de autorização usado para o modelo de programação do eXtreme Scale local é o mesmo que o usado para o modelo cliente/servidor.

Configuração e Programação

Para obter mais informações sobre como configurar e programar a segurança, consulte o Integração de Segurança com Provedores Externos e o API de Segurança.

Visão Geral do Serviço de Dados REST

O serviço de dados REST WebSphere eXtreme Scale é um serviço HTTP Java compatível com Microsoft WCF Data Services (formalmente, ADO.NET Data Services) e implementa o Open Data Protocol (OData). O Microsoft WCF Data Services é compatível com essa especificação quando utiliza Visual Studio 2008 SP1 e .NET Framework 3.5 SP1.

Requisitos de Compatibilidade

O serviço de dados REST permite que qualquer cliente HTTP acesse uma grade de dados. O serviço de dados REST é compatível com o suporte do WCF Data Services fornecido com o Microsoft .NET Framework 3.5 SP1. Aplicativos RESTful

podem ser desenvolvidos com um rico conjunto de ferramentas fornecido pelo Microsoft Visual Studio 2008 SP1. A figura fornece uma visão geral de como o WCF Data Services interage com clientes e bancos de dados.

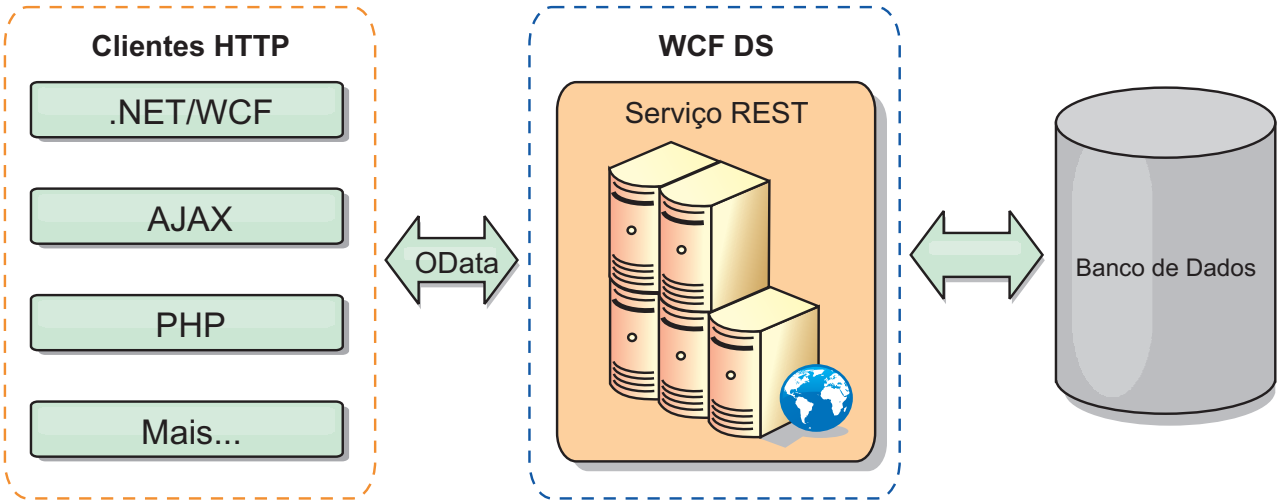


Figura 36. Microsoft WCF Data Services

O WebSphere eXtreme Scale inclui um conjunto de APIs com várias funções para clientes Java. Conforme mostrado na figura a seguir, o serviço de dados REST é um gateway entre clientes HTTP e a grade de dados do WebSphere eXtreme Scale, comunicando-se com a grade por meio de um cliente do WebSphere eXtreme Scale. O serviço de dados REST é um servlet Java, que permite implementações flexíveis para Plataforma Java comum, plataformas Enterprise Edition (JEE), como o WebSphere Application Server. O serviço de dados REST se comunica com a grade de dados WebSphere eXtreme Scale usando as APIs Java WebSphere eXtreme Scale. Ele permite clientes do WCF Data Services ou qualquer outro cliente que possa se comunicar com HTTP e XML.

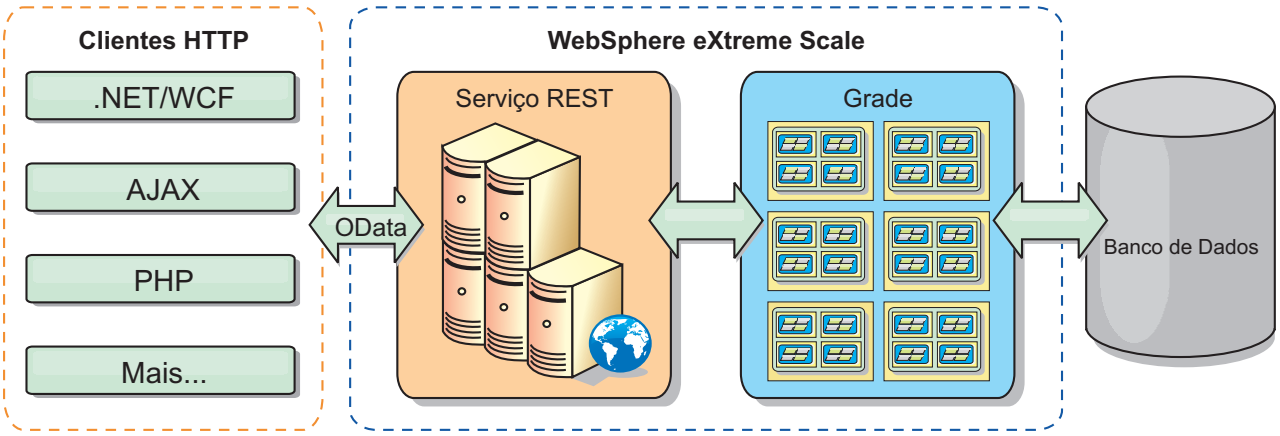


Figura 37. Serviço de Dados REST do WebSphere eXtreme Scale

Consulte o Configurando Serviços de Dados REST ou use os seguintes links para aprender mais sobre o WCF Data Services.

- Microsoft WCF Data Services Developer Center
- Visão geral do ADO.NET Data Services no MSDN
- White Paper: Utilizando 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
- FAQ do Open Data Protocol

Características

Esta versão do serviço de dados REST do eXtreme Scale suporta os seguintes recursos:

- Modelagem automática de entidades de API EntityManager do eXtreme Scale como entidades do WCF Data Services, que inclui o seguinte suporte:
  - Tipo de dados Java para conversão de tipo do Entity Data Model
  - Suporte de associação de entidade
  - Suporte à raiz do esquema e à associação de chave, que é requerido para grades de dados particionadas

  Consulte o Modelo de Entidade para obter informações adicionais.
- Formato de carga útil de dados Atom Publish Protocol (AtomPub ou APP) XML e JavaScript Object Notation (JSON).
- Operações Create, Read, Update and Delete (CRUD) utilizando os respectivos métodos de pedido de HTTP: POST, GET, PUT e DELETE. Além disso, a extensão da Microsoft MERGE é suportada.
- Consultas simples, usando filtros
- Pedidos de recuperação de lote e do conjunto de mudanças
- Suporte de grade de dados particionada para alta disponibilidade
- Interoperabilidade com clientes de API EntityManager do eXtreme Scale
- Suporte para servidores da Web JEE padrão
- Simultaneidade otimista
- Autorização e autenticação de usuário entre o serviço de dados REST e a grade de dados do eXtreme Scale

Limitações e Problemas Conhecidos

- Pedidos em túnel não são suportados.

Capítulo 2. Planejamento

Antes de instalar o WebSphere eXtreme Scale e de implementar seus aplicativos de grade de dados, você deve decidir sobre sua topologia de armazenamento em cache, concluir o planejamento da capacidade, revisar os requisitos de hardware e de software, as configurações de rede e ajustes, e assim por diante. Também é possível usar a lista de verificação operacional para garantir que seu ambiente esteja pronto para ter um aplicativo implementado.

Para uma discussão das boas práticas que é possível usar ao projetar seus aplicativos WebSphere eXtreme Scale, leia o seguinte artigo em developerWorks: Princípios e boas práticas para construir aplicativos WebSphere eXtreme Scale de alta execução e alta resiliência.

Planejando a Topologia

Com WebSphere eXtreme Scale, sua arquitetura pode utilizar armazenamento em cache de dados em memória local ou armazenamento em cache de dados de cliente/servidor distribuídos. A arquitetura pode ter relacionamentos variados com seus bancos de dados. Também é possível configurar a topologia para estender diversos datacenters.

O WebSphere eXtreme Scale requer infraestrutura adicional mínima para operar. A infraestrutura consiste em scripts para instalar, iniciar e parar um aplicativo Java Platform, Enterprise Edition em um servidor. Os dados em cache são armazenados nos servidores de contêiner e os clientes se conectam remotamente com o servidor.

Ambientes em Memória

Quando implementar em um local, um ambiente em memória, o WebSphere eXtreme Scale é executado em uma única Java Virtual Machine e não é replicado. Para configurar um ambiente local, é possível usar um arquivo XML do ObjectGrid ou as APIs do ObjectGrid.

Ambientes Distribuídos

Quando você implementa em um ambiente distribuído, o WebSphere eXtreme Scale é executado em um conjunto de Java Virtual Machines, aumentando o desempenho, a disponibilidade e a escalabilidade. Com essa configuração, poderá utilizar a replicação de dados e criação de partições. Também é possível incluir servidores adicionais sem restaurar seus servidores eXtreme Scale existentes. Assim como ocorre com um ambiente local, um arquivo XML do ObjectGrid, ou uma configuração programática equivalente, é necessário em um ambiente distribuído. Você também deve fornecer um arquivo XML de política de implementação com detalhes de configuração.

É possível criar implementações simples ou grandes a nível de terabytes, em que milhares de servidores são necessários.

Cache de Memória Local

Em um caso mais simples, o WebSphere eXtreme Scale pode ser utilizado como um cache de grade de dados em memória local (não distribuído). O caso local

pode beneficiar especialmente aplicativos de alta simultaneidade nos quais vários encadeamentos precisam acessar e modificar dados transientes. Os dados mantidos em uma grade de dados local podem ser indexados e recuperados utilizando consultas. As consultas ajudam você a trabalhar com grandes em conjuntos de dados de memória. O suporte fornecido com o Java Virtual Machine (JVM), embora esteja pronto para uso, tem uma estrutura de dados limitada.

A topologia de cache em memória local para WebSphere eXtreme Scale é usado para oferecer acesso transacional e consistente aos dados temporários dentro de uma única Java virtual machine.

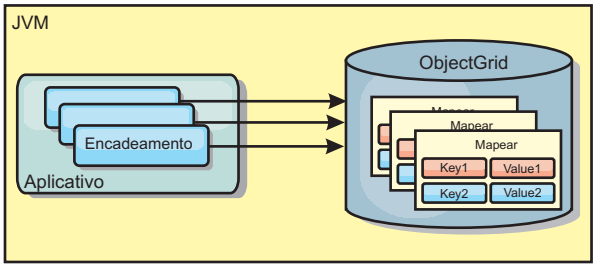


Figura 38. Cenário de Cache em Memória Local

Vantagens

- Configuração simples: Um ObjectGrid pode ser criado programaticamente ou declarativamente com o arquivo XML do descritor de implementação ObjectGrid ou com outras estruturas como Spring.
- Rápido: Cada BackingMap pode ser ajustado de maneira independente para utilização de memória e simultaneidade ideais.
- Ideal para topologias de uma única Java virtual machine com pequenos conjuntos de dados ou para armazenamento em cache de dados frequentemente acessados.
- Transacional. As atualizações BackingMap podem ser agrupadas em uma única unidade de trabalho e podem ser integradas como um último participante nas transações de duas fases como transações JTA (Java Transaction Architecture).

Desvantagens

- Não tolerante a falhas.
- Os dados não são replicados. Caches em memória são melhores para dados de referência somente para leitura.
- Não escalável. A quantidade de memória necessária pelo banco de dados pode ultrapassar a capacidade da Java virtual machine.
- Ocorrem problemas na inclusão de Java virtual machines:
  - Os dados não podem ser facilmente particionados
  - Você deve replicar manualmente o estado entre as Java virtual machines ou cada instância do cache poderá ter diferentes versões dos mesmos dados.
  - A invalidação é custosa.
  - Cada cache deve ser aquecido de maneira independente. O aquecimento é o período de carregamento de um conjunto de dados para que o cache seja preenchido com dados válidos.

Quando Utilizar

A topologia de implementação de cache em memória local deve ser usada somente quando a quantidade de dados a serem armazenados em cache for pequena (puder ser colocada em uma única Java virtual machine) e for relativamente estável. Dados antigos devem ser tolerados com esta abordagem. A utilização de evictors para manter os dados mais frequentemente ou recentemente usados no cache pode ajudar a diminuir o tamanho do cache e a aumentar a relevância dos dados.

Cache Local Replicado pelo Peer

Você deverá assegurar-se de que o cache esteja sincronizado se vários processos com instâncias de cache independentes existirem. Para assegurar-se de que as instâncias de cache estejam sincronizadas, ative um cache replicado por peer com o Java Message Service (JMS).

WebSphere eXtreme Scale inclui dois plug-ins que propagam automaticamente mudanças de transação entre instâncias do ObjectGrid peer. O plug-in JMSObjectGridEventListener propaga automaticamente as mudanças do eXtreme Scale usando JMS.

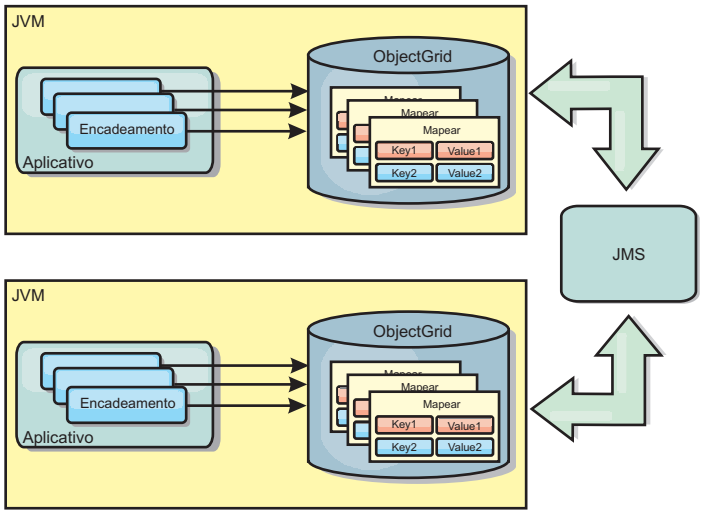


Figura 39. Cache Replicado pelo Peer com Alterações que são Propagadas com JMS

Se você estiver executando um ambiente doWebSphere Application Server, o plug-in TranPropListener também estará disponível. O plug-in TranPropListener usa o gerenciador de alta disponibilidade (HA) para propagar as mudanças em cada instância do cache de peer.

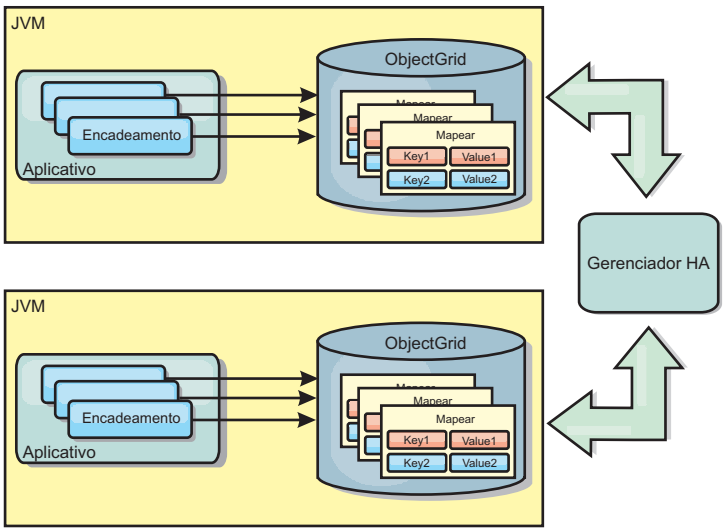


Figura 40. Cache Replicado pelo Peer com Alterações que são Propagadas com o Gerenciador de Alta Disponibilidade

Vantagens

- Os dados são mais válidos porque os dados são atualizados mais frequentemente.
- Com o plug-in TranPropListener, como no ambiente local, o eXtreme Scale pode ser criado programaticamente ou declarativamente com o arquivo XML descritor de implementação do eXtreme Scale ou com outras estruturas como Spring. A integração com o gerenciador de alta disponibilidade é feita automaticamente.
- Cada BackingMap pode ser independentemente ajustado para melhor utilização da memória e concorrência.
- As atualizações BackingMap podem ser agrupadas em uma única unidade de trabalho e podem ser integradas como um último participante nas transações de duas fases como transações JTA (Java Transaction Architecture).
- Ideal para poucas topologias JVM com um conjunto de dados razoavelmente pequeno ou para armazenamento em cache de dados frequentemente acessados.
- As atualizações em cada eXtreme Scale são replicadas para todas as instâncias do eXtreme Scale do peer. As alterações são consistentes desde que uma assinatura durável seja utilizada.

Desvantagens

- A configuração e a manutenção para o JMSObjectGridEventListener podem ser complexas. O eXtreme Scale pode ser criado programaticamente ou declarativamente com o arquivo XML descritor de implementação do eXtreme Scale ou com outras estruturas tais como Spring.
- Não escalável: A quantidade de memória necessária para que o banco de dados possa dominar a JVM.
- Funciona inadequadamente ao incluir Java Virtual Machines:
  - Os dados não podem ser facilmente particionados
  - A invalidação é custosa.
  - Cada cache deve ser aquecido de maneira independente

Quando Utilizar

Use a topologia de implementação apenas quando a quantia de dados a ser armazenada em cache for pequena, podendo ajustar-se a uma única JVM e se for relativamente estável.

Cache Integrado

As grades do WebSphere eXtreme Scale podem ser executadas nos processos existentes como servidores eXtreme Scale integrados ou podem ser gerenciadas como processos externos.

As grades integradas são úteis quando você está executando em um servidor de aplicativos, como o WebSphere Application Server. É possível iniciar servidores eXtreme Scale que não são integrados usando scripts da linha de comandos e executar em um processo Java.

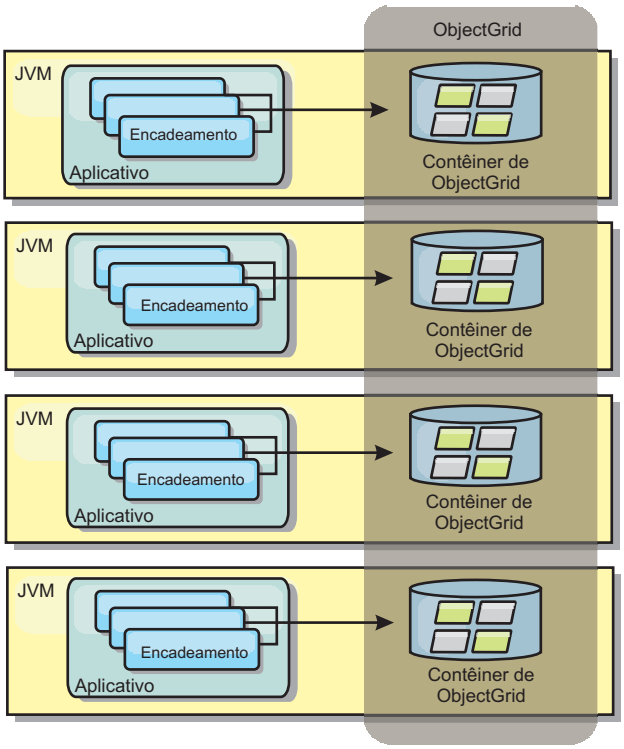


Figura 41. Cache Integrado

Vantagens

- Administração simplificada já que há menos processos para gerenciar.
- Implementação do aplicativo simplificada, já que a grade usa o carregador de classe do aplicativo cliente.
- Suporta particionamento e alta disponibilidade.

Desvantagens

- Aumento da área de cobertura da memória no processo do cliente já que todos os dados são colocados no processo.
- Aumento da utilização da CPU para atender pedidos de clientes.

- Mais difícil para manipular atualizações de aplicativo, pois os clientes estão usando os mesmos arquivos archive de Java do aplicativo que os servidores.
- Menos flexível. A escala dos clientes e servidores de grade não pode aumentar na mesma proporção. Quando os servidores são externamente definidos, é possível ter mais flexibilidade no gerenciamento do número de processos.

Quando Utilizar

Utilize grades integradas quando há grande quantidade de memória livre no processo do cliente para dados da grade e dados de failover potenciais.

Para obter mais informações, consulte o tópico sobre como ativar o mecanismo de invalidação do cliente no *Guia de Administração*.

Cache Distribuído

O WebSphere eXtreme Scale é mais frequentemente usado como um cache compartilhado, para fornecer acesso transacional a dados para múltiplos componentes onde, caso contrário, um banco de dados tradicional seria usado. O cache compartilhado elimina a necessidade de configurar um banco de dados.

Coerência do Cache

O cache é coerente porque todos os clientes veem os mesmos dados no cache. Cada pedaço de dado é armazenado em exatamente um servidor no cache, evitando cópias de registros desperdiçadas que poderiam potencialmente conter diferentes versões dos dados. Um cache coerente também pode conter mais dados, à medida que mais servidores são incluídos na grade de dados, e escalado linearmente à medida que a grade cresce em tamanho. Como os clientes acessam dados a partir desta grade de dados com chamadas de processo remotas, ela também é conhecida como um cache remoto, ou cache distante. Através do particionamento de dados, cada processo contém um subconjunto exclusivo do conjunto de dados total. As grades de dados maiores podem conter mais dados e atender mais solicitações para esses dados. A coerência também elimina a necessidade de enviar dados de invalidação ao redor da grade de dados porque não há dados antigos. O cache coerente retém somente a cópia mais recente de cada pedaço de dados.

Se você estiver executando um ambiente doWebSphere Application Server, o plug-in TranPropListener também estará disponível. O plug-in TranPropListener usa o componente de alta disponibilidade (Gerenciador HA) do WebSphere Application Server para propagar as alterações para cada instância de cache ObjectGrid de peer.

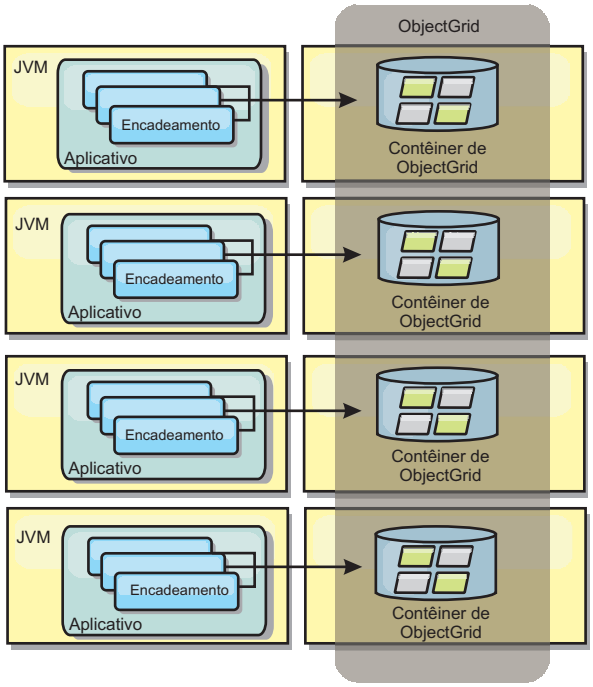


Figura 42. Cache Distribuído

Cache Local

Opcionalmente, os clientes têm um cache local, sequencial quando o eXtreme Scale é usado em uma topologia distribuída. Este cache opcional é chamado de cache local, um ObjectGrid independente em cada cliente, servindo como um cache para o cache remoto, do lado do cliente. O cache local é ativado por padrão quando o bloqueio é configurado como otimista ou nenhum e não pode ser utilizado quando é configurado como pessimista.

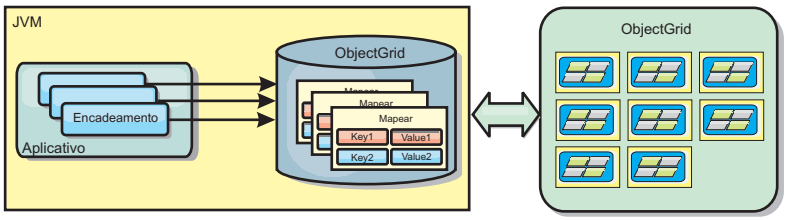


Figura 43. Cache Local

Um cache local é muito rápido porque fornece acesso em memória a um subconjunto do conjunto inteiro de dados em cache que é armazenado remotamente nos servidores do eXtreme Scale. O cache local não é particionado e contém dados de qualquer uma das partições eXtreme Scale remotas. O WebSphere eXtreme Scale pode ter até três camadas de cache, como a seguir.

1. O cache da camada da transação contém todas as alterações para uma única transação. O cache da transação contém uma cópia de trabalho dos dados até que a transação seja confirmada. Quando uma transação do cliente solicita dados de um ObjectMap, a transação é verificada primeiro

2. O cache local na camada do cliente contém um subconjunto de dados da camada do servidor. Quando a camada da transação não possui os dados, eles são buscados em uma camada do cliente, se disponíveis, e inseridos no cache da transação.
3. A grade de dados na camada do servidor contém a maioria dos dados e é compartilhada entre todos os clientes. A camada do servidor pode ser particionada, o que permite que uma grande quantidade de dados seja armazenada em cache. Quando o cache local do cliente não possui os dados, eles são buscados na camada do servidor e inseridos no cache cliente. A camada do servidor também pode ter um plug-in do Utilitário de Carga. Quando a grade não tem os dados necessários, o Utilitário de Carga é chamado e os dados resultantes são inseridos do armazém de dados de backend para a grade.

Para desativar o cache local, configure o atributo numberOfBuckets como 0 no arquivo descritor do ObjectGrid de substituição do cliente. Consulte o tópico sobre bloqueio de entrada de mapa para obter detalhes sobre estratégias de bloqueio do eXtreme Scale. O cache local também pode ser configurado para ter uma política de despejo configurada e diferentes plug-ins usando uma configuração do descritor do eXtreme Scale de substituição.

Vantagem

- Tempo de resposta rápido porque todos os acessos aos dados é local. Procurar pelos dados no cache próximo primeiro economiza acesso à grade de servidores, o que torna até mesmo os dados remotos acessíveis localmente.

Desvantagens

- Aumenta a duração dos dados antigos porque o cache próximo em cada camada pode estar fora de sincronização com os dados atuais na grade de dados.
- Depende de um evictor para invalidar dados a fim de evitar a falta de memória.

Quando Utilizar

Utilize quando o tempo de resposta for importante e dados antigos puderem ser tolerados.

Integração com o Banco de Dados: Armazenamento em Cache Write-behind, Sequencial e Lateral

O WebSphere eXtreme Scale é usado para colocar um banco de dados tradicional na frente e eliminar a atividade de leitura que normalmente é armazenada no banco de dados. Um cache coerente pode ser utilizado com um aplicativo direta ou indiretamente, utilizando um mapeador relacional de objeto. O cache coerente pode transferir o banco de dados ou o backend a partir das leituras. Em um cenário levemente mais complexo, tal como o acesso transacional a um conjunto de dados no qual apenas parte dos dados requer garantias de persistência tradicional, a filtragem pode ser utilizada para transferir até mesmo transações de gravação.

É possível configurar o WebSphere eXtreme Scale para funcionar como um espaço de processamento de banco de dados em memória altamente flexível. Entretanto, o WebSphere eXtreme Scale não é um object relational mapper (ORM). Ele não reconhece de onde vieram os dados na grade de dados. Um aplicativo ou um ORM pode colocar dados em um servidor eXtreme Scale. É responsabilidade da origem dos dados certificar-se de que eles permaneçam consistentes com o banco de dados no qual os dados se originaram. Isto significa que o eXtreme Scale não

pode invalidar dados que são extraídos de um banco de dados automaticamente. O aplicativo ou mapeador deve fornecer esta função e gerenciar os dados armazenados no eXtreme Scale.

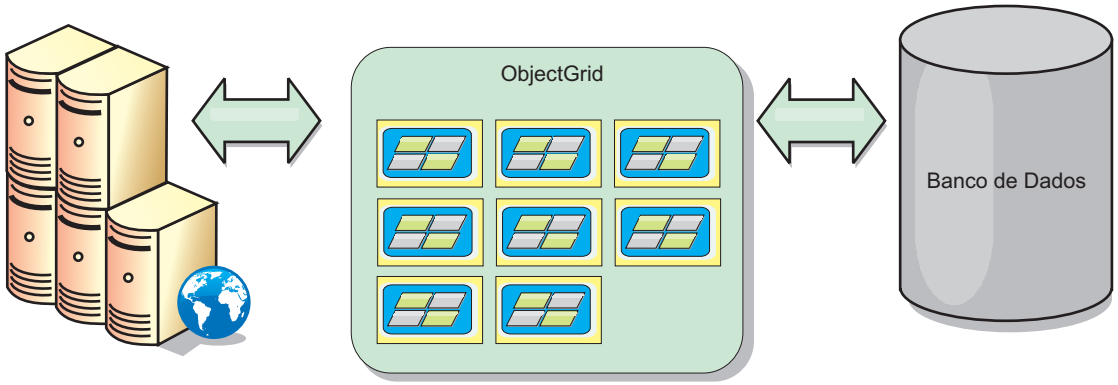


Figura 44. ObjectGrid como um Buffer de Banco de Dados

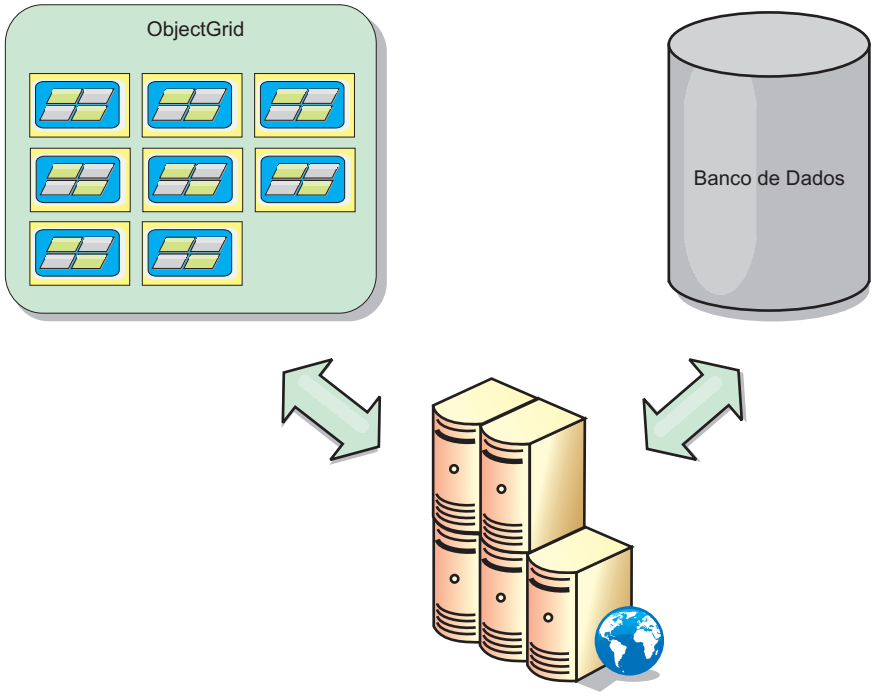


Figura 45. ObjectGrid como um Cache Secundário

Cache Disperso e Completo

O WebSphere eXtreme Scale pode ser utilizado como um cache disperso ou um cache completo. Um cache disperso mantém apenas um subconjunto do total de dados, enquanto que um cache completo mantém todos os dados. Ele também pode ser preenchido gradualmente, conforme os dados são necessários. Os caches dispersos normalmente são acessados usando chaves (ao invés de índices ou consultas) porque os dados estão disponíveis apenas parcialmente.

Cache Disperso

Quando uma chave não está presente em um cache disperso, ou os dados não estão disponíveis e uma falta de cache ocorre, a próxima camada é chamada. Os

dados são buscados, a partir de um banco de dados, por exemplo, e inseridos na camada de cache da grade de dados. Se estiver usando uma consulta ou um índice, apenas os valores atualmente carregados serão acessados e as solicitações não serão encaminhadas para as outras camadas.

Cache Completo

Um cache completo contém todos os dados necessários e pode ser acessado usando atributos não-chaves com índices ou consultas. Um cache completo é pré-carregado com dados a partir do banco de dados antes que o aplicativo tente acessar os dados. Um cache completo pode funcionar como uma substituição do banco de dados após os dados serem carregados. Como todos os dados estão disponíveis, as consultas e índices podem ser usados para localizar e agregar dados.

Cache Secundário

Quando o WebSphere eXtreme Scale é usado como um cache secundário, o backend é usado com a grade de dados.

Cache Secundário

É possível configurar o produto como um cache secundário para a camada de acesso a dados de um aplicativo. Neste cenário, o WebSphere eXtreme Scale é utilizado para armazenar temporariamente objetos que normalmente poderiam ser recuperados de um banco de dados de backend. Aplicativos verificam se a grade de dados contém os dados. Se os dados estiverem na grade de dados, eles serão retornados para o responsável pela chamada. Se os dados não existirem, eles serão recuperados a partir do banco de dados de backend. Os dados são então inseridos na grade de dados para que a próxima solicitação possa usar a cópia em cache. O diagrama a seguir ilustra como o WebSphere eXtreme Scale pode ser usado como um cache secundário com uma camada de acesso a dados arbitrários, como OpenJPA ou Hibernate.

Plug-ins do cache secundário para Hibernate e OpenJPA

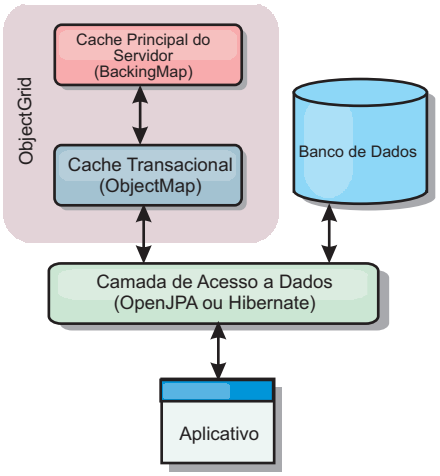


Figura 46. Cache Secundário

Os plug-ins de cache para ambos OpenJPA e Hibernate são incluídos no WebSphere eXtreme Scale, o que permite usar o produto como um cache secundário automático. Usar o WebSphere eXtreme Scale como um provedor de cache aumenta o desempenho ao ler e enfileirar dados e reduz a carga para o

banco de dados. Existem vantagens que o WebSphere eXtreme Scale tem sobre as implementações de cache integrado porque o cache é automaticamente replicado entre todos os processos. Quando um cliente armazena em cache um valor, todos os outros clientes podem usar o valor em cache.

Cache Sequencial

É possível configurar em cache sequencial para um backend de banco de dados ou como um cache secundário para um banco de dados. O armazenamento em cache sequencial utiliza o eXtreme Scale como o meio principal de interação com os dados. Quando o eXtreme Scale é usado como um cache sequencial, o aplicativo interage com o backend usando um plug-in Loader.

Cache Sequencial

Quando usado como um cache sequencial, o WebSphere eXtreme Scale interage com o backend usando um plug-in Loader. Este cenário pode simplificar o acesso a dados porque os aplicativos podem acessar as APIs do eXtreme Scale diretamente. Vários cenários de armazenamento em cache diferentes são suportados no eXtreme Scale para garantir que os dados no cache e os dados no backend sejam sincronizados. O diagrama a seguir ilustra como um cache sequencial interage com o aplicativo e o back end.

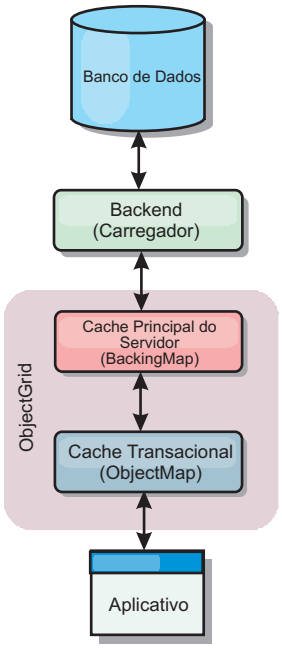


Figura 47. Cache Sequencial

A opção de armazenamento em cache em linha simplifica o acesso aos dados pois ela permite que os aplicativos acessem diretamente as APIs do eXtreme Scale. O WebSphere eXtreme Scale suporta diversos cenários de armazenamento em cache em linha, como os seguintes:

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

Cenário de Armazenamento em Cache Read-through

Um cache read-through é um cache disperso que lentamente carrega entradas de dados por chave à medida que elas são solicitadas. Isto é feito sem exigir que o responsável pela chamada saiba quais entradas estão preenchidas. Se os dados não puderem ser localizados no cache do eXtreme Scale, o eXtreme Scale irá recuperar os dados ausentes do plug-in do utilitário de carga, que carrega os dados do banco de dados backend e insere os dados no cache. Pedidos subsequentes para a mesma chave de dados serão localizados no cache até que ele possa ser removido, invalidado ou despejado.

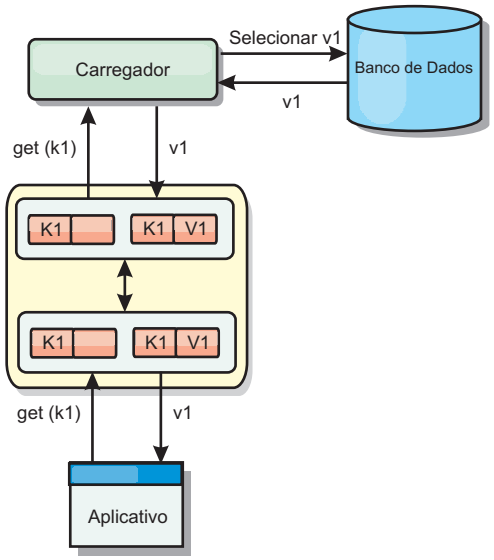


Figura 48. Armazenamento em Cache Read-through

Cenário de Armazenamento em Cache Write-through

Em um cache write-through, cada gravação no cache é gravada de maneira síncrona no banco de dados utilizando o Utilitário de Carga. Este método fornece consistência com o backend, mas diminui o desempenho de gravação pois a operação do banco de dados é síncrona. Como o cache e o banco de dados são ambos atualizados, as leituras subsequentes para os mesmos dados serão localizadas no cache, evitando a chamada do banco de dados. Um cache write-through sempre é utilizado em conjunto com um cache read-through.

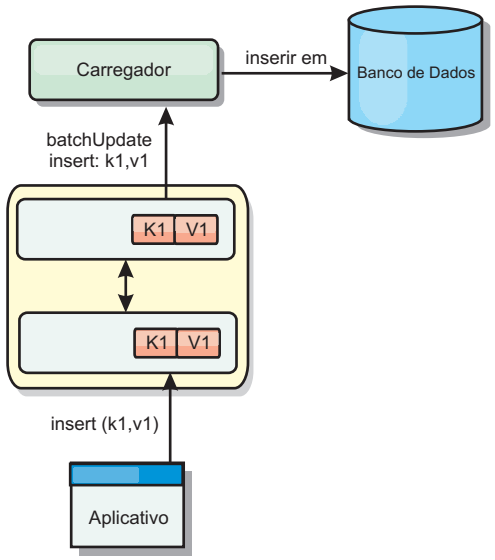


Figura 49. Armazenamento em Cache Write-through

Cenário de Armazenamento em Cache Write-behind

A sincronização do banco de dados pode ser aprimorada pela gravação de alterações de maneira assíncrona. Isto é conhecido como um cache write-behind ou write-back. Alterações que normalmente poderiam ser gravadas de maneira síncrona no utilitário de carga são, ao invés disso, armazenadas em buffer no eXtreme Scale e gravadas no banco de dados utilizando um encadeamento secundário. O desempenho de gravação é significativamente aumentado pois a operação do banco de dados é removida da transação do cliente e as gravações do banco de dados podem ser compactadas.

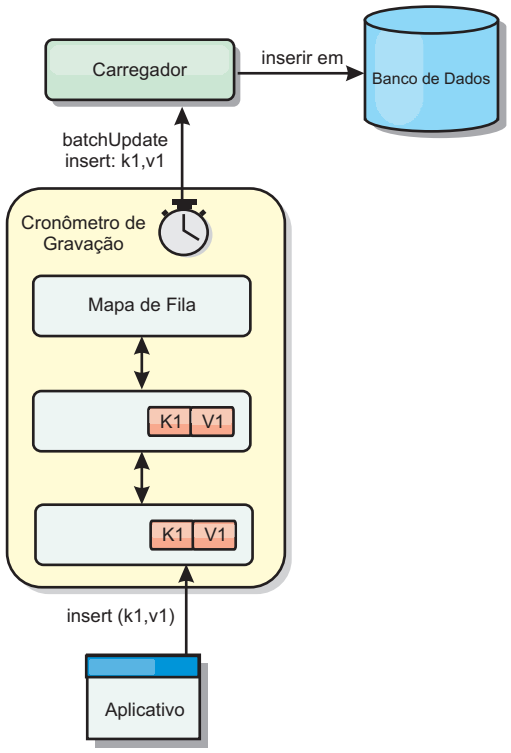


Figura 50. Armazenamento em Cache Write-behind

Armazenamento em Cache Write-behind

É possível utilizar armazenamento em cache write-behind para reduzir o gasto adicional que ocorre durante a atualização de um banco de dados que você está utilizando como back end.

Visão Geral do Armazenamento em Cache Write-Behind

O armazenamento em cache write-behind enfileira assincronamente as atualizações no plug-in do Utilitário de Carga. É possível melhorar o desempenho desconectando atualizações, inserções e remoções para um mapa, a sobrecarga de atualização do banco de dados de backend. A atualização assíncrona é executada após um atraso baseado em tempo (por exemplo, cinco minutos) ou um atraso baseado em entradas (1000 entradas).

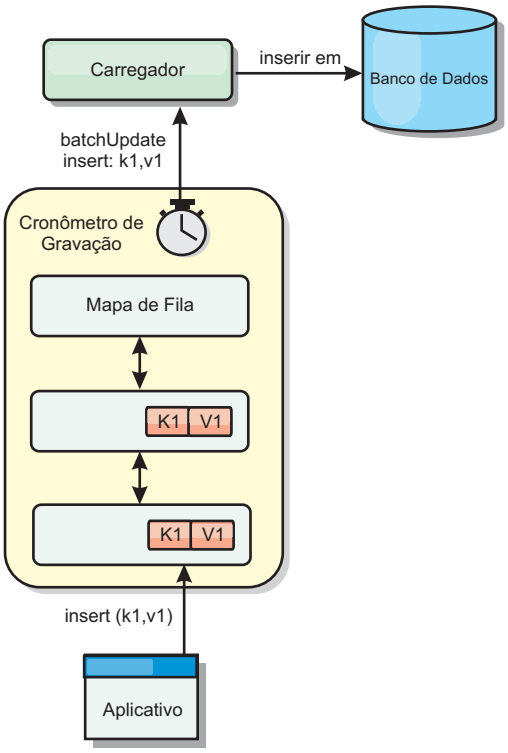


Figura 51. Armazenamento em Cache Write-behind

A configuração write-behind em um BackingMap cria um encadeamento entre o utilitário de carga e o mapa. O utilitário de carga então delega pedidos de dados através do encadeamento de acordo com as definições da configuração no método BackingMap.setWriteBehind. Quando uma transação do eXtreme Scale insere, atualiza ou remove uma entrada de um mapa, um objeto LogElement é criado para cada um destes registros. Estes elementos são enviados para o utilitário de carga write-behind e enfileirados em um ObjectMap especial denominado mapa de fila. Cada mapa de apoio com a configuração write-behind ativada possui seus próprios mapas de fila. Um encadeamento write-behind remove periodicamente os dados enfileirados dos mapas de fila e executa o push deles para o utilitário de carga de backend real.

O utilitário de carga write-behind enviará apenas os tipos insert, update e delete dos objetos LogElement para o utilitário de carga real. Todos os outros tipos de objetos LogElement, por exemplo, o tipo EVICT, são ignorados.

O suporte write-behind *é* uma extensão do plug-in do Carregador, que você usa para integrar o eXtreme Scale ao banco de dados. Por exemplo, consulte as informações do Configurando Utilitários de Carga do JPA sobre como configurar um carregador JPA.

Benefícios

Ativar o suporte write-behind possui os seguintes benefícios:

- **Isolamento de falha de backend:** O armazenamento em cache write-behind fornece uma camada de isolamento das falhas de backend. Quando o banco de dados de backend falha, as atualizações são enfileiradas no mapa de fila. Os

aplicativos podem continuar a conduzir transações para o eXtreme Scale. Quando o backend se recupera, os dados no mapa de fila são enviados para o backend.

- **Carga de backend reduzida:** O utilitário de carga write-behind mescla as atualizações em uma base de chave, portanto, apenas uma atualização mesclada por chave existe no mapa de fila. Esta mesclagem diminui o número de atualizações no backend.
- **Desempenho de transação aprimorado:** Tempos de transação do eXtreme Scale individuais são reduzidos porque a transação não precisa aguardar até que os dados sejam sincronizados com o backend.

Considerações de Design do Aplicativo

Ativar o suporte write-behind é simples, mas o design de um aplicativo para trabalhar com o suporte write-behind precisa de consideração cuidadosa. Sem o suporte de write-behind, a transação de ObjectGrid engloba a transação de backend. A transação do ObjectGrid inicia antes da transação de backend iniciar e termina após a transação de backend terminar.

Com suporte write-behind ativado, a transação do ObjectGrid é concluída antes que a transação de backend inicie. A transação do ObjectGrid e a transação de backend não estão acopladas.

Limitadores de Integridade Referencial

Cada mapa de apoio que é configurado com suporte write-behind possui seu próprio encadeamento write-behind para enviar os dados para o backend. Portanto, os dados que são atualizados em diferentes mapas em uma transação do ObjectGrid são atualizadas no backend em diferentes transações de backend. Por exemplo, a transação T1 atualiza a chave key1 no mapa Map1 e a chave key2 no mapa Map2. A atualização da key1 para o mapa Map1 é atualizada no backend em uma transação de backend e a key2 atualizada para o mapa Map2 é atualizada no backend em outra transação de backend por encadeamentos write-behind diferentes. Se os dados armazenados no Map1 e Map2 possuírem relações, tais como limitadores de chave estrangeira no backend, as atualizações podem falhar.

Ao projetar os limitadores de integridade referencial em seu banco de dados de backend, certifique-se de que atualizações fora de ordem sejam permitidas.

Comportamento do Bloqueio de Mapa de Fila

Outra grande diferença de comportamento da transação é o comportamento do bloqueio. O ObjectGrid suporta três diferentes estratégias de bloqueio: PESSIMISTIC, OPTIMISITIC e NONE. Os mapas de fila write-behind utilizam a estratégia de bloqueio pessimista não importando qual estratégia de bloqueio está configurada para seu mapa de apoio. Existem dois diferentes tipos de operações que adquirem um bloqueio no mapa de fila:

- Quando uma transação do ObjectGrid é confirmada ou um flush (flush de mapa ou flush de sessão) acontece, a transação lê a chave no mapa de fila e coloca um bloqueio S na chave.
- Quando uma transação do ObjectGrid é confirmada, a transação tenta atualizar o bloqueio S para o bloqueio X na chave.

Devido a este comportamento do mapa de fila extra, é possível visualizar algumas diferenças de comportamento de bloqueio.

- Se o mapa do usuário for configurado como a estratégia de bloqueio PESSIMISTIC, não há muita diferença no comportamento de bloqueio. Sempre que um flush ou commit é chamado, um bloqueio S é colocado na mesma chave no mapa de fila. Durante o momento do commit, um bloqueio X não é adquirido apenas para a chave no mapa do usuário, ele também é adquirido para a chave no mapa de fila.
- Se o mapa do usuário for configurado com a estratégia de bloqueio OPTIMISTIC ou NONE, a transação do usuário seguirá o padrão de estratégia de bloqueio PESSIMISTIC. Sempre que um flush ou commit é chamado, um bloqueio S é adquirido para a mesma chave no mapa de fila. Durante o momento do commit, um bloqueio X é adquirido para a chave no mapa de fila utilizando a mesma transação.

Novas Tentativas de Transações do Utilitário de Carga

O ObjectGrid não suporta transações 2-phase ou XA. O encadeamento write-behind remove registros do mapa de fila e atualiza os registros no backend. Se o servidor falhar no meio da transação, algumas atualizações de backend podem ser perdidas.

O utilitário de carga write-behind automaticamente tentará gravar novamente transações falhas e enviará uma LogSequence duvidosa para o backend para evitar a perda de dados. Esta ação requer que o utilitário de carga seja idempotente, o que significa que o Loader.batchUpdate(TxId, LogSequence) é chamado duas vezes com o mesmo valor, ele fornece o mesmo resultado como se tivesse sido aplicado uma vez. As implementações do utilitário de carga devem implementar a interface RetryableLoader para ativar este recurso. Consulte a documentação da API para obter mais detalhes.

Falha do Utilitário de Carga

O plug-in do utilitário de carga pode falhar quando não consegue se comunicar com o back end do banco de dados. Isto pode acontecer se o servidor de banco de dados ou a conexão de rede estiver inativa. O utilitário de carga write-behind irá enfileirar as atualizações e tentará executar o push das alterações de dados para o utilitário de carga periodicamente. O utilitário de carga deve notificar o tempo de execução do ObjectGrid que há um problema de conectividade do banco de dados lançando uma exceção LoaderNotAvailableException.

Portanto, a implementação do Utilitário de Carga deve poder distinguir uma falha de dados ou uma falha de utilitário de carga físico. A falha de dados deve ser lançada ou relançada como uma LoaderException ou uma OptimisticCollisionException, mas uma falha de utilitário de carga físico deve ser lançada ou relançada como uma LoaderNotAvailableException. O ObjectGrid manipula estas exceções de maneira diferente:

- Se uma LoaderException for capturada pelo utilitário de carga write-behind, o utilitário de carga write-behind a considerará falha devido a alguma falha de dados, tal como uma falha de chave duplicada. O utilitário de carga write-behind irá remover a atualização do lote e tentará atualizar um registro em um momento para isolar a falha de dados. Se uma {{LoaderException}} for capturada durante uma atualização de registro, um registro de atualização falho é criado e registrado no mapa de atualização falho.
- Se uma LoaderNotAvailableException for capturada pelo utilitário de carga write-behind, o utilitário de carga write-behind a considerará falha porque não pode se conectar ao final do banco de dados, por exemplo, porque o backend do

banco de dados estiver inativo, uma conexão com o banco de dados não estiver disponível ou a rede estiver inativa. O utilitário de carga write-behind aguardará por 15 segundo e, em seguida, tentará novamente executar uma atualização de lote no banco de dados.

O erro comum é lançar uma LoaderException enquanto uma LoaderNotAvailableException deve ser lançada. Todos os registros enfileirados no utilitário de carga write-behind se tornarão atualizações de registro falhas, o que frustra o propósito do isolamento de falha do backend.

Considerações sobre Desempenho

O suporte ao armazenamento em cache write-behind aumenta o tempo de resposta removendo a atualização do utilitário de carga da transação. Ele também aumenta o rendimento do banco de dados porque as atualizações de banco de dados são combinadas. É importante compreender o gasto adicional introduzido pelo encadeamento write-behind, que executa o pull dos dados da mapa de fila e executa o push para o utilitário de carga.

A contagem máxima de atualização ou o tempo máximo de atualização necessário a ser ajustado com base nos padrões de uso e no ambiente esperados. Se o valor da contagem máxima de atualização ou o tempo máximo de atualização for muito pequeno, o gasto adicional do encadeamento write-behind pode exceder os benefícios. Configurar um valor maior para estes dois parâmetros também pode aumentar o uso da memória para enfileirar os dados e aumentar o tempo de envelhecimento dos registros do banco de dados.

Para obter um melhor desempenho, ajuste os parâmetros write-behind com base nos seguintes fatores:

- Proporção de transações de leitura e gravação
- Mesma frequência de atualização de registro
- Latência de atualização de banco de dados.

Utilitários de Carga

Com um plug-in Carregador, uma grade de dados pode se comportar como um cache de memória para dados que normalmente são mantidos em um armazenamento persistente no mesmo sistema ou em outro sistema. Geralmente, um banco de dados ou sistema de arquivos é utilizado como o armazenamento persistente. Uma JVM (Java Virtual Machine) também pode ser usada como a origem de dados, permitindo que caches baseados em hub seja construído usando o eXtreme Scale. Um utilitário de carga possui a lógica para leitura e gravação de dados para um armazenamento persistente e a partir dele.

Visão Geral

Os utilitários de carga são plug-ins de mapa de apoio que são chamados quando são feitas alterações no mapa de apoio ou quando o mapa de apoio não pode atender a um pedido de dados (um erro de cache). O utilitário de carga é chamado quando o cache não pode satisfazer uma solicitação para uma chave, fornecendo capacidade read-through e lazy-population do cache. Um utilitário de carga também permite atualizações no banco de dados quando os valores do cache mudam. Todas as mudanças dentro de uma transação são agrupadas para permitir que o número de interações do banco de dados seja minimizado. Um plug-in TransactionCallback é usado em conjunto com o utilitário de carga para acionar a demarcação da transação backend. O uso deste plug-in é importante quando

múltiplos mapas são incluídos em uma única transação ou quando os dados da transação forem enviados para o cache sem consolidação.

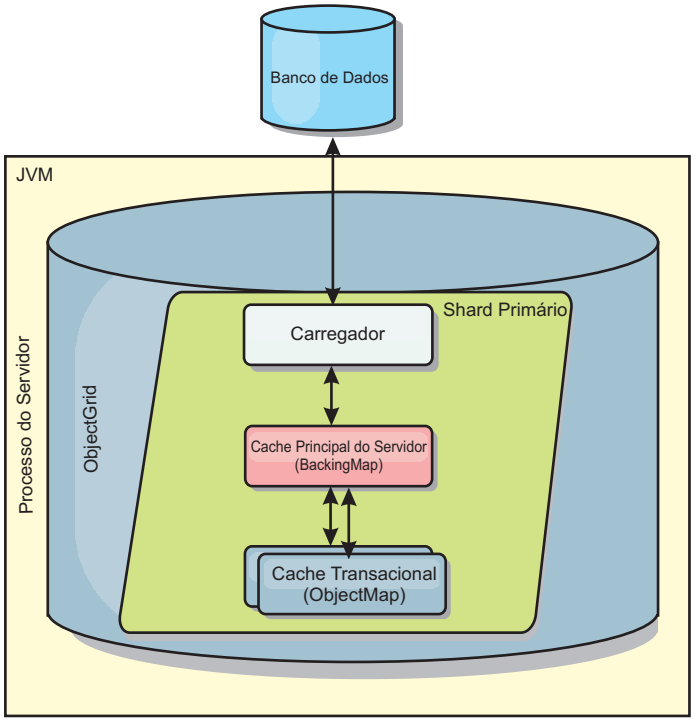


Figura 52. Utilitário de Carga

O utilitário de carga também pode usar atualizações super qualificadas para evitar manter bloqueios do banco de dados. Armazenar um atributo de versão no valor do cache, permite ao utilitário de carga ver a imagem antes e depois do valor quando ele for atualizado no cache. Este valor pode assim ser usado ao atualizar o banco de dados ou backend para verificar se os dados foram atualizados. Um Loader também pode ser configurado para pré-carregar a grade de dados quando for iniciado. Quando particionado, uma instância do utilitário de carga é associada a cada partição. Se o Mapa "Company" tiver dez partições, haverá dez instâncias do utilitário de carga, uma por partição primária. Quando shard primário para o Mapa é ativado, o método preloadMap para o utilitário de carga é chamado síncrona ou assincronamente, o qual permite o carregamento da partição do mapa com dados a partir do backend ocorra automaticamente. Quando chamado sincronamente, todas as transações do cliente são bloqueadas, evitando o acesso inconsistente à grade de dados. Como alternativa, um pré-utilitário de cliente pode ser usado para carregar a grade de dados inteira.

Dois utilitários de carga integrados podem simplificar muito a integração com back ends de banco de dados relacional. Os utilitários de carga JPA utilizam os recursos ORM (Object-Relational Mapping) de ambas as implementações OpenJPA e Hibernate da especificação JPA (Java Persistence API). Consulte “Carregadores JPA” na página 66 para obter mais informações.

Se estiver usando carregadores em uma configuração de diversos datacenters, você deverá considerar como dados de revisão e a consistência de cache são mantidos entre as grades de dados. Para obter informações adicionais, consulte

"Considerações Sobre o Carregador em uma Topologia Multimestre" na página 168.

Configuração do Utilitário de Carga

Para incluir um Utilitário de Carga na configuração do BackingMap, é possível utilizar a configuração programática ou a configuração do XML. Um utilitário de carga possui o seguinte relacionamento com um mapa de apoio.

- Um mapa de apoio pode ter apenas um utilitário de carga.
- Um mapa de apoio de cliente (cache local) não pode ter um utilitário de carga.
- Uma definição de utilitário de carga pode ser aplicado a múltiplos mapas de apoio, mas cada mapa de apoio possui sua própria instância do utilitário de carga.

Pré-carregamento de Dados e Aquecimento

Em vários cenários que incorporam o uso de um carregador, é possível preparar sua grade de dados ao pré-carregá-lo com dados.

Quando usado como um cache completo, a grade de dados deve manter todos os dados e deve ser carregada antes que quaisquer clientes possam se conectar a ele. Quando estiver usando um cache esparso, é possível efetuar um warm-up do cache com dados para que os clientes possam ter acesso imediato aos dados quando eles se conectarem.

Existem duas abordagens para o pré-carregamento de dados na grade de dados: Usando um plug-in do Carregador ou usando um carregador do cliente, conforme descrito nas seguintes seções.

Plug-in do Utilitário de Carga

O plug-in do carregador é associado a cada mapa e é responsável pela sincronização de um único shard de partição primário com o banco de dados. O método preloadMap do plug-in do utilitário de carga é chamado automaticamente quando um shard é ativado. Por exemplo, se você tiver 100 partições, existem 100 instâncias do carregador, cada um carregando os dados para sua partição. Quando executado de modo síncrono, todos os clientes serão bloqueados até que o pré-carregamento seja concluído.

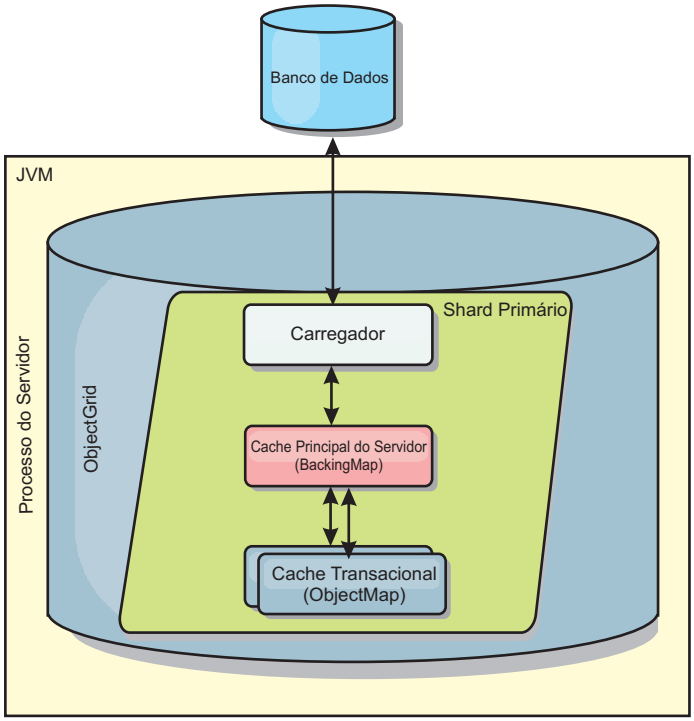


Figura 53. Plug-in do Utilitário de Carga

Utilitário de Carga do Cliente

Um utilitário de carga do cliente é um padrão para uso de um ou mais clientes para carregar a grade com dados. O uso de múltiplos clientes para carregamento de dados da grade pode ser efetivo quando o esquema de partições não está armazenado no banco de dados. É possível chamar os carregadores de cliente manual ou automaticamente quando a grade de dados é iniciada. Os carregadores do cliente podem usar, opcionalmente, o StateManager para configurar o estado da grade de dados no modo de pré-carregamento, para que os clientes não possam acessar a grade enquanto ela estiver pré-carregando os dados. WebSphere eXtreme Scale inclui um carregador baseado em Java Persistence API (JPA)pode ser usado para carregar automaticamente a grade de dados com os provedores JPA OpenJPA ou Hibernate. Para obter mais informações sobre os provedores de cache, consulte “Plug-in do Cache JPA Nível 2 (L2)” na página 26.

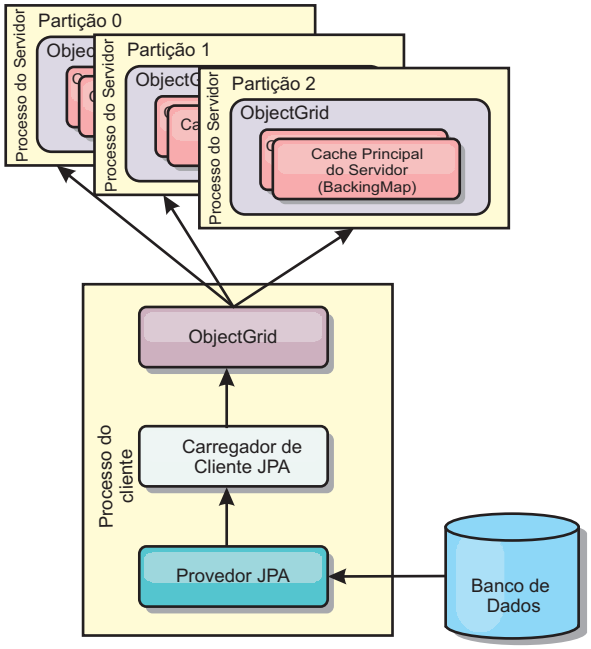


Figura 54. Utilitário de Carga do Cliente

Técnicas de Sincronização de Banco de Dados

Quando o WebSphere eXtreme Scale é utilizado como um cache, os aplicativos devem ser criados para tolerar dados antigos se o banco de dados puder ser atualizado de maneira independente de uma transação do eXtreme Scale. Para atuar como um espaço de processamento de banco de dados de memória sincronizado, o eXtreme Scale fornece várias maneiras de manter o cache atualizado.

Técnicas de Sincronização de Banco de Dados

Atualização periódica

O cache pode ser automaticamente invalidado ou atualizado automaticamente usando o atualizador de banco de dados baseado em tempo JPA (Java Persistence API).O atualizador periodicamente consulta o banco de dados usando um provedor JPA para todas as atualizações ou inserções que ocorreram desde a atualização anterior. Quaisquer alterações identificadas são automaticamente invalidadas ou atualizadas quando utilizadas com um cache disperso. Se utilizadas com um cache completo, as entradas podem ser descobertas e inseridas no cache. As entradas nunca são removidas do cache.

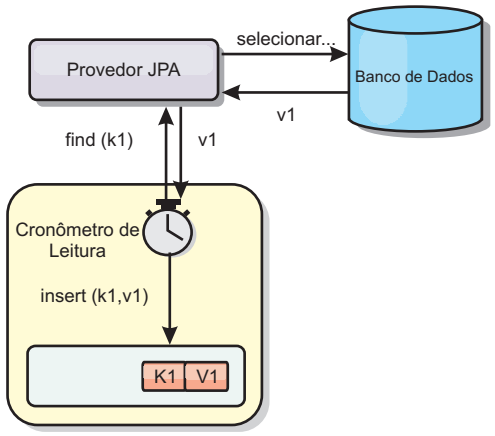


Figura 55. Atualização Periódica

Despejo

Os caches dispersos podem utilizar políticas de despejo para automaticamente remover dados do cache sem afetar o banco de dados. Há três políticas integradas incluídas no eXtreme Scale: time-to-live, least-recently-used e least-frequently-used. Todas as três políticas podem opcionalmente despejar dados mais agressivamente à medida que a memória torna-se restrita ao ativar a opção de despejo baseada em memória.

Invalidação Baseada em Eventos

Caches dispersos e completos podem ser invalidados ou atualizados usando um gerador de eventos como JMS (Java Message Service). A invalidação utilizando JMS pode ser manualmente vinculada a qualquer processo que atualiza o backend utilizando um acionador do banco de dados. Um plug-in ObjectGridEventListener do JMS é fornecido no eXtreme Scale que pode notificar quando o cache do servidor tiver qualquer alteração. Isto pode diminuir a quantidade de tempo que o cliente pode visualizar dados antigos.

Invalidação programática

As APIs do eXtreme Scale permitem interação manual do cache local e do servidor usando os métodos de API Session.beginNoWriteThrough(), ObjectMap.invalidate() e EntityManager.invalidate(). Se um processo do cliente ou servidor não precisar mais de uma parte dos dados, os métodos de invalidação podem ser utilizados para remover dados do cache local ou do servidor. O método beginNoWriteThrough aplica qualquer operação ObjectMap ou EntityManager para o cache local sem chamar o utilitário de carga. Se chamada a partir de um cliente, a operação é aplicável apenas para o cache local (o utilitário de carga remoto não é chamado). Se chamada no servidor, a operação é aplicável apenas ao cache principal do servidor sem chamar o utilitário de carga.

Invalidação de Dados

Para remover os dados de cache de escala, é possível usar um mecanismo de invalidação programático ou baseado em evento.

Invalidação Baseada em Evento

Caches dispersos e completos podem ser invalidados ou atualizados usando um gerador de eventos como JMS (Java Message Service). A invalidação utilizando JMS pode ser manualmente vinculada a qualquer processo que atualiza o backend utilizando um acionador do banco de dados. É fornecido um plug-in JMS ObjectGridEventListener no eXtreme Scale que pode notificar os clientes quando o cache do servidor é alterado. Esse tipo de notificação diminui a quantidade de tempo em que o cliente pode ver dados antigos.

A invalidação baseada em evento normalmente consiste nos três componentes a seguir.

- **Fila de eventos:** Uma fila de eventos armazena os eventos de mudança de dados. Ela pode ser uma fila JMS, um banco de dados, uma fila FIFO em memória ou qualquer tipo de manifesto, contanto que possa gerenciar os eventos de mudança de dados.
- **Publicador de evento:** Um publicador de evento publica os eventos de mudança de dados na fila de eventos. Um publicador de evento geralmente é um aplicativo que você cria ou uma implementação de plug-in do eXtreme Scale. O publicador de evento sabe quando os dados são alterados ou quando ele mesmo altera os dados. Quando uma transação é confirmada, eventos são gerados para os dados alterados e o publicador de eventos publica esses eventos na fila de eventos.
- **Consumidor de evento:** Um consumidor de evento consome eventos de mudança de dados. O consumidor de evento geralmente é um aplicativo para garantir que os dados da grade de destino sejam atualizados com a mudança mais recente das outras grades. Esse consumidor de evento interage com a fila de eventos para obter a mudança de dados mais recente, além de aplicar as mudanças de dados na grade de destino. Os consumidores de evento podem utilizar APIs do eXtreme Scale para invalidar dados antigos ou atualizar a grade com os dados mais recentes.

Por exemplo, JMSObjectGridEventListener tem uma opção para um modelo de cliente/servidor, no qual a fila de eventos é um destino JMS designado. Todos os processos do servidor são publicadores de eventos. Quando uma transação é confirmada, o servidor obtém as mudanças de dados e as publica no destino JMS designado. Todos os processos do cliente são consumidores de evento. Eles recebem as mudanças de dados do destino JMS designado e aplicam as mudanças no cache perto do cliente.

Consulte o tópico sobre a ativação do mecanismo de invalidação de cliente no *Guia de Administração* para obter mais informações.

Invalidação Programática

As APIs do WebSphere eXtreme Scale permitem interação manual do cache local e do servidor usando os métodos de API Session.beginNoWriteThrough(), ObjectMap.invalidate() e EntityManager.invalidate(). Se um processo do cliente ou servidor não precisar mais de uma parte dos dados, os métodos de invalidação podem ser utilizados para remover dados do cache local ou do servidor. O método beginNoWriteThrough aplica qualquer operação ObjectMap ou EntityManager para o cache local sem chamar o utilitário de carga. Se chamada a partir de um cliente, a operação é aplicável apenas para o cache local (o utilitário de carga remoto não é chamado). Se chamada no servidor, a operação é aplicável apenas ao cache principal do servidor sem chamar o utilitário de carga.

É possível utilizar invalidação programática com outras técnicas para determinar quando invalidar os dados. Por exemplo, esse método de invalidação utiliza mecanismos de invalidação baseados em evento para receber eventos de mudança de dados e depois utiliza as APIs para invalidar os dados antigos.

Indexação

Use o plug-in MapIndexPlugin para construir um índice ou vários índices em um BackingMap para suportar acesso a dados sem chave.

Tipos e Configuração de Índice

O recurso de indexação é representado pelo plug-in MapIndexPlugin ou Index, para abreviar. O Índice é um plug-in BackingMap. Um BackingMap pode ter múltiplos plug-ins de Índice configurados, enquanto cada um seguir as regras de configuração de Índice.

O recurso de indexação pode ser usado para construir um ou mais índices em um BackingMap. Um índice é construído a partir de um atributo ou uma lista de atributos de um objeto no BackingMap. Este recurso fornece uma maneira para os aplicativos localizarem determinados objetos mais rapidamente. Com o recurso de indexação, os aplicativos podem localizar objetos com um valor específico ou em um intervalo com os valores de atributos indexados.

Dois tipos de indexação são possíveis: estática e dinâmica. Com a indexação estática, é necessário configurar o plug-in de índice no BackingMap antes de inicializar a instância do ObjectGrid. É possível fazer esta configuração com a configuração XML ou programática do BackingMap. A indexação estática inicia a construção de um índice durante a inicialização do ObjectGrid. O índice é sempre sincronizado com o BackingMap e está pronto para utilização. Depois de o processo de indexação estático iniciar, a manutenção do índice é parte do processo de gerenciamento de transação do eXtreme Scale. Quando as consolidações de transações mudam, estas alterações também atualizam o índice estático, e as alterações de índice são recuperadas se a transação for recuperada.

Com a indexação dinâmica, é possível criar um índice num BackingMap antes ou depois da inicialização da instância do ObjectGrid que o contém. Os aplicativos possuem controle de ciclo de vida sobre o processo de indexação dinâmica para que você possa remover um índice dinâmico quando ele não for mais necessário. Quando um aplicativo cria um índice dinâmico, o índice pode não estar pronto para utilização imediata devido ao tempo gasto na conclusão do processo de construção do índice. Como a quantidade de tempo depende da quantidade de dados indexados, a interface DynamicIndexCallback é fornecido para aplicativos que desejam receber notificações quando ocorrem determinados eventos de indexação. Estes eventos incluem ready, error e destroy. Os aplicativos podem implementar esta interface de retorno de chamada e registrar-se no processo de indexação dinâmica.

Se um BackingMap tiver um plug-in de índice configurado, você pode obter o objeto de proxy do índice do aplicativo a partir do ObjectMap correspondente. Chamar o método getIndex no ObjectMap e passar o nome do plug-in de índice retornam o objeto de proxy do índice. Você deve efetuar o cast do objeto de proxy do índice para uma interface de índice adequada do aplicativo, como MapIndex, MapRangeIndex ou uma interface de índice customizada. Após obter o objeto de proxy do índice, é possível utilizar métodos definidos na interface do índice do aplicativo para localizar objetos armazenados em cache.

As etapas para utilizar a indexação estão resumidas na lista a seguir:

- Incluir plug-ins de indexação, estáticos ou dinâmicos, no BackingMap;
- Obter um objeto de proxy de indexação do aplicativo, emitindo o método getIndex do ObjectMap.
- Direcione o objeto de proxy de índice a uma interface de índice de aplicativo apropriado, como MapIndex, MapRangeIndex, ou uma interface de índice customizada.
- Utilizar os métodos definidos na interface Index do aplicativo para localizar objetos no cache.

A classe HashIndex é a implementação de plug-in de índice integrada que pode suportar ambas as interfaces integradas de índice do aplicativo: MapIndex e MapRangeIndex. Também é possível criar seus próprios índices. É possível incluir o HashIndex como um índice estático ou dinâmico no BackingMap, obter o objeto proxy do índice MapIndex ou MapRangeIndex e usar o objeto proxy do índice para localizar objetos em cache.

Índice Padrão

Se desejar iterar por meio das chaves em um mapa local, o índice padrão poderá ser usado. Este índice não requer nenhuma configuração, porém ele deve ser usado com relação ao shard, usando um agente ou uma instância do ObjectGrid recuperados a partir do método ShardEvents.shardActivated(shard do ObjectGrid).

Consideração sobre a Qualidade dos Dados

Os resultados dos métodos de consulta de índice somente representar uma captura instantânea de dados em um determinado ponto no tempo. Nenhum bloqueio contra as entradas de dados é obtido depois do retorno dos resultados para o aplicativo. O aplicativo deve estar ciente de que podem ocorrer atualizações de dados em um conjunto de dados retornado. Por exemplo, o aplicativo obtém a chave de um objeto armazenado em cache executando o método findAll de MapIndex. Este objeto de chave retornado está associado a uma entrada de dados no cache. O aplicativo deve poder executar o método get no ObjectMap para localizar um objeto fornecendo o objeto chave. Se outra transação remover o objeto de dados do cache imediatamente antes de o método get ser chamado, o resultado retornado será nulo.

Considerações sobre Desempenho de Indexação

Um dos principais objetivos do recurso de indexação é melhorar o desempenho geral do BackingMap. Se a indexação não for utilizada corretamente, o desempenho do aplicativo poderá ficar comprometido. Considere os seguintes fatores antes de usar este retorno.

- **A quantidade de transações de gravação concorrentes:** O processamento de índices pode ocorrer todas as vezes que uma transação gravar dados em um BackingMap. O desempenho será afetado se muitas transações gravarem dados no mapa simultaneamente quando um aplicativo tentar operações de consulta ao índice.
- **O tamanho do conjunto de resultados que é retornado por uma operação de consulta:** Como o tamanho do conjunto de resultados aumenta, o desempenho da consulta cai. O desempenho tente a degradas quando o tamanho do conjunto de resultados é de 15% (ou mais) do BackingMap.

- **A quantidade de índices construídos sobre o mesmo BackingMap:** Cada índice consome os recursos do sistema. Conforme a quantidade dos índices construídos sobre o BackingMap aumenta, o desempenho diminui.

A função de indexação pode aumentar significativamente o desempenho do BackingMap. Os casos ideais ocorrem quando o BackingMap possui a maioria de operações de leitura, o conjunto de resultados da consulta é de uma porcentagem pequena das entradas do BackingMap, e somente poucos índices são construídos sobre o BackingMap.

Planejando Diversas Topologias do Datacenter

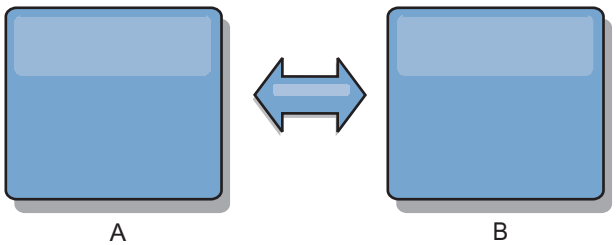
Ao usar a replicação assíncrona multimestre, duas ou mais grades de dados podem se tornar cópias exatas de uns dos outros. Cada grade de dados é hospedada em um domínio do serviço de catálogo independente, com seu próprio serviço de catálogo, servidores de contêiner e um nome exclusivo. Com a replicação assíncrona multimestre, é possível usar links para conectar uma coleção de domínios do serviço de catálogo. Os domínios do serviço de catálogo são então sincronizados usando a replicação sobre os links. É possível construir quase qualquer topologia por meio da definição de links entre os domínios de serviço de catálogo.

Topologias de Replicação Multimestre

Há várias opções diferentes quando escolher a topologia para sua implementação que incorpora a replicação multimestre.

Links Conectando Domínios de Serviço de Catálogos

Uma infraestrutura de grade de dados de replicação é um gráfico conectado dos domínios do serviço de catálogo com links bidirecionais entre eles. Com um link, dois domínios de serviço de catálogo podem comunicar as mudanças de dados. Por exemplo, a topologia mais simples é um par de domínios do serviço de catálogo com um único link entre eles. Os domínios do serviço de catálogo são denominados em ordem alfabética: A, B, C e assim por diante, a partir da esquerda. Um link pode cruzar uma rede de longa distância (WAN), expandindo-se para grandes distâncias. Mesmo se o link for interrompido, os dados ainda poderão ser alterados em qualquer um dos domínios de serviço de catálogo. A topologia reconcilia as mudanças quando o link reconecta os domínios do serviço de catálogo. Os links tentarão automaticamente reconectar se a conexão com a rede for interrompida.

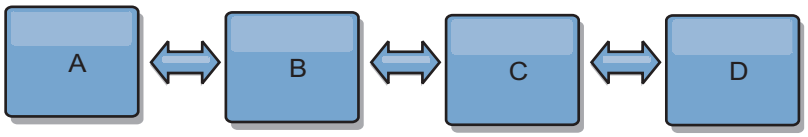


Após configurar os links, o eXtreme Scale primeiro tentará tornar idêntico cada domínio do serviço de catálogo. Em seguida, o eXtreme Scale tenta manter as condições idênticas conforme as mudanças ocorrerem em qualquer domínio de serviço de catálogo. O objetivo é que cada domínio de serviço de catálogo seja um espelho exato de qualquer outro domínio de serviço de catálogo conectado pelos links. Os links de replicação entre os domínios de serviço de catálogo ajudam a

assegurar que quaisquer mudanças feitas em um domínio sejam copiadas para os outros domínios.

Topologias em Linha

Embora esta seja uma implementação muito simples, uma topologia em linha demonstra algumas qualidades dos links. Primeiro, um domínio de serviço de catálogo não precisa estar conectado diretamente a outro domínio de serviço de catálogo para receber as mudanças. O Domínio B pega as mudanças de Domínio A. O Domínio C recebe mudanças do Domínio A por meio do Domínio B, que se conecta os Domínios A e C. Da mesma forma, o Domínio D recebe as mudanças dos outros domínios por meio do Domínio C. Esta habilidade envia o carregamento das mudanças de distribuição para longe da origem das mudanças.



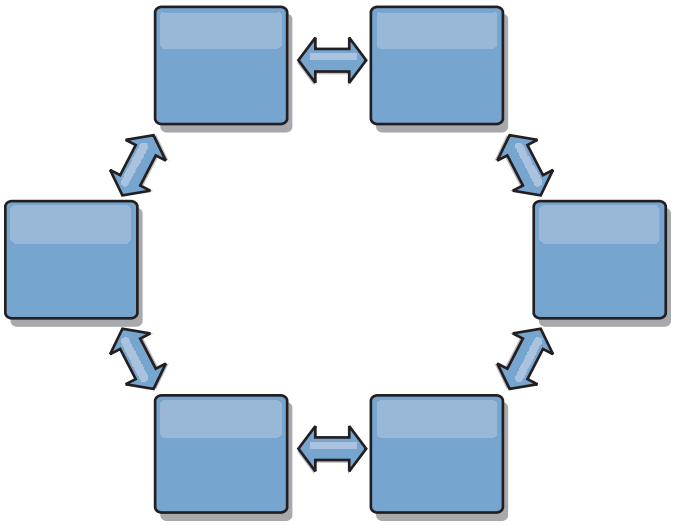
Observe que, se o Domínio C falhar, as seguintes ações ocorrerão:

1. O Domínio D pode ficar órfão até o Domínio C ser reiniciado
2. O Domínio C pode se sincronizar com o Domínio B, que é uma cópia do Domínio A
3. O Domínio D pode usar o Domínio C para se sincronizar com as mudanças nos Domínios A e B. Essas mudanças inicialmente ocorreram enquanto o Domínio D estava órfão (enquanto o Domínio C estava inativo).

Por fim, os Domínios A, B, C e D podem se tornar todos idênticos entre si novamente.

Topologias em Anel

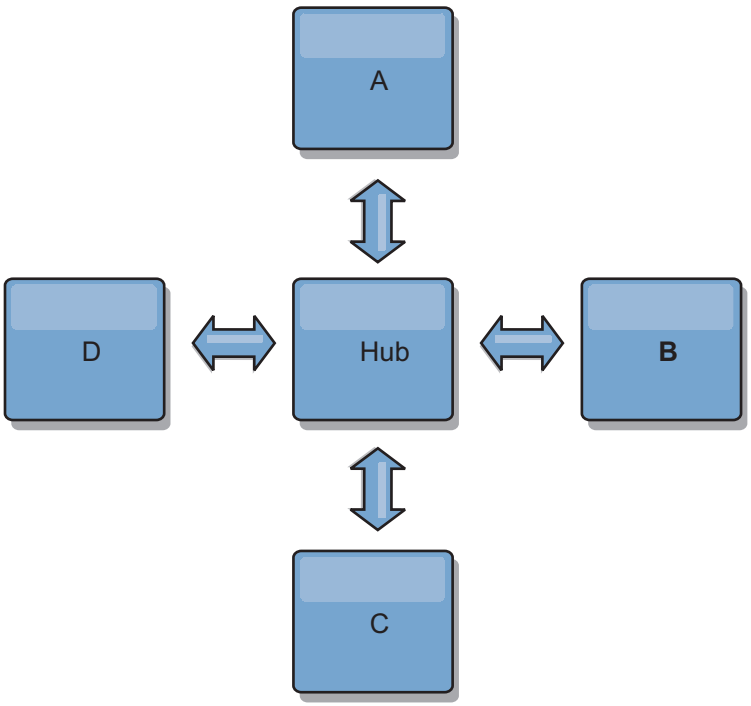
As topologias em anel são um exemplo de uma topologia mais resiliente. Quando um domínio do serviço de catálogo ou um único link falhar, os domínios do serviço de catálogo sobreviventes ainda podem obter as mudanças. Os domínios de serviço de catálogo se deslocam ao redor do anel, longe da falha. Cada domínio de serviço de catálogo tem no máximo dois links para outros domínios de serviço de catálogo, independente do tamanho da topologia em anel. A latência para propagar as mudanças pode ser grande. As mudanças a partir de um domínio de serviço de catálogo particular podem precisar percorrer vários links antes que todos os domínios de serviço de catálogo possuam as mudanças. Uma topologia em linha tem a mesma característica.



Também é possível implementar uma topologia em anel mais sofisticada, com um domínio de serviço de catálogo raiz no centro do anel. O domínio de serviço de catálogo raiz funciona como o ponto central da reconciliação. Os outros domínios de serviço de catálogo atuam como pontos remotos de reconciliação para que as mudanças ocorram no domínio de serviço de catálogo raiz. O domínio de serviço de catálogo raiz pode arbitrar mudanças entre os domínios de serviço de catálogo. Se uma topologia em anel contiver mais de um anel ao redor de um domínio de serviço de catálogo raiz, o domínio poderá arbitrar apenas as mudanças entre o anel mais interno. No entanto, os resultados da arbitragem se propagam por todos os domínios de serviço de catálogo nos outros anéis.

Topologias Hub-and-Spoke

Com uma topologia hub-and-spoke, as mudanças se deslocam por um domínio de serviço de catálogo hub. Como o hub é apenas o domínio de serviço de catálogo intermediário que é especificado, as topologias hub e spoke possuem baixa latência. O domínio hub é conectado a cada domínio de spoke por meio de um link. O hub distribui as mudanças entre os domínios de serviço de catálogo. O hub atua como um ponto de reconciliação de colisões. Em um ambiente com uma alta taxa de atualização, o hub pode precisar executar em mais hardware do que os spokes devem permanecer sincronizados. O WebSphere eXtreme Scale é projetado para escalar linearmente, o que significa que é possível aumentar o hub, conforme necessário, sem dificuldade. No entanto, se o hub falhar, as mudanças não são distribuídas até ele ser reiniciado. As mudanças nos domínios de serviço de catálogo spoke serão distribuídas depois que o hub for reconectado.



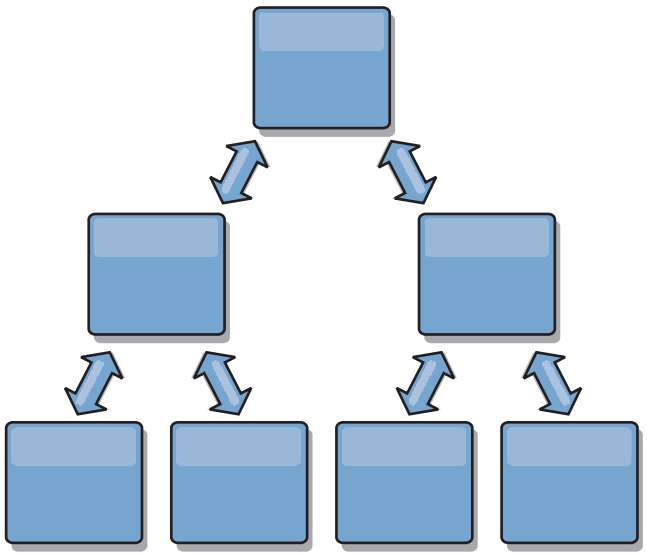
Também é possível usar uma estratégia com clientes totalmente replicados, uma variação de topologia que usa um par de servidores eXtreme Scale sendo executados como um hub. Cada cliente cria uma grade de dados autocontida de contêiner único com um catálogo na JVM do cliente. Um cliente usa a grade de dados para se conectar com o catálogo do hub. Esta conexão faz com que o cliente seja sincronizado com o hub assim que o cliente estabelecer uma conexão com o hub.

Todas as mudanças feitas pelo cliente são locais para o cliente e são replicadas de forma assíncrona para o hub. O hub age como um domínio de arbitragem, distribuindo mudanças para todos os clientes conectados. A topologia de clientes totalmente replicados fornece um cache L2 confiável para um mapeador relacional de objeto, como OpenJPA. As mudanças serão distribuídas rapidamente entre as JVMs de cliente por meio do hub. Se o tamanho do cache puder estar contido no espaço de heap disponível, a topologia será uma arquitetura confiável para este estilo de L2.

Use várias partições para escalar o domínio do hub em várias JVMs, se necessário. Como todos os dados ainda devem se ajustar em uma única JVM de cliente, diversas partições aumentam a capacidade do hub para distribuir e arbitrar as mudanças. No entanto, ter diversas partições não altera a capacidade de um único domínio.

Topologias em Árvore

Também é possível usar uma árvore direcionada acíclica. Uma árvore acíclica não tem ciclos ou loops e uma configuração direcionada limita a criação de links apenas entre pais e filhos. Esta configuração pode ser útil para topologias que possuem muitos domínios de serviço de catálogo e não é viável ter um hub central conectado a cada spoke possível. Este tipo de topologia também pode ser útil quando os domínios de serviço de catálogo filho tiverem que ser incluídos sem atualizar o domínio de serviço de catálogo raiz.



Uma topologia em árvore ainda pode ter um ponto central de reconciliação no domínio de serviço de catálogo raiz. O segundo nível pode ainda funcionar como um ponto de reconciliação remoto para que as mudanças ocorram no domínio de serviço de catálogo abaixo deles. O domínio do serviço de catálogo raiz pode arbitrar as mudanças entre os domínios de serviço de catálogo apenas no segundo nível. Também é possível usar N árvores, cada uma delas possuindo N filhos em cada nível. Cada domínio de serviço de catálogo se conecta com *n* links .

Clientes Totalmente Replicados

Essa variação de topologia envolve um par de servidores do eXtreme Scale sendo executados como um hub. Cada cliente cria uma grade de dados autocontida de contêiner único com um catálogo na JVM do cliente. Um cliente usa a grade de dados para se conectar com o catálogo do hub, fazendo com que o cliente seja sincronizado com o hub assim que estabelecer uma conexão com o hub.

Todas as mudanças feitas pelo cliente são locais para o cliente e são replicadas de forma assíncrona para o hub. O hub age como um domínio de arbitragem, distribuindo mudanças para todos os clientes conectados. A topologia de clientes totalmente replicada fornece um bom cache L2 para um mapeador relacional de objeto, como OpenJPA. As mudanças serão distribuídas rapidamente entre as JVMs de cliente por meio do hub. Contanto que o tamanho do cache possa estar contido no espaço de heap disponível dos clientes, esta topologia será uma boa arquitetura para este estilo de L2.

Use várias partições para escalar o domínio do hub em várias JVMs, se necessário. Como todos os dados ainda devem se ajustar em uma única JVM de cliente, usar várias partições aumenta a capacidade do hub para distribuir e arbitrar mudanças, mas não altera a capacidade de um único domínio.

Considerações de Configuração para Topologias Multimestre

Considere os seguintes problemas ao decidir se e como usar as topologias de replicação multimestre.

- **Requisitos do conjunto de mapa**

  Os conjuntos de mapa devem ter as seguintes características para replicar as mudanças nos links de domínio do serviço de catálogo:

- O nome do ObjectGrid e o nome do conjunto de mapas dentro de um domínio do serviço de catálogo devem corresponder ao nome do ObjectGrid e ao nome do conjunto de mapas de outros domínios de serviço de catálogo. Por exemplo, o ObjectGrid "og1" e o conjunto de mapas "ms1" devem ser configurados no domínio do serviço de catálogo A e no domínio do serviço de catálogo B para replicar os dados no conjunto de mapas entre os domínios do serviço de catálogo.
- Ser uma grade de dados `FIXED_PARTITION`. As grades de dados `PER_CONTAINER` não podem ser replicadas.
- Ter o mesmo número de partições em cada domínio do serviço de catálogo. O conjunto de mapa pode ou não ter o mesmo número e tipos de réplicas.
- Ter os mesmos tipos de dados sendo replicados em cada domínio do serviço de catálogo.
- Contém os mesmos mapas e modelos de mapas dinâmicos em cada domínio do serviço de catálogo.
- Não usar o gerenciador de entidades. Um conjunto de mapas contendo um mapa de entidade não é replicado entre os domínios do serviço de catálogo.
- Não usar o suporte de armazenamento em cache write-behind. Um conjunto de mapas contendo um mapa configurado com o suporte write-behind não é replicado entre os domínios de serviço de catálogo.

Quaisquer conjuntos de mapas com as características anteriores começam a serem replicados depois que os domínios do serviço de catálogo na topologia forem iniciados.

- **Carregadores de classe com diversos domínios do serviço de catálogo**

  Os domínios do serviço de catálogo devem ter acesso a todas as classes que são usadas como chaves e valores. Todas as dependências devem ser refletidas em todos os caminhos da classe para as Java virtual machine do contêiner da grade para todos os domínios. Se um plug-in CollisionArbiter recuperar o valor para uma entrada de cache, as classes para os valores deverão estar presentes para o domínio que está iniciando o mecanismo de resolução de conflitos.

Considerações Sobre o Carregador em uma Topologia Multimestre

Quando estiver usando os carregadores em uma topologia multimestre, você deve considerar a possibilidade de colisão e desafios de manutenção das informações de revisão. A grade de dados mantém as informações de revisão sobre os itens nela para que colisões possam ser detectadas quando outros shards primários na configuração gravarem entradas na grade de dados. Quando as entradas são incluídas a partir de um carregador, essas informações de revisão não são incluídas e a entrada assume uma nova revisão. Como a revisão da entrada parece ser uma nova inserção, uma colisão false poderá ocorrer se outro shard primário também alterar esse estado ou obtiver as mesmas informações a partir de um carregador.

As mudanças na replicação chamam o método get no carregador com uma lista das chaves que ainda não estão na grade de dados, porém serão alteradas durante a transação da replicação. Quando a replicação ocorre, essas entradas são entradas de colisão. Quando as colisões são definidas e a revisão é aplicada, uma atualização em lote é chamada no carregador para aplicar as mudanças no banco de dados. Todos os mapas que foram alterados na janela de revisão são atualizados na mesma transação.

Desafio do Pré-Carregamento

Considere uma topologia de dois datacenters, com o datacenter A e o datacenter B. Ambos os datacenters possuem bancos de dados independente, mas apenas o datacenter A possui uma grade de dados em execução. Quando você estabelece um link entre os datacenters por uma configuração multimestre, as grades de dados no datacenter A começam a enviar dados para as novas grades de dados do datacenter B, causando uma colisão com cada entrada. Outro maior problema que ocorre é com os dados que estão no banco de dados do datacenter B, mas não no banco de dados no datacenter A. Essas linhas não são preenchidas e definidas, resultando em inconsistências que não são resolvidas.

Solução para o Desafio de Pré-Carregamento

Como os dados que residem apenas no banco de dados não podem ter revisões, a grade de dados sempre deve ser pré-carregada completamente a partir do banco de dados local antes de estabelecer o link multimestre. Em seguida, ambas as grades de dados podem revisar e definir os dados, eventualmente chegando a um estado consistente.

Desafio de Cache Disperso

Com um cache disperso, o primeiro aplicativo tenta localizar os dados na grade de dados. Se os dados não estiverem na grade de dados, eles serão procurados no banco de dados usando o carregador. As entradas são despejadas da grade de dados periodicamente para manter um tamanho de cache pequeno.

Este tipo de cache pode ser problemático em um cenário de configuração multimestre porque as entradas dentro da grade de dados possuem metadados de revisão que ajudam a detectar quando colisões ocorrem e qual lado fez as mudanças. Quando os links entre os datacenters não estiverem funcionando, um datacenter pode atualizar uma entrada e depois, eventualmente, atualizar o banco de dados e invalidar a entrada na grade de dados. Quando o link é recuperado, os datacenters tentam sincronizar as revisões entre si. No entanto, como o banco de dados foi atualizado e a entrada da grade de dados foi invalidada, a mudança é perdida a partir da perspectiva do datacenter que ficou inativo. Como resultado, os dois lados da grade de dados estão fora de sincronização e não estão consistentes.

Solução para o Desafio de Cache Disperso

Topologia Hub e Spoke:

É possível executar o carregador apenas no hub de uma topologia de hub e spoke, mantendo a consistência dos dados, enquanto a escala da grade de dados é ajustada. No entanto, se você estiver considerando essa implementação, observe que os carregadores podem permitir que a grade de dados seja parcialmente carregada, significando que um evictor foi configurado. Se o spokes de sua configuração forem caches dispersos, mas não possuírem nenhum carregador, quaisquer perdas de cache não terão como recuperar dados do banco de dados. Devido a esta restrição, você deve usar uma topologia de cache totalmente preenchida com uma configuração de hub e spoke.

Invalidações e Despejo

Uma invalidação cria inconsistência entre a grade de dados e o banco de dados. Os dados podem ser removidos da grade de dados, seja de modo programático ou

com o despejo. Ao desenvolver seu aplicativo, você deve estar ciente de que a manipulação de revisão não replica mudanças que são invalidadas, resultando em inconsistências entre os shards primários.

Os eventos de invalidação não são mudanças de estado de cache e não resultam em replicação. Todos os evictors configurados são executados independentemente de outros evictors na configuração. Por exemplo, você pode ter um evictor configurado para um limite de memória em um domínio do serviço de catálogo, e um tipo diferente de evictor menos agressivo no outro domínio do serviço de catálogo vinculado. Quando as entradas da grade dedados são removidas devido à política de limite de memória, as entradas no outro domínio do serviço de catálogo não são afetadas.

Atualizações do banco de dados e invalidação da grade de dados

Problemas ocorrem quando o banco de dados é atualizado diretamente no plano de fundo ao chamar a invalidação na grade de dados para as entradas atualizadas em uma configuração multimestre. Esse problema ocorre porque a grade de dados não pode replicar a mudança para os outros shards primários até que algum tipo de acesso ao cache mova a entrada para a grade de dados.

Diversos Gravadores para um Único Banco de Dados Lógico

Quando um banco de dados único é usado com diversos shards primários conectados por meio de um carregador, isso pode resultar em conflitos transacionais. Sua implementação do carregador deve manipular especialmente esses tipos de cenários.

Espelhando Dados Usando Replicação Multimestre

É possível configurar bancos de dados independentes que estão conectados a domínios do serviço de catálogo independentes. Nessa configuração, o carregador pode enviar mudanças de um datacenter para o outro.

Considerações de Design para Replicação Multimestre

Ao implementar da replicação multimestre, você deve considerar aspectos de design, como arbitragem, vinculação e desempenho.

Considerações sobre Arbitragem no Design de Topologia

Poderão ocorrer colisões de mudanças se os mesmos registros puderem ser alterados simultaneamente em dois locais. Configure cada domínio de serviço de catálogo para ter aproximadamente a mesma quantia de processador, memória e recursos de rede. Observe que os domínios de serviço de catálogo que executam a manipulação da colisão de mudanças (arbitragem) usam mais recursos que outros domínios de serviço de catálogo. As colisões são detectadas automaticamente. Elas são manipuladas com um desses dois mecanismo:

- **Árbitro de conflito padrão**: O protocolo padrão deve usar as mudanças a partir do domínio de serviço de catálogo mais baixo denominado de maneira lexical. Por exemplo, se os domínios de serviço de catálogo A e B gerarem um conflito para um registro, a mudança no domínio de serviço de catálogo B será ignorada. O domínio de serviço de catálogo A mantém sua versão e o registro no domínio de serviço de catálogo B é alterado para corresponder ao registro do domínio de serviço de catálogo A. Esse comportamento também se aplica aos aplicativos nos quais os usuários ou as sessões são normalmente ligados ou têm afinidade com uma das grades de dados.

- **Árbitro de colisão customizado**: Os aplicativos podem fornecer um árbitro customizado. Quando um domínio do serviço de catálogo detecta uma colisão, ele inicia o árbitro. Para obter informações sobre como desenvolver um árbitro útil customizado, consulte Desenvolvendo Árbitros Customizados para a Replicação Multimestre.

Para topologias nas quais as colisões são possíveis, considere implementar uma topologia hub-and-spoke ou uma topologia em árvore. Essas duas topologias tendem a evitar colisões constantes, o que pode acontecer nos seguintes cenários:

1. Diversos domínios de serviço de catálogo experimentam uma colisão.
2. Cada domínio de serviço de catálogo manipula a colisão localmente, produzindo revisões.
3. As revisões colidem, resultando em revisões de revisões

Para evitar colisões, escolha um domínio de serviço de catálogo específico, chamado de *domínio de serviço de catálogo de arbitragem* como o árbitro de colisão para um subconjunto de domínios de serviço de catálogo. Por exemplo, uma topologia hub-and-spoke pode usar o hub como o manipulador de colisão. O manipulador de colisão spoke ignora quaisquer colisões que forem detectadas pelos domínios de serviço de catálogo spoke. O domínio de serviço de catálogo do hub cria revisões, evitando revisões de colisão inesperadas. O domínio de serviço de catálogo designado para manipular colisões deve ser vinculado a todos os domínios os quais ele é responsável por manipular as colisões. Em uma topologia em árvore, os domínios pais internos manipulam colisões para seus filhos imediatos. Em contraste, se usar uma topologia em anel, não será possível designar um domínio de serviço de catálogo no anel como o árbitro.

A tabela a seguir resume as abordagens de arbitragem que são mais compatíveis com várias topologias.

Tabela 8. Abordagens de Arbitragem. Esta tabela define se a arbitragem de aplicativo é compatível com várias tecnologias.

Topologia	Arbitragem do Aplicativo?	Notes
Uma linha de dois domínios de serviço de catálogo	Sim	Escolha um domínio do serviço de catálogo como o árbitro.
Uma linha de três domínios de serviço de catálogo	Sim	O domínio de serviço de catálogo médio deve ser o árbitro. Considere o domínio de serviço de catálogo médio como sendo o hub em uma topologia hub-and-spoke simples.
Uma linha de mais de três domínios de serviço de catálogo	Não	A arbitragem de aplicativo não é suportada.
Um hub com N spokes	Sim	O hub com links para todos os spokes deve ser o domínio de serviço de catálogo de arbitragem.
Um anel de N domínios de serviço de catálogo	Não	A arbitragem de aplicativo não é suportada.
Uma árvore acíclica, direcionada (árvore n-ary)	Sim	Todos os nós-raiz devem classificar apenas seus descendentes diretos.

Considerações sobre Links no Design de Topologia

De forma ideal, uma topologia inclui um número mínimo de links enquanto otimiza trade-offs entre latência de mudança, tolerância a falhas e características de desempenho.

- **Latência de mudança**

  A latência de mudança é determinada pelo número de domínios de serviço de catálogo intermediários onde uma mudança deve passar antes de chegar a um domínio do serviço de catálogo específico.

  Uma topologia tem a melhor latência de mudança quando ele elimina os domínios de serviço de catálogo intermediários ao vincular cada domínio de serviço de catálogo a outro domínio. No entanto, um domínio de serviço de catálogo deve executar o trabalho de replicação proporcionalmente ao seu número de links. Para topologias grandes, o número de links absoluto a ser definido pode causar uma carga administrativa.

  A velocidade com que uma mudança é copiada para outros domínios do serviço de catálogo depende de fatores adicionais, como:

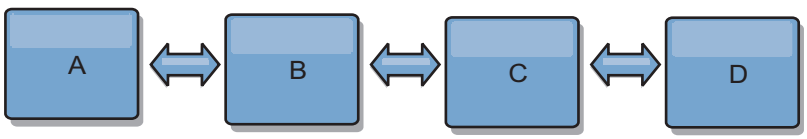
  – Processador e largura da banda da rede no domínio de serviço de catálogo de origem
  – O número de domínios de serviço de catálogo intermediários e de links entre o domínio de serviço de catálogo de origem e de destino
  – Os recursos de processador e de rede disponíveis para os domínios de serviço de catálogo de origem, de destino e intermediários

- **Tolerância a falhas**

  A tolerância a falhas é determinada pela quantia de caminhos que existem entre dois domínios de serviço de catálogo para a replicação de mudança.

  Se houver apenas um link entre um determinado par de domínios de serviço de catálogo, uma falha de link não permitirá a propagação das mudanças. Da mesma forma, as mudanças não serão propagadas entre os domínios de serviço de catálogo se ocorrer uma falha de link em qualquer um dos domínios intermediários. Sua topologia pode ter um único link a partir de um domínio de serviço de catálogo para outro, de modo que o link passe pelos domínios intermediários. Caso positivo, as mudanças não serão propagadas se qualquer um dos domínios de serviço de catálogo intermediários estiver inativo.

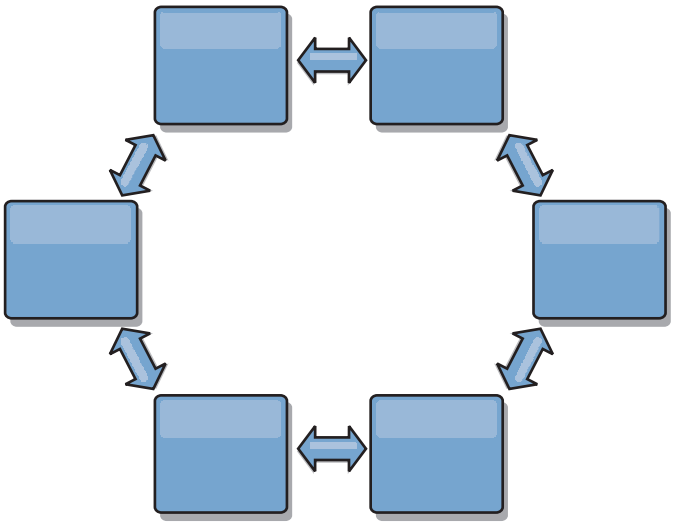
  Considere a topologia em linha com quatro domínios de serviço de catálogo, A, B, C e D:

  Se qualquer uma dessas condições for mantida, o Domínio D não verá nenhuma mudança a partir do A:

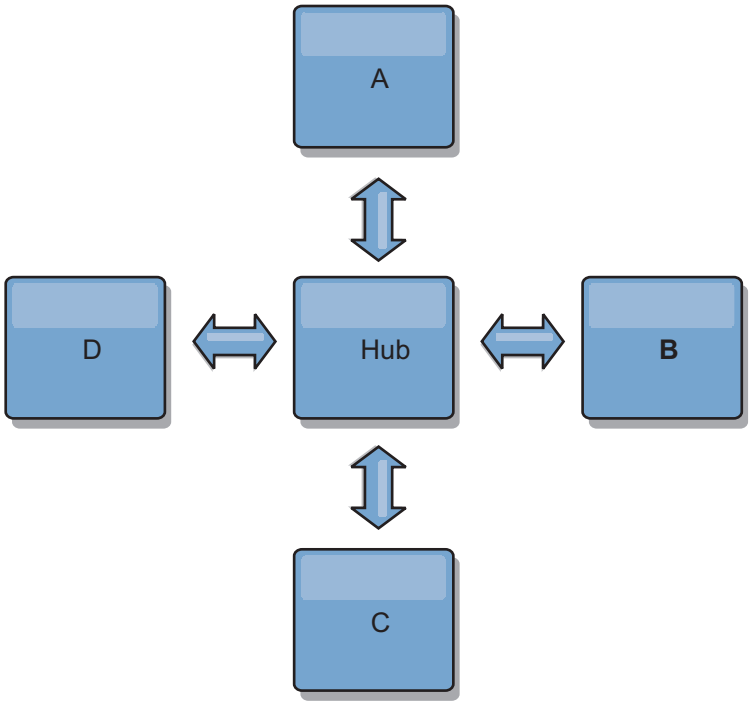
  – O Domínio A está ativo e o B está inativo
  – Os Domínios A e B estão ativos e C está inativo
  – O link entre A e B está inativo
  – O link entre B e C está inativo
  – O link entre C e D está inativo

  Em contraste, com uma topologia em anel, cada domínio de serviço de catálogo pode receber mudanças a partir de qualquer direção.



Por exemplo, se um determinado serviço de catálogo em sua topologia em anel estiver inativo, os dois domínios adjacentes ainda poderão obter as mudanças diretamente entre si.

Todas as mudanças são propagadas por meio do hub. Assim, em oposição às topologia de linha e em anel, o design hub-and-spoke será suscetível à interrupção se o hub falhar.



Um domínio de serviço de catálogo único é resiliente a uma determinada perda de quantidade de serviço. No entanto, falhas maiores, como interrupções ou perda de links em uma rede maior entre os datacenters físicos pode interromper qualquer um dos domínios de serviço de catálogo.

- **Vínculo e desempenho**

  O número de links definido em um domínio de serviço de catálogo afeta o desempenho. Mais links usam mais recursos e o desempenho de replicação pode diminuir como resultado. A capacidade de recuperar mudanças para um domínio A por meio de outros domínios isenta efetivamente o domínio A da replicação de suas transações em todo lugar. O carregamento de distribuição de

mudança em um domínio é limitado pelo número de links que ele usa, e não pela quantia de domínios presentes na topologia. Esta propriedade carregamento fornece escalabilidade, de modo que os domínios na topologia possam compartilhar a carga de distribuição de mudança.

Um domínio de serviço de catálogo pode recuperar as mudanças indiretamente por meio de outros domínios de serviço de catálogo. Considere uma topologia em linha com cinco domínios de serviço de catálogo.

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

- A pega mudanças de B, C, D e E por meio de B
- B pega mudanças de A e C diretamente e mudanças de D e E por meio de C
- C pega mudanças de B e D diretamente e mudanças de A por meio de B e E por meio de D
- D pega mudanças de C e E diretamente e mudanças de A e B por meio de C
- E pega mudanças de D diretamente e mudanças de A, B e C por meio de D

O carregamento de distribuição nos domínios de serviço de catálogo A e E é o mais baixo, porque cada um deles possui um link apenas para um domínio de serviço de catálogo único. Cada um dos domínios B, C e D possui um link para dois domínios. Assim, o carregamento de distribuição nos domínios B, C e D é o dobro do carregamento nos domínios A e E. A carga de trabalho depende do número de links em cada domínio, e não do número geral de domínios na topologia. Assim, a distribuição dos carregamentos descrita permaneceria constante, mesmo se a linha contivesse 1.000 domínios.

Considerações de Desempenho de Replicação Multimestre

Leve em consideração as seguintes limitações quando usar topologias de replicação multimestre:

- **Alterar ajuste de distribuição**, conforme discutido na seção anterior.
- **Desempenho do link de replicação** O WebSphere eXtreme Scale cria um único soquete TCP/IP entre qualquer par de JVMs. Todo o tráfego entre as JVMs ocorre por meio deste soquete único, incluindo o tráfego de replicação multimestre. Os domínios do serviço de catálogo são hospedados em pelo menos *n* JVMs de contêiner, fornecendo pelo menos *n* links TCP para os domínios de serviço de catálogo equivalentes. Assim, os domínios de serviço de catálogo com números maiores de contêineres têm níveis maiores de desempenho de replicação. Mais contêineres requerem mais recursos de processador e de rede.
- **Ajuste da janela deslizante TCP e o RFC 1323** O suporte do RFC 1323 em ambas as extremidades de um link geram mais dados para um roundtrip. Este suporte resulta em maior rendimento, expandindo a capacidade da janela em um fator de aproximadamente 16.000.

  A rechamada desses soquetes TCP usa um mecanismo de janela deslizante para controlar o fluxo de dados em massa. Este mecanismo geralmente limita o soquete para 64 KB para um intervalo de roundtrip. Se o intervalo de roundtrip for de 100 ms, a largura de banda será limitada a 640 KB/segundo sem ajuste adicional. Usar totalmente a largura de banda disponível em um link pode exigir um ajuste específico para um sistema operacional. A maioria dos sistemas operacionais inclui parâmetros de ajuste, inclusive opções RFC 1323, para aprimorar o rendimento por meio dos links de alta latência.

  Vários fatores podem afetar o desempenho de replicação:

  - A velocidade com que o eXtreme Scale recupera as mudanças.

  - A velocidade com que o eXtreme Scale pode atender às solicitações de replicação de recuperação.
  - A capacidade da janela de deslizamento.
  - Com o ajuste do buffer de rede em ambos os lados de um link, o eXtreme Scale recupera as mudanças por meio do soquete de modo eficiente.
- **Serialização de Objeto** Todos os dados devem ser serializáveis. Se um domínio de serviço de catálogo não estiver usando `COPY_TO_BYTES`, o domínio de serviço de catálogo deverá usar a serialização Java ou ObjectTransformers para otimizar o desempenho da serialização.
- **Compactação** O WebSphere eXtreme Scale compacta todos os dados enviados entre os domínios de serviço de catálogo por padrão. Desativar a compactação não está disponível atualmente.
- **Ajuste de memória** O uso de memória para uma topologia de replicação multimestre é altamente independente do número de domínios de serviço de catálogo na topologia.

  A replicação multimaster inclui uma quantidade fixa de processamento por entrada de Mapa para manipular a versão. Cada contêiner também controla uma quantia fixa de dados em cada domínio de serviço de catálogo na topologia. Uma topologia com dois domínios de serviço de catálogo usa aproximadamente a mesma memória que uma topologia com 50 domínios de serviço de catálogo. O WebSphere eXtreme Scale não usa logs de reprodução ou filas semelhantes em sua implementação. Assim, não há nenhuma estrutura de recuperação pronta caso um link de replicação esteja indisponível por um período prolongado e reinícios posteriores.

Interoperabilidade com Outros Produtos WebSphere

Você pode integrar o WebSphere eXtreme Scale com outros produtos servidores como WebSphere Application Server e WebSphere Application Server Community Edition.

WebSphere Application Server

É possível integrar o WebSphere Application Server em vários aspectos de sua configuração do WebSphere eXtreme Scale. É possível implementar os aplicativos da grade de dados e usar o WebSphere Application Server para hospedar os servidores de contêiner e de catálogos. Também é possível usar a segurança do WebSphere Application Server no ambiente do WebSphere eXtreme Scale .

Portal WebSphere

É possível persistir as sessões HTTP a partir do WebSphere Portal em uma grade de dados no WebSphere eXtreme Scale.

WebSphere Application Server Community Edition

O WebSphere Application Server Community Edition pode compartilhar estado de sessão, mas não de uma maneira eficiente e escalável. O WebSphere eXtreme Scale fornece uma camada de persistência distribuída e de alto desempenho, que pode ser utilizada para replicar o estado, mas não se integra prontamente a qualquer servidor de aplicativos fora do WebSphere Application Server. Estes dois produtos podem ser integrados para fornecer uma solução de gerenciamento de sessões escalável.

WebSphere Real Time

Com suporte para o WebSphere Real Time, a oferta Java real líder de mercado, o WebSphere eXtreme Scale permite que os aplicativos do Extreme Transaction Processing (XTP) tenham tempos de resposta mais consistentes e previsíveis.

Capítulo 3. Cenários

Um cenário apresenta exemplos para ajudar o usuário a entender um conceito ou realizar uma tarefa. O cenário usa as informações do mundo real para construir uma imagem completa.

Usando um Ambiente OSGi para Desenvolver e Executar Plug-ins do eXtreme Scale

Use estes cenários para concluir tarefas comuns em um ambiente OSGi. Por exemplo, a estrutura do OSGi é ideal para iniciar os servidores e clientes em um contêiner OSGi, que permite incluir e atualizar dinamicamente plug-ins do WebSphere eXtreme Scale para o ambiente de tempo de execução.

Os cenários a seguir são sobre construir e executar dinamicamente os plug-ins, que permite instalar, iniciar, parar, modificar e desinstalar os plug-ins de modo dinâmico. Você também pode concluir um outro cenário provável, que permite usar a estrutura do OSGi sem os recursos dinâmicos. Também é possível incluir os aplicativos em pacotes configuráveis, que são definidos e comunicados por meio de serviços. Esses pacotes configuráveis baseados em serviço oferecem vários benefícios, incluindo capacidades de desenvolvimento e de implementação mais eficientes.

Objetivos do Cenário

Depois de concluir as lições deste módulo, você saberá como concluir as tarefas:

- Construa plug-ins dinâmicos do eXtreme Scale para uso em um ambiente OSGi.
- Execute os contêineres do eXtreme Scale em um ambiente OSGi sem os recursos dinâmicos.

Pré-requisitos

Leia o tópico “Visão Geral da Estrutura do OSGi” na página 24 para aprender mais sobre o suporte de OSGi e os benefícios que ele pode oferecer.

Visão Geral da Estrutura do OSGi

O OSGi define um sistema módulo dinâmico para Java. A plataforma de serviço OSGi possui uma arquitetura em camadas e é projetada para ser executada em vários perfis padrão Java. É possível iniciar servidores e clientes do WebSphere eXtreme Scale em um contêiner OSGi.

Benefícios de Executar Aplicativos no Contêiner OSGi

O suporte do WebSphere eXtreme Scale OSGi permite implementar o produto na estrutura do Eclipse Equinox OSGi. Anteriormente, se você desejava atualizar os plug-ins usados pelo eXtreme Scale, era necessário reiniciar a Java Virtual Machine (JVM) para aplicar as novas versões dos plug-ins. Com a capacidade de atualização dinâmica que a estrutura OSGi fornece, agora é possível atualizar as classes de plug-in sem reiniciar a JVM. Esses plug-ins são exportados pelos pacotes configuráveis do usuário como serviços. O WebSphere eXtreme Scale acessa o serviço ou serviços consultando-os no registro OSGi.

Os contêineres do eXtreme Scale podem ser configurados para iniciar mais fácil e dinamicamente usando o serviço administrativo de configuração do OSGi ou com o OSGi Blueprint. Se desejar implementar uma nova grade de dados com sua estratégia de posicionamento, será possível fazer isso criando uma configuração de OSGi ou implementando um pacote configurável com arquivos XML do descritor eXtreme Scale. Com o suporte do OSGi, os pacotes configuráveis de aplicativo que contém dados de configuração do eXtreme Scale podem ser instalados, iniciados, interrompidos, atualizados e desinstalados sem reiniciar o sistema inteiro. Com esta capacidade, é possível fazer upgrade do aplicativo sem interromper a grade de dados.

Beans de plug-in e serviços podem ser configurados com escopos de shard customizados, permitindo opções de integração sofisticadas com outros serviços em execução na grade de dados. Cada plug-in pode usar classificações do OSGi Blueprint para verificar se cada instância do plug-in ativada está na versão correta. Um bean gerenciado por OSGi (MBean) e o utilitário **xscmd** são fornecidos, o que permite que você consulte os serviços OSGi de plug-in do eXtreme Scale e suas classificações.

Este recurso permite que os administradores reconheçam rapidamente erros de configuração e administração em potencial e atualize as classificações de serviço de plug-in em uso pelo eXtreme Scale.

Pacotes Configuráveis OSGi

Para interagir com, e implementar, plug-ins na estrutura do OSGi, você deve usar os *pacotes configuráveis*. Na plataforma de serviço OSGi, um pacote configurável é um arquivo Java archive (JAR) que contém código Java, recursos e um manifesto que descrevem o pacote configurável e suas dependências. O pacote configurável é a unidade de implementação para um aplicativo. O produto eXtreme Scale suporta os seguintes tipos de pacotes configuráveis:

Pacote configurável do servidor
O pacote configurável do servidor é o arquivo `objectgrid.jar` e é instalado com a instalação do servidor independente do eXtreme Scale e é necessário para executar servidores do eXtreme Scale e também pode ser usado para executar clientes do eXtreme Scale ou caches na memória locais. O ID do pacote configurável para o arquivo `objectgrid.jar` é com.ibm.websphere.xs.server_`<version>`, em que a versão está no formato: `<Version>.<Release>.<Modification>`. Por exemplo, o pacote configurável do servidor para o eXtreme Scale versão 7.1.1 é com.ibm.websphere.xs.server_7.1.1.

Pacote configurável do cliente
O pacote configurável do cliente é o arquivo `ogclient.jar` e é instalado com instalações independentes e do cliente do eXtreme Scale e é usado para executar os clientes do eXtreme Scale ou caches na memória locais. O ID do pacote configurável para o arquivo `ogclient.jar` é com.ibm.websphere.xs.client_`version`, em que a versão está no formato: `<Version>.<Release>.<Modification>`. Por exemplo, o pacote configurável do cliente para o eXtreme Scale versão 7.1.1 é com.ibm.websphere.xs.client_7.1.1.

Limitações

Não é possível reiniciar o pacote configurável do eXtreme Scale porque você não pode reiniciar o object request broker (ORB). Para reiniciar o servidor do eXtreme

Scale, você deve reiniciar a estrutura do OSGi.

Instalando a Estrutura do Eclipse Equinox OSGi com o Eclipse Gemini para Clientes e Servidores

Se desejar implementar o WebSphere eXtreme Scale na estrutura do OSGi, você deverá configurar o Ambiente do Eclipse Equinox.

Sobre Esta Tarefa

A tarefa requer fazer o download e instalar a estrutura do blueprint, que permite configurar posteriormente o JavaBeans e expô-los como serviços. O uso de serviços é importante porque é possível expor plug-ins como serviços OSGi, de modo que eles possam ser usados pelo ambiente de tempo de execução do eXtreme Scale. O produto suporta dois contêineres blueprint dentro da estrutura OSGi principal do Eclipse Equinox: Eclipse Gemini e Aries Apache. Use este procedimento para configurar o contêiner do Eclipse Gemini.

Procedimento

1. Faça download do Eclipse Equinox SDK Versão 3.6.1 ou posterior a partir do website do Eclipse. Crie um diretório para a estrutura do Equinox, por exemplo: `/opt/equinox`. Essas instruções referenciam esse diretório como `equinox_root`. Extraia o arquivo compactado no diretório `equinox_root`.
2. Faça download do arquivo compactado gemini-blueprint incubation 1.0.0 a partir do Website Eclipse. Extraia o conteúdo do arquivo em um diretório temporário e copie os seguintes arquivos extraídos para o diretório `equinox_root/plugins`:

   ```
   dist/gemini-blueprint-core-1.0.0.jar
   dist/gemini-blueprint-extender-1.0.0.jar
   dist/gemini-blueprint-io-1.0.0.jar
   ```

3. Faça download do Spring Framework Versão 3.0.5 a partir da página da web SpringSource a seguir: http://www.springsource.com/download/community. Extraia-o em um diretório temporário e copie os seguintes arquivos extraídos para o diretório `equinox_root/plugins`:

   ```
   org.springframework.aop-3.0.5.RELEASE.jar
   org.springframework.asm-3.0.5.RELEASE.jar
   org.springframework.beans-3.0.5.RELEASE.jar
   org.springframework.context-3.0.5.RELEASE.jar
   org.springframework.core-3.0.5.RELEASE.jar
   org.springframework.expression-3.0.5.RELEASE.jar
   ```

4. Faça download do arquivo Java archive (JAR) AOP Alliance a partir da página da web do SpringSource. Copie o arquivo com.springsource.org.aopalliance-1.0.0.jar para o diretório `equinox_root/plugins` .
5. Faça download do arquivo JAR Apache Commons Logging 1.1.1 a partir da página da web do SpringSource. Copie o arquivo `com.springsource.org.apache.commons.logging-1.1.1.jar` para o diretório `equinox_root/plugins`.
6. Faça download do cliente de linha de comandos Luminis OSGi Configuration Admin. Use esse pacote configurável para gerenciar as configurações administrativas do OSGi. É possível fazer o download do arquivo JAR a partir da seguinte página da web: https://opensource.luminis.net/wiki/display/SITE/OSGi+Configuration+Admin+command+line+client. Copie o arquivo `net.luminis.cmc-0.2.5.jar` para o diretório `equinox_root/plugins`.

7. Faça download do pacote configurável do arquivo de instalação Apache Felix Versão 3.0.2 a partir da seguinte página da web: http://felix.apache.org/site/index.html. Copie o arquivo `org.apache.felix.fileinstall-3.0.2.jar` para o diretório `equinox_root/plugins`.
8. Crie um diretório de configuração dentro do diretório `equinox_root/plugins`, por exemplo:

   ```
   mkdir equinox_root/plugins/configuration
   ```

9. Crie o arquivo `config.ini` a seguir no diretório `equinox_root/plugins/configuration`, substituindo *equinox_root* com o caminho absoluto para seu diretório `equinox_root` e removendo todos os espaçoss à direita após a barra invertida em cada linha. Você deve incluir uma linha em branco no final do arquivo; por exemplo:

   ```
   osgi.noShutdown=true
   osgi.java.profile.bootdelegation=none
   org.osgi.framework.bootdelegation=none
   eclipse.ignoreApp=true
   osgi.bundles=\
   org.eclipse.osgi.services_3.2.100.v20100503.jar@1:start, \
   org.eclipse.osgi.util_3.2.100.v20100503.jar@1:start, \
   org.eclipse.equinox.cm_1.0.200.v20100520.jar@1:start, \
   com.springsource.org.apache.commons.logging-1.1.1.jar@1:start, \
   com.springsource.org.aopalliance-1.0.0.jar@1:start, \
   org.springframework.aop-3.0.5.RELEASE.jar@1:start, \
   org.springframework.asm-3.0.5.RELEASE.jar@1:start, \
   org.springframework.beans-3.0.5.RELEASE.jar@1:start, \
   org.springframework.context-3.0.5.RELEASE.jar@1:start, \
   org.springframework.core-3.0.5.RELEASE.jar@1:start, \
   org.springframework.expression-3.0.5.RELEASE.jar@1:start, \
   org.apache.felix.fileinstall-3.0.2.jar@1:start, \
   net.luminis.cmc-0.2.5.jar@1:start, \
   gemini-blueprint-core-1.0.0.jar@1:start, \
   gemini-blueprint-extender-1.0.0.jar@1:start, \
   gemini-blueprint-io-1.0.0.jar@1:start
   ```

   Se já tiver configurado o ambiente, será possível limpar o repositório de plug-in do Equinox ao remover o seguinte diretório: `equinox_root\plugins\configuration\org.eclipse.osgi`.
10. Execute os seguintes comandos para iniciar o console do equinox.

    Se você estiver executando uma versão diferente do Equinox, o nome do seu arquivo JAR será diferente do nome no exemplo a seguir:

    ```
    java -jar plugins\org.eclipse.osgi_3.6.1.R36x_v20100806.jar -console
    ```

Instalando Pacotes Configuráveis do eXtreme Scale

O WebSphere eXtreme Scale inclui pacotes configuráveis que podem ser instalados em uma estrutura do Eclipse Equinox OSGi. Esses pacotes configuráveis são necessários para iniciar servidores do eXtreme Scale ou usar clientes do eXtreme Scale no OSGi.

Antes de Iniciar

Esta tarefa assume que os produtos a seguir foram instalados:

- Estrutura do Eclipse Equinox OSGi
- Cliente ou servidor independente do eXtreme Scale

Sobre Esta Tarefa

O eXtreme Scale inclui dois pacotes configuráveis. Apenas um dos pacotes configuráveis a seguir é requerido em uma estrutura do OSGi:

objectgrid.jar
: O pacote configurável do servidor é o arquivo `objectgrid.jar` e é instalado com a instalação do servidor independente do eXtreme Scale e é requerido para executar servidores do eXtreme Scale e também pode ser usado para executar clientes do eXtreme Scale ou caches na memória

locais. O ID do pacote configurável para o arquivo objectgrid.jar é com.ibm.websphere.xs.server_<version>, em que a versão está no formato: <Version>.<Release>.<Modification>. Por exemplo, o pacote configurável do servidor para o eXtreme Scale versão 7.1.1 é com.ibm.websphere.xs.server_7.1.1.

ogclient.jar
O pacote configurável do ogclient.jar é instalado com as instalações independentes e de cliente do eXtreme Scale e é usado para executar clientes do eXtreme Scale ou caches na memória locais. O ID do pacote configurável para o arquivo ogclient.jar é com.ibm.websphere.xs.client_<version>, em que a versão está no formato: <Version>_<Release>_<Modification. Por exemplo, o pacote configurável do cliente para o eXtreme Scale Versão 7.1.1 é com.ibm.websphere.xs.client_7.1.1.

Para obter mais informações sobre como desenvolver plug-ins do eXtreme Scale, consulte o tópico APIs e Plug-ins do Sistema.

Procedimento

Para instalar o pacote configurável do cliente ou servidor do eXtreme Scale na estrutura do Eclipse Equinox OSGi usando o console do OSGi:

1. Inicie a estrutura do Eclipse Equinox com o console ativado; por exemplo:
   ```
   java_home/bin/java -jar <equinox_root>/plugins/
   org.eclipse.osgi_3.6.1.R36x_v20100806.jar -console
   ```
2. Instale o pacote configurável do cliente ou servidor do eXtreme Scale no console do Equinox:
   ```
   osgi> install file:///<path to bundle>
   ```
3. O Equinox exibe o ID do pacote configurável para o pacote configurável recém-instalado:
   ```
   Bundle id is 25
   ```
4. Inicie o pacote configurável no console do Equinox, em que <id> é o ID do pacote configurável designado quando o pacote configurável foi instalado:
   ```
   osgi> start <id>
   ```
5. Recupere o status de serviço no console do Equinox para verificar se o pacote configurável foi iniciado; por exemplo:
   ```
   osgi> ss
   ```
   Quando o pacote configurável foi iniciado com êxito, o pacote configurável exibe o estado ACTIVE; por exemplo:
   ```
   25      ACTIVE      com.ibm.websphere.xs.server_7.1.1
   ```

Instale o pacote configurável do cliente ou servidor do eXtreme Scale na estrutura do Eclipse Equonix OSGi usando o arquivo config.ini:

6. Copie o pacote configurável do cliente ou servidor do eXtreme Scale (objectgrid.jar ou ogclient.jar) do <wxs_install_root>/ObjectGrid/lib para o diretório de plug-ins do Eclipse Equinox; por exemplo: <equinox_root>/plugins
7. Edite o arquivo de configuração config.ini do Eclipse Equinox e inclua o pacote configurável na propriedade osgi.bundles; por exemplo:

```
osgi.bundles=\
org.eclipse.osgi.services_3.2.100.v20100503.jar@1:start, \
org.eclipse.osgi.util_3.2.100.v20100503.jar@1:start, \
org.eclipse.equinox.cm_1.0.200.v20100520.jar@1:start, \
objectgrid.jar@1:start
```

Importante: Verifique se existe uma linha em branco após o nome do último pacote configurável. Cada pacote configurável é separado por uma vírgula.

8. Inicie a estrutura do Eclipse Equinox com o console ativado; por exemplo:

   ```
   java_home/bin/java -jar <equinox_root>/plugins/
   org.eclipse.osgi_3.6.1.R36x_v20100806.jar -console
   ```

9. Recupere o status do serviço no console do Equinox para verificar se o pacote configurável foi iniciado:

   ```
   osgi> ss
   ```

   Quando o pacote configurável tiver iniciado com sucesso, o pacote configurável exibirá o estado ACTIVE; por exemplo:

   ```
   25      ACTIVE      com.ibm.websphere.xs.server_7.1.1
   ```

Resultados

O pacote configurável do servidor ou cliente do eXtreme Scale é instalado e iniciado em sua estrutura do Eclipse Equinox OSGi.

Construindo e Executando Plug-ins Dinâmicos do eXtreme Scale para Uso em um Ambiente OSGi

Todos os plug-ins do eXtreme Scale podem ser configurados para um ambiente OSGi. O benefício principal dos plug-ins dinâmicos é que o upgrade deles pode ser feito sem encerrar a grade. Isso permite desenvolver um aplicativo sem reiniciar os processos do contêiner de grade.

Sobre Esta Tarefa

O suporte do WebSphere eXtreme Scale OSGi permite implementar o produto em uma estrutura OSGi, como o Eclipse Equinox. Anteriormente, se você desejava atualizar os plug-ins usados pelo eXtreme Scale, era necessário reiniciar a Java Virtual Machine (JVM) para aplicar as novas versões dos plug-ins. Com o suporte de plug-in dinâmico fornecido pelo eXtreme Scale e a possibilidade de atualizar pacotes configuráveis que a estrutura do OSGi fornece, agora é possível atualizar as classes de plug-in sem reiniciar a JVM. Esses plug-ins são exportados pelo *pacotes configuráveis* como serviços. O WebSphere eXtreme Scale acessa o serviço ao consultar o registro do OSGi. Na plataforma de serviço OSGi, um pacote configurável é um arquivo Java archive (JAR) que contém código Java, recursos e um manifesto que descrevem o pacote configurável e suas dependências. O pacote configurável é a unidade de implementação para um aplicativo.

Procedimento

1. Construa plug-ins dinâmicos do eXtreme Scale.
2. Configure plug-ins do eXtreme Scale with OSGi Blueprint.
3. Instale e inicie plug-ins ativados por OSGi.

Construindo Plug-ins Dinâmicos do eXtreme Scale

O WebSphere eXtreme Scale inclui plug-ins ObjectGrid e BackingMap. Estes plug-ins são implementados em Java e são configurados usando o arquivo XML do

descritor do ObjectGrid. Para criar um plug-in dinâmico que pode ser dinamicamente atualizado, eles precisam estar cientes dos eventos de ciclo de vida de ObjectGrid e BackingMap porque eles podem precisar concluir algumas ações durante uma atualização. Aprimorar um pacote configurável de plug-in com métodos de retorno de chamada, listeners de eventos, ou ambos, do ciclo de vida permite que o plug-in conclua essas ações em momentos apropriados.

Antes de Iniciar

Este tópico supõe que você construiu o plug-in apropriado. Para obter informações adicionais sobre como desenvolver plug-ins do eXtreme Scale, consulte o tópico APIs e Plug-ins do Sistema.

Sobre Esta Tarefa

Todos os plug-ins do eXtreme Scale se aplicam a uma instância de BackingMap ou de ObjectGrid. Muitos plug-ins também interagem com outros plug-ins. Por exemplo, um plug-in Loader e TransactionCallback trabalham juntos para interagir corretamente com uma transação do banco de dados e as várias chamadas de banco de dados JDBC. Alguns plug-ins também pode precisar armazenar em cache dados de configuração a partir de outros plug-ins para melhorar o desempenho.

Os plug-ins BackingMapLifecycleListener e ObjectGridLifecycleListener fornecem operações de ciclo de vida para as respectivas instâncias de BackingMap e ObjectGrid. Este processo permite que plug-ins sejam notificados quando o BackingMap ou ObjectGrid pai e seus respectivos plug-ins podem ser alterados. Os plug-ins BackingMap implementam a interface de BackingMapLifecyleListener e os plug-ins ObjectGrid implementam a interface de ObjectGridLifecycleListener. Estes plug-ins são chamados automaticamente quando o ciclo de vida do BackingMap ou ObjectGrid pai é alterado. Para obter mais informações sobre os plug-ins de ciclo de vida, consulte o tópico Gerenciando Ciclos de Vida de Plug-in.

É possível aprimorar os pacotes configuráveis usando os métodos ou os listeners de evento do ciclo de vida nas seguintes tarefas comuns:

- Iniciar e parar recursos, como encadeamentos ou assinantes de sistema de mensagens.
- Especificar que uma notificação ocorra quando os plug-ins equivalentes forem atualizados, permitindo o acesso direto ao plug-in e a detecção de quaisquer mudanças.

Sempre que outro plug-in for acessado diretamente, acesse esse plug-in por meio do contêiner OSGi para assegurar que todas as partes do sistema referenciem o plug-in correto. Se, por exemplo, algum componente no aplicativo referenciar diretamente, ou armazenar em cache, uma instância de um plug-in, ele manterá sua referência para essa versão do plug-in, mesmo depois que o plug-in tiver sido atualizado dinamicamente. Esse comportamento pode causar problemas relacionados ao aplicativo, bem como fugas de memória. Portanto, grave o código que depende dos plug-ins dinâmicos que obtêm sua referência usando semânticas getService() do OSGi. Se o aplicativo precisar armazenar em cache um ou mais plug-ins, ele atenderá eventos de ciclo de vida usando interfaces ObjectGridLifecycleListener e BackingMapLifecycleListener. O aplicativo também deve poder atualizar seu cache quando necessário, de modo thread safe.

Todos os plug-ins do eXtreme Scale usados com o OSGi também devem implementar as respectivas interfaces BackingMapPlugin ou ObjectGridPlugin.

Novos plug-ins, tal como a interface MapSerializerPlugin impingem essa prática. Essas interfaces fornecem ao ambiente de tempo de execução do eXtreme Scale e ao OSGi uma interface consistente para injeção de estado no plug-in e controle de seu ciclo de vida.

Ao usar esta tarefa para especificar que uma notificação ocorre quando os plug-ins equivalentes são atualizados, é possível criar um factory de listener que produz uma instância do listener.

Procedimento

- Atualize a classe de plug-in ObjectGrid para implementar a interface ObjectGridPlugin. Esta interface inclui métodos que permitem que o eXtreme Scale inicialize, configure a instância do ObjectGrid e destrua o plug-in. Consulte o exemplo de código a seguir:

```
package com.mycompany;
import com.ibm.websphere.objectgrid.plugins.ObjectGridPlugin;
...

public class MyTranCallback implements TransactionCallback, ObjectGridPlugin {

    private ObjectGrid og = null;

    private enum State {
        NEW, INITIALIZED, DESTROYED
    }

    private State state = State.NEW;

    public void setObjectGrid(ObjectGrid grid) {
        this.og = grid;
    }

    public ObjectGrid getObjectGrid() {
        return this.og;
    }
    void initialize() {
        // Handle any plug-in initialization here.  This is called by
        // eXtreme Scale, and not the OSGi bean manager.
        state = State.INITIALIZED;
    }
    boolean isInitialized() {
        return state == State.INITIALIZED;
    }

    public void destroy() {
        // Destroy the plug-in and release any resources.  This
        // can be callsed by the OSGi Bean Manager or by eXtreme Scale.
        state = State.DESTROYED;
    }

    public boolean isDestroyed() {
        return state == State.DESTROYED;
    }
}
```

- Atualize a classe de plug-in do ObjectGrid para implementar a interface ObjectGridLifecycleListener. Consulte o exemplo de código a seguir:

```
package com.mycompany;
import com.ibm.websphere.objectgrid.plugins.ObjectGridLifecycleListener;
      import com.ibm.websphere.objectgrid.plugins.ObjectGridLifecycleListener.LifecycleEvent;
      ...

      public class MyTranCallback implements TransactionCallback, ObjectGridPlugin, ObjectGridLifecycleListener{
          public void objectGridStateChanged(LifecycleEvent event) {
              switch(event.getState()) {
              case NEW:
              case DESTROYED:
              case DESTROYING:
              case INITIALIZING:
                  break;
              case INITIALIZED:
                  // Lookup a Loader or MapSerializerPlugin using
                  // OSGi or directly from the ObjectGrid instance.
                  lookupOtherPlugins()
                  break;
              case STARTING:
              case PRELOAD:
                  break;
              case ONLINE:
                  if (event.isWritable()) {
                      startupProcessingForPrimary();
                  } else {
```

```
                    startupProcessingForReplica();
                }
                break;
            case QUIESCE:
                if (event.isWritable()) {
                    quiesceProcessingForPrimary();
                } else {
                    quiesceProcessingForReplica();
                }
                break;
            case OFFLINE:
                shutdownShardComponents();
                break;
            }
        }
        ...
    }
```

- Atualize um plug-in do BackingMap. Atualize a classe de plug-in do BackingMap para implementar a interface de plug-in do BackingMap. Esta interface inclui métodos que permitem que o eXtreme Scale inicialize, configure a instância do BackingMap e destrua o plug-in. Consulte o exemplo de código a seguir:

```
package com.mycompany;
import com.ibm.websphere.objectgrid.plugins.BackingMapPlugin;
...

public class MyLoader implements Loader, BackingMapPlugin {

    private BackingMap bmap = null;

    private enum State {
        NEW, INITIALIZED, DESTROYED
    }

    private State state = State.NEW;

    public void setBackingMap(BackingMap map) {
        this.bmap = map;
    }

    public BackingMap getBackingMap() {
        return this.bmap;
    }
    void initialize() {
        // Handle any plug-in initialization here.  This is called by
        // eXtreme Scale, and not the OSGi bean manager.
        state = State.INITIALIZED;
    }
    boolean isInitialized() {
        return state == State.INITIALIZED;
    }

    public void destroy() {
        // Destroy the plug-in and release any resources.  This
        // can be callsed by the OSGi Bean Manager or by eXtreme Scale.
        state = State.DESTROYED;
    }

    public boolean isDestroyed() {
        return state == State.DESTROYED;
    }
}
```

- Atualize a classe de plug-in do BackingMap para implementar a interface BackingMapLifecycleListener. Consulte o exemplo de código a seguir:

```
package com.mycompany;

      import com.ibm.websphere.objectgrid.plugins.BackingMapLifecycleListener;
      import com.ibm.websphere.objectgrid.plugins.BackingMapLifecycleListener.LifecycleEvent;
      ...

      public class MyLoader implements Loader, ObjectGridPlugin, ObjectGridLifecycleListener{
          ...
          public void backingMapStateChanged(LifecycleEvent event) {
              switch(event.getState()) {
              case NEW:
              case DESTROYED:
              case DESTROYING:
              case INITIALIZING:
                  break;
              case INITIALIZED:
                  // Lookup a MapSerializerPlugin using
                  // OSGi or directly from the ObjectGrid instance.
                  lookupOtherPlugins()
                  break;
              case STARTING:
              case PRELOAD:
                  break;
```

```
            case ONLINE:
                if (event.isWritable()) {
                    startupProcessingForPrimary();
                } else {
                    startupProcessingForReplica();
                }
                break;
            case QUIESCE:
                if (event.isWritable()) {
                    quiesceProcessingForPrimary();
                } else {
                    quiesceProcessingForReplica();
                }
                break;
            case OFFLINE:
                shutdownShardComponents();
                break;
            }
        }
        ...
    }
```

Resultados

Implementando a interface ObjectGridPlugin ou BackingMapPlugin, o eXtreme Scale pode controlar o ciclo de vida de seu plug-in nos momentos certos.

Implementando a interface ObjectGridLifecycleListener ou BackingMapLifecycleListener, o plug-in é automaticamente registrado como um listener dos eventos de ciclo de vida do ObjectGrid ou do BackingMap associados. O evento INITIALIZING é usado para sinalizar que todos os plug-ins do ObjectGrid e do BackingMap foram inicializados e estão disponíveis para consultar e usar. O evento ONLINE é usado para sinalizar que o ObjectGrid está on-line e pronto para iniciar eventos de processamento.

Configurando os Plug-ins do eXtreme Scale com o OSGi Blueprint

Todos os plug-ins do ObjectGrid e do BackingMap do eXtreme Scale podem ser definidos como beans e serviços OSGi usando o Serviço OSGi Blueprint disponível com o Eclipse Gemini ou o Aries Apache.

Antes de Iniciar

Antes de poder configurar seus plug-ins como serviços OSGi, você deve primeiro empacotar seus plug-ins em um pacote configurável OSGi e entender os princípios fundamentais dos plug-ins necessários. O pacote configurável deve importar os pacotes do cliente ou servidor do WebSphere eXtreme Scale e outros pacotes dependentes requeridos pelos plug-ins ou criar uma dependência do pacote configurável nos pacotes configuráveis do servidor ou cliente doeXtreme Scale. Este tópico descreve como configurar o XML Blueprint para criar beans de plug-in e expô-los como serviços OSGi para o eXtreme Scale usar.

Sobre Esta Tarefa

Beans e serviços são definidos em um arquivo XML do Blueprint e o contêiner do Blueprint descobre, cria e liga os beans juntos e os expõe como serviços. O processo torna os beans disponíveis para outros pacotes configuráveis OSGi, incluindo os pacotes configuráveis de servidor e cliente do eXtreme Scale.

Ao criar serviços de plug-in customizados para uso com o eXtreme Scale, o pacote configurável que deve hospedar os plug-ins deve ser configurado para usar Blueprint. Além disso, um arquivo XML do Blueprint deve ser criado e armazenado dentro do pacote configurável. Leia sobre construção de aplicativos

OSGi com a especificação do Contêiner do Blueprint para obter um entendimento geral da especificação.

Procedimento

1. Crie um arquivo XML do Blueprint. É possível nomear o arquivo de qualquer jeito. No entanto, você deve incluir o namespace do projeto:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">
   ...
   </blueprint>
   ```

2. Crie definições de bean no arquivo XML do Blueprint para cada plug-in do eXtreme Scale.

   Beans são definidos usando o elemento `<bean>` e podem ser ligados a outras referências de bean e podem incluir parâmetros de inicialização.

   Importante: Ao definir um bean, você deve usar o escopo correto. O Blueprint suporta os escopos singleton e de protótipo. O eXtreme Scale também suporta um escopo de shard customizado.

   Defina a maioria dos plug-ins do eXtreme Scale como protótipo ou beans com escopo definido em shard, uma vez que todos os beans devem ser exclusivos para cada shard do ObjectGrid ou instância do BackingMap à qual eles estão associados. Beans com escopo definido em shard podem ser úteis ao usar osbeans em outros contextos para permitir a recuperação da instância correta.

   Para definir um bean com escopo definido em protótipo, use o atributo `scope="prototype"` no bean:

   ```
   <bean id="myPluginBean" class="com.mycompany.MyBean" scope="prototype">
   ...
   </bean>
   ```

   Para definir um bean com escopo definido em shard, você deve incluir o namespace `objectgrid` no esquema XML e usar o atributo `scope="objectgrid:shard"` no bean:

   ```
   <?xml version="1.0" encoding="UTF-8"?>

   <blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
              xmlns:objectgrid="http://www.ibm.com/schema/objectgrid"

       xsi:schemaLocation="http://www.ibm.com/schema/objectgrid
                   http://www.ibm.com/schema/objectgrid/objectgrid.xsd">

       <bean id="myPluginBean" class="com.mycompany.MyBean"
      scope="objectgrid:shard">
       ...
       </bean>

   ...
   ```

3. Crie definições de bean PluginServiceFactory para cada bean de plug-in. Todos os beans do eXtreme Scale devem ter um bean do PluginServiceFactory definido para que o escopo do bean correto possa ser aplicado. O eXtreme Scale inclui um BlueprintServiceFactory que você pode usar. Ele inclui duas propriedades que devem ser configuradas. Você deve configurar a propriedade blueprintContainer com a referência `blueprintContainer` e a propriedade beanId deve ser configurada com o nome do identificador de bean. Quando o eXtreme Scale consulta o serviço para instanciar os beans apropriados, o servidor procura a instância do componente do bean usando o contêiner Blueprint.

```
bean id="myPluginBeanFactory"
    class="com.ibm.websphere.objectgrid.plugins.osgi.BluePrintServiceFactory">
    <property name="blueprintContainer" ref="blueprintContainer"/>
    <property name="beanId" value="myPluginBean" />
</bean>
```

4. Crie um gerenciador de serviços para cada bean PluginServiceFactory. Cada gerenciador de serviços expõe o bean PluginServiceFactory, usando o elemento `<service>`. O elemento de serviço identifica o nome a ser exposto para o OSGi, a referência para o bean PluginServiceFactory, a interface a ser exposta e a classificação do serviço. O eXtreme Scale usa a classificação do gerenciador de serviços para executar upgrades de serviço quando a grade do eXtreme Scale está ativa. Se a classificação não for especificada, a estrutura do OSGi assumirá uma classificação igual a 0. Leia sobre como atualizar as classificações de serviço para obter mais informações.

   Blueprint inclui várias opções para configurar gerenciadores de serviços. Para definir um gerenciador de serviços simples para um bean PluginServiceFactory, crie um elemento `<service>` para cada bean PluginServiceFactory:

   ```
   <service ref="myPluginBeanFactory"
       interface="com.ibm.websphere.objectgrid.plugins.osgi.PluginServiceFactory"
       ranking="1">
   </service>
   ```

5. Armazene o arquivo XML do Blueprint no pacote configurável de plug-ins. O arquivo XML do Blueprint deve ser armazenado no diretório `OSGI-INF/blueprint` para o contêiner do Blueprint a ser descoberto.

   Para armazenar o arquivo XML do Blueprint em um diretório diferente, você deve especificar o seguinte cabeçalho de manifesto Bundle-Blueprint:

   ```
   Bundle-Blueprint: OSGI-INF/blueprint.xml
   ```

Resultados

Os plug-ins do eXtreme Scale agora são configurados para serem expostos em um contêiner OSGi Blueprint, além disso, o arquivo XML do descritor do ObjectGrid é configurado para referenciar os plug-ins usando o serviço OSGi Blueprint.

Instalando e Iniciando Plug-ins Ativados pelo OSGi

Nesta tarefa, você instala o pacote configurável de plug-in dinâmico na estrutura do OSGi. Em seguida, inicia o plug-in.

Antes de Iniciar

Este tópico assume que as seguintes tarefas foram concluídas:

- O pacote configurável do servidor ou cliente do eXtreme Scale foi instalado na estrutura do Eclipse Equinox OSGi. Consulte “Instalando Pacotes Configuráveis do eXtreme Scale” na página 180.
- Um ou mais plug-ins BackingMap ou ObjectGrid dinâmicos foram implementados. Consulte “Construindo Plug-ins Dinâmicos do eXtreme Scale” na página 182.
- Os plug-ins dinâmicos foram empacotados como serviços OSGi nos pacotes configuráveis OSGi.

Sobre Esta Tarefa

Esta tarefa descreve como instalar o pacote configurável usando o console do Eclipse Equinox. O pacote configurável pode ser instalado usando vários métodos diferentes, incluindo a modificação do arquivo de configuração `config.ini`. Os

produtos que incorporam o Eclipse Equinox incluem métodos alternativos para gerenciar pacotes configuráveis. Para obter mais informações sobre como incluir pacotes configuráveis no arquivo `config.ini` no Eclipse Equinox, consulte as opções de tempo de execução do Eclipse.

O OSGi permite que pacotes configuráveis que possuem serviços duplicados sejam iniciados. O WebSphere eXtreme Scale usa a classificação de serviço mais recente. Ao iniciar diversas estruturas OSGi em uma grade de dados do eXtreme Scale, você deve se certificar de que as classificações de serviço corretas sejam iniciados em cada servidor. Não fazer isso faz com que a grade seja iniciada com uma mistura de diferentes versões.

Para ver quais versões estão em uso pela grade de dados, use o utilitário xscmd para verificar as classificações atuais e disponíveis. Para obter mais informações sobre classificações de serviço disponíveis, consulte Atualizando Serviços OSGi para Plug-ins do eXtreme Scale com **xscmd**.

Procedimento

Instale o pacote configurável de plug-in na estrutura do Eclipse Equinox OSGi usando o console do OSGi.

1. Inicie a estrutura do Eclipse Equinox com o console ativado; por exemplo:
   ```
   <java_home>/bin/java -jar <equinox_root>/plugins/org.eclipse.osgi_3.6.1.R36x_v20100806.jar -console
   ```
2. Instale o pacote configurável de plug-in no console do Equinox.
   ```
   osgi> install file:///<path to bundle>
   ```
   O Equinox exibe o ID do pacote configurável para o pacote configurável recém-instalado:
   ```
   Bundle id is 17
   ```
3. Insira a linha a seguir para iniciar o pacote configurável no console do Equinox, em que `<id>` é o ID do pacote configurável designado quando o pacote configurável foi instalado:
   ```
   osgi>  install <id>
   ```
4. Recupere o status do serviço no console do Equinox para verificar se o pacote configurável foi iniciado:
   ```
   osgi> ss
   ```
   Quando o pacote configurável foi iniciado com êxito, o pacote configurável exibe o estado ACTIVE; por exemplo:
   ```
   17      ACTIVE      com.mycompany.plugin.bundle_VRM
   ```

Instale o pacote configurável de plug-in na estrutura do Eclipse Equonix OSGi usando o arquivo config.ini.

5. Copie o pacote configurável de plug-in no diretório de plug-ins do Eclipse Equinox; por exemplo:
   ```
   <equinox_root>/plugins
   ```
6. Edite o arquivo de configuração `config.ini` do Eclipse Equinox e inclua o pacote configurável na propriedade osgi.bundles; por exemplo:
   ```
   osgi.bundles=\
   org.eclipse.osgi.services_3.2.100.v20100503.jar@1:start, \
   org.eclipse.osgi.util_3.2.100.v20100503.jar@1:start, \
   org.eclipse.equinox.cm_1.0.200.v20100520.jar@1:start, \
   com.mycompany.plugin.bundle_VRM.jar@1:start
   ```

Importante: Verifique se existe uma linha em branco após o nome do último pacote configurável. Cada pacote configurável é separado por uma vírgula.

7. Inicie a estrutura do Eclipse Equinox com o console ativado; por exemplo:

```
<java_home>/bin/java -jar <equinox_root>/plugins/org.eclipse.osgi_3.6.1.R36x_v20100806.jar -console
```

8. Recupere o status de serviço no console do Equinox para verificar se o pacote configurável foi iniciado; por exemplo:

```
osgi> ss
```

Quando o pacote configurável foi iniciado com êxito, o pacote configurável exibe o estado ACTIVE; por exemplo:

```
17      ACTIVE      com.mycompany.plugin.bundle_VRM
```

Resultados

O pacote configurável de plug-in agora está instalado e iniciado. O contêiner ou cliente do eXtreme Scale agora pode ser iniciado. Para obter mais informações sobre como desenvolver plug-ins do eXtreme Scale, consulte o tópico APIs e Plug-ins do Sistema APIs e Plug-ins do Sistema.

Executando os Contêineres do eXtreme Scale com Plug-ins Dinâmicos em um Ambiente do OSGi

Se seu aplicativo estiver hospedado na estrutura do Eclipse Equinox OSGi com o Eclipse Gemini ou o Apache Aries, será possível usar esta tarefa para ajudá-lo a instalar e configurar seu aplicativo WebSphere eXtreme Scale no OSGi.

Antes de Iniciar

Antes de iniciar esta tarefa, certifique-se de concluir as tarefas a seguir:

- Instale a estrutura do Eclipse Equinox OSGi com o Eclipse Gemini
- Construa e execute plug-ins dinâmicos do eXtreme Scale para usar em um ambiente OSGi

Sobre Esta Tarefa

Com plug-ins dinâmicos, é possível atualizar dinamicamente o plug-in enquanto a grade ainda está ativa. Isso permite atualizar um aplicativo sem reiniciar os processos do contêiner de grade. Para obter informações adicionais sobre como desenvolver plug-ins do eXtreme Scale, consulte APIs e Plug-ins do Sistema.

Procedimento

1. Configure plug-ins ativados por OSGi usando o arquivo XML do descritor do ObjectGrid.
2. Inicie os servidores de contêiner do eXtreme Scale usando a estrutura do Eclipse Equinox OSGi.
3. Administre serviços OSGi para plug-ins do eXtreme Scale com o utilitário xscmd.
4. Configure servidores com o OSGi Blueprint.

Configurando Plug-ins Ativados pelo OSGi Usando o Arquivo Descritor XML do ObjectGrid

Nesta tarefa, você inclui serviços OSGi existentes em um arquivo XML descritor de forma que os contêineres do WebSphere eXtreme Scale possam reconhecer e carregar os plug-ins ativados pelo OSGi corretamente.

Antes de Iniciar

Para configurar seus plug-ins, certifique-se de:

- Criar seu pacote e ativar plug-ins dinâmicos para implementação do OSGi.
- Ter os nomes dos serviços OSGi que representam seus plug-ins disponíveis.

Sobre Esta Tarefa

Você criou um serviço OSGi para agrupar seu plug-in. Agora, esses serviços devem ser definidos no arquivo `objectgrid.xml` de modo que os contêineres do eXtreme Scale possam carregar e configurar o plug-in ou plug-ins com êxito.

Procedimento

1. Quaisquer plug-ins específicos da grade, tal como TransactionCallback, devem ser especificados sob o elemento objectGrid. Consulte o exemplo a seguir a partir do arquivo `objectgrid.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="MyGrid" txTimeout="60">
            <bean id="myTranCallback" osgiService="myTranCallbackFactory"/>
            ...
        </objectGrid>
        ...
    </objectGrids>
    ...
/objectGridConfig>
```

   Importante: O valor de atributo `osgiService` deve corresponder ao valor de atributo `ref` que é especificado no arquivo XML `blueprint`, no qual o serviço foi definido para myTranCallback PluginServiceFactory.

2. Quaisquer plug-ins específicos do mapa, como carregadores ou serializadores, por exemplo, devem ser especificados no elemento backingMapPluginCollections e referenciados a partir do elemento backingMap. Consulte o exemplo a seguir a partir do arquivo `objectgrid.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="MyGrid" txTimeout="60">
            <backingMap name="MyMap1" lockStrategy="PESSIMISTIC"
                copyMode="COPY_TO_BYTES" nullValuesSupported="false"
                pluginCollectionRef="myPluginCollectionRef1"/>
            <backingMap name="MyMap2" lockStrategy="PESSIMISTIC"
                copyMode="COPY_TO_BYTES" nullValuesSupported="false"
                pluginCollectionRef="myPluginCollectionRef2"/>
            ...
        </objectGrid>
        ...
    </objectGrids>
    ...
    <backingMapPluginCollections>
        <backingMapPluginCollection id="myPluginCollectionRef1">
            <bean id="MapSerializerPlugin" osgiService="mySerializerFactory"/>
        </backingMapPluginCollection>
        <backingMapPluginCollection id="myPluginCollectionRef2">
            <bean id="MapSerializerPlugin" osgiService="myOtherSerializerFactory"/>
            <bean id="Loader" osgiService="myLoader"/>
        </backingMapPluginCollection>
        ...
    </backingMapPluginCollections>
    ...
</objectGridConfig>
```

Resultados

O arquivo `objectgrid.xml` neste exemplo informa ao eXtreme Scale para criar uma grade denominada `MyGrid` com dois mapas, `MyMap1` e `MyMap2`. O mapa `MyMap1` usa o serializador agrupado pelo serviço OSGi, mySerializerFactory. O mapa `MyMap2` usa um serializador do serviço OSGi, myOtherSerializerFactory, e um carregador a partir do serviço OSGi, myLoader.

Iniciando Servidores do eXtreme Scale Usando a Estrutura do Eclipse Equinox OSGi

Os servidores de contêiner do WebSphere eXtreme Scale podem ser iniciados em uma estrutura do Eclipse Equinox OSGi usando vários métodos.

Antes de Iniciar

Antes de poder iniciar um contêiner do eXtreme Scale, você deve ter concluído as tarefas a seguir:

1. O pacote configurável do servidor do WebSphere eXtreme Scale deve estar instalado no Eclipse Equinox.
2. Seu aplicativo deve ser empacotado como um pacote configurável OSGi.
3. Seus plug-ins do WebSphere eXtreme Scale (se houver) devem ser empacotados como um pacote configurável OSGi. Eles podem ser empacotados no mesmo pacote configurável que seu aplicativo ou como pacotes configuráveis separados.

Sobre Esta Tarefa

Esta tarefa descreve como iniciar um servidor de contêiner do eXtreme Scale em uma estrutura do Eclipse Equinox OSGi. É possível usar qualquer um dos seguintes métodos para iniciar os servidores de contêiner usando a implementação do Eclipse Equinox:

- Serviço do OSGi Blueprint

  É possível incluir toda a configuração e os metadados em um pacote configurável OSGi. Consulte a imagem a seguir para compreender o processo do Eclipse Equinox para este método:

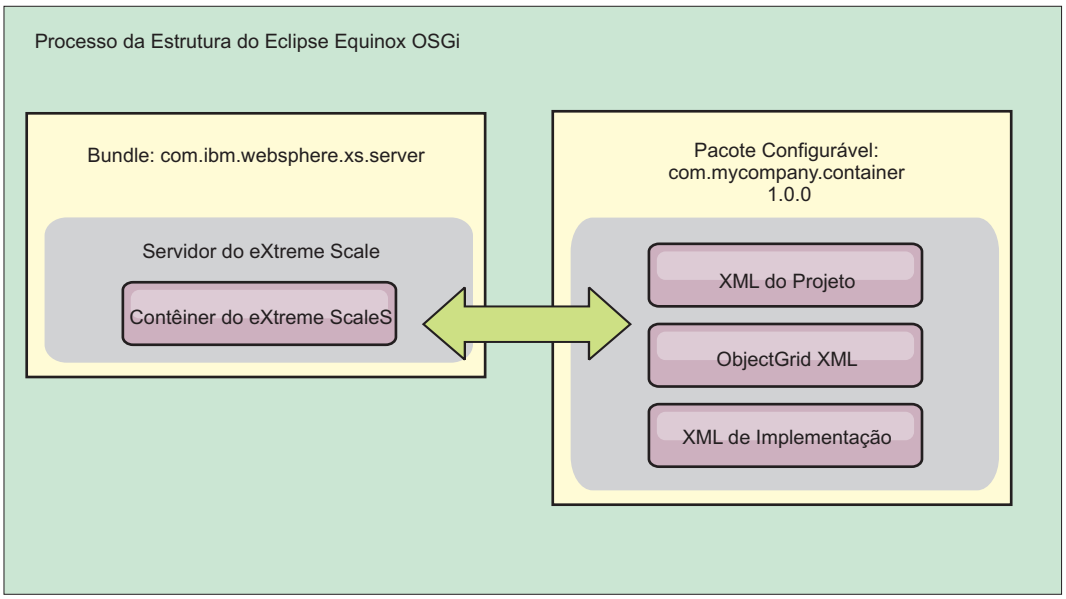


Figura 56. Processo do Eclipse Equinox para Incluir Toda a Configuração e Todos os Metadados em um Pacote Configurável OSGi

- Serviço de Administração de Configuração do OSGi

  É possível especificar a configuração e os metadados fora de um pacote configurável OSGi. Consulte a imagem a seguir para compreender o processo do Eclipse Equinox para este método:

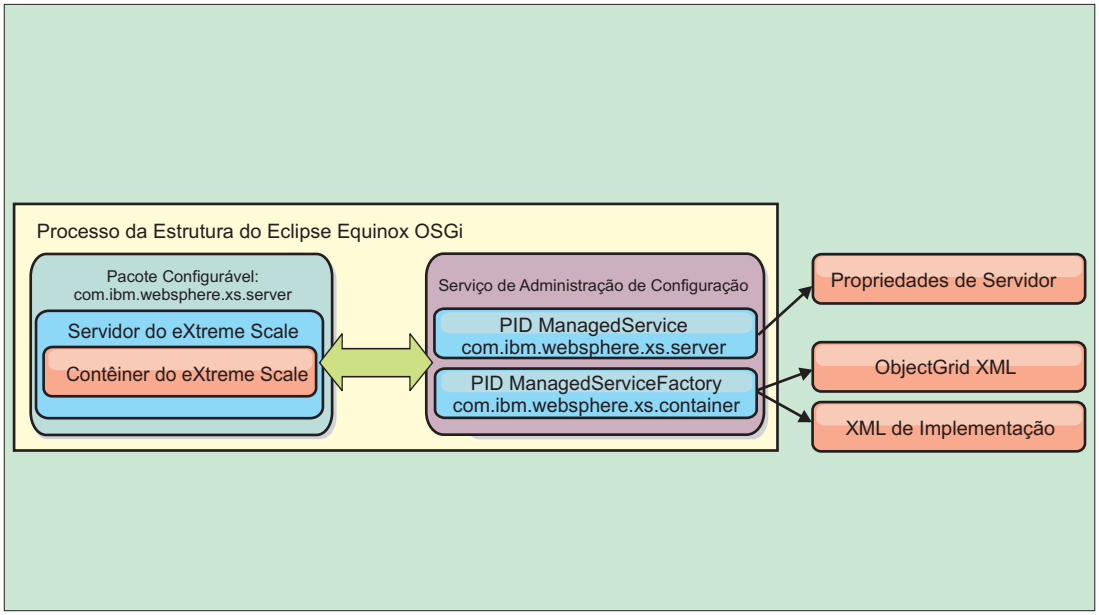


Figura 57. Processo do Eclipse Equinox para Especificar a Configuração e os Metadados Fora de um Pacote Configurável OSGi

- Programaticamente

  Suporta soluções de configuração customizadas.

Em cada caso, um singleton do servidor do eXtreme Scale é configurado e um ou mais contêineres são configurados.

O pacote configurável do servidor do eXtreme Scale, `objectgrid.jar`, inclui todas as bibliotecas necessárias para iniciar e executar um contêiner de grade do eXtreme Scale em uma estrutura OSGi. O ambiente de tempo de execução do servidor se comunica com os plug-ins e objetos de dados fornecidos pelo usuário usando o gerenciador de serviço OSGi.

Importante: Depois de um pacote configurável do servidor do eXtreme Scale ser iniciado e o servidor do eXtreme Scale ser inicializado, ele não poderá ser reiniciado. O processo do Eclipse Equinox deve ser reiniciado para reiniciar um servidor do eXtreme Scale.

É possível usar o suporte do eXtreme Scale para o namespace do Spring para configurar os servidores de contêiner do eXtreme Scale em um arquivo XML do Blueprint. Quando os elemento XML do servidor e do contêiner são incluídos no arquivo XML do Blueprint, o manipulador de namespace do eXtreme Scale inicia automaticamente um servidor de contêiner usando os parâmetros que são definidos no arquivo XML do Blueprint quando o pacote configurável é iniciado. A manipulação para o contêiner quando o pacote configurável é interrompido.

Para configurar servidores de contêiner do eXtreme Scale com o XML do Blueprint, conclua as etapas a seguir:

Procedimento

- Inicie um servidor de contêiner do eXtreme Scale usando o OSGi Blueprint.
  1. Crie um pacote configurável de contêiner.
  2. Instale o pacote configurável de contêiner na estrutura do Eclipse Equinox OSGi. Consulte "Instalando e Iniciando Plug-ins Ativados pelo OSGi" na página 188.
  3. Inicie o pacote configurável do contêiner.
- Inicie uma servidor de contêiner do eXtreme Scale usando a administração de configuração do OSGi.
  1. Configure o servidor e o contêiner usando a administração de configuração.
  2. Quando o pacote configurável do servidor do eXtreme Scale é iniciado ou os identificadores persistentes são criados com config admin, o servidor e o contêiner iniciam automaticamente.
- Inicie um servidor de contêiner do eXtreme Scale usando a API do ServerFactory. Consulte a documentação da API do servidor.
  1. Crie uma classe de ativador do pacote configurável OSGi e use a API do ServerFactory do eXtreme Scale para iniciar um servidor.

Administrando Serviços Ativados pelo OSGi Usando o Utilitário `xscmd`

É possível usar o utilitário **`xscmd`** para concluir as tarefas de administrador, como visualizar serviços e suas classificações que estão sendo usados por cada contêiner e atualizar o ambiente de tempo de execução para utilizar novas versões dos pacotes configuráveis.

Sobre Esta Tarefa

Com a estrutura do Eclipse Equinox OSGi, é possível instalar diversas versões do mesmo pacote configurável e você pode atualizar esses pacotes configuráveis durante o tempo de execução. O WebSphere eXtreme Scale é um ambiente distribuído que executa os servidores de contêiner em muitas instâncias da estrutura do OSGi.

Os administradores são responsáveis por copiar, instalar e iniciar manualmente pacotes configuráveis na estrutura do OSGi. O eXtreme Scale inclui um ServiceTrackerCustomizer OSGi para controlar quaisquer serviços que foram identificados como plug-ins do eXtreme Scale no arquivo XML do descritor do ObjectGrid. Use o utilitário **xscmd** para validar qual versão do plug-in é usado, quais versões estão disponíveis para serem usadas e para executar upgrades do pacote configurável.

O eXtreme Scale usa o número de classificação do serviço para identificar a versão de cada serviço. Quando dois ou mais serviços são carregados com a mesma referência, o eXtreme Scale usa automaticamente o serviço com a classificação mais alta.

Procedimento

- Execute o comando **osgiCurrent** e verifique se cada servidor eXtreme Scale está usando a classificação do serviço de plug-in correta.

  Como o eXtreme Scale escolhe automaticamente a referência de serviço com a classificação mais alta, é possível que a grade de dados possa iniciar com diversas classificações de um serviço de plug-in.

  Se o comando detecta uma incompatibilidade de classificações ou se ele é incapaz de localizar um serviço, um nível de erro diferente de zero é configurado. Se o comando foi concluído com êxito, o nível de erro é configurado como 0.

  O exemplo a seguir mostra a saída do comando **osgiCurrent** quando dois plug-ins estão instalados na mesma grade em quatro servidores. O plug-in loaderPlugin está usando classificação de 1 e txCallbackPlugin está usando classificação 2.

```
OSGi Service Name Current Ranking ObjectGrid Name MapSet Name Server Name
----------------- --------------- --------------- ----------- -----------
loaderPlugin      1               MyGrid          MapSetA     server1
loaderPlugin      1               MyGrid          MapSetA     server2
loaderPlugin      1               MyGrid          MapSetA     server3
loaderPlugin      1               MyGrid          MapSetA     server4
txCallbackPlugin  2               MyGrid          MapSetA     server1
txCallbackPlugin  2               MyGrid          MapSetA     server2
txCallbackPlugin  2               MyGrid          MapSetA     server3
txCallbackPlugin  2               MyGrid          MapSetA     server4
```

  O exemplo a seguir mostra a saída do comando **osgiCurrent** quando server2 foi iniciado com uma classificação mais nova do loaderPlugin:

```
OSGi Service Name Current Ranking ObjectGrid Name MapSet Name Server Name
----------------- --------------- --------------- ----------- -----------
loaderPlugin      1               MyGrid          MapSetA     server1
loaderPlugin      2               MyGrid          MapSetA     server2
loaderPlugin      1               MyGrid          MapSetA     server3
loaderPlugin      1               MyGrid          MapSetA     server4
txCallbackPlugin  2               MyGrid          MapSetA     server1
txCallbackPlugin  2               MyGrid          MapSetA     server2
txCallbackPlugin  2               MyGrid          MapSetA     server3
txCallbackPlugin  2               MyGrid          MapSetA     server4
```

- Execute o comando **osgiAll** para verificar se os serviços de plug-in foram iniciados corretamente em cada servidor de contêiner do eXtreme Scale.

  Quando pacotes configuráveis que contêm serviços que uma configuração do ObjectGrid está referenciando são iniciados, o ambiente de tempo de execução do eXtreme Scale controla automaticamente o plug-in, mas não o usa imediatamente. O comando **osgiAll** mostra quais plug-ins estão disponíveis para cada servidor.

Quando executados sem quaisquer parâmetros, todos os serviços são mostrados para todas as grades e todos os servidores. Filtros adicionais, incluindo o filtro **-serviceName** <service_name>, podem ser especificados para limitar a saída para um único serviço ou um subconjunto da grade de dados.

O exemplo a seguir mostra a saída do comando **osgiAll** quando dois plug-ins são iniciados em dois servidores. O loaderPlugin possui ambas as classificações, 1 e 2, iniciadas e o txCallbackPlugin tem a classificação 1 iniciada. A mensagem de resumo no final da saída confirma que ambos os servidores consultam as mesmas classificações de serviço:

```
Server: server1
   OSGi Service Name   Available Rankings
   -----------------   ------------------
   loaderPlugin        1, 2
   txCallbackPlugin    1

Server: server2
   OSGi Service Name   Available Rankings
   -----------------   ------------------
   loaderPlugin        1, 2
   txCallbackPlugin    1

Summary - All servers have the same service rankings.
```

O exemplo a seguir mostra a saída do comando **osgiAll** quando o pacote configurável que inclui o loaderPlugin com classificação 1 é interrompido no server1. A mensagem de resumo na parte inferior da saída confirma que o server1 agora está ausente no loaderPlugin com classificação 1:

```
Server: server1
   OSGi Service Name   Available Rankings
   -----------------   ------------------
   loaderPlugin        2
   txCallbackPlugin    1

Server: server2
   OSGi Service Name   Available Rankings
   -----------------   ------------------
   loaderPlugin        1, 2
   txCallbackPlugin    1

Summary - The following servers are missing service rankings:
   Server  OSGi Service Name Missing Rankings
   ------  ----------------- ----------------
   server1 loaderPlugin      1
```

O exemplo a seguir mostra a saída se o nome do serviço é especificado com o argumento **-sn**, mas o serviço não existe:

```
Server: server2
   OSGi Service Name   Available Rankings
   ----------------- ------------------
   invalidPlugin     No service found

Server: server1
   OSGi Service Name   Available Rankings
   ----------------- ------------------
   invalidPlugin     No service found

Summary - All servers have the same service rankings.
```

- Execute o comando **osgiCheck** para verificar conjuntos de serviços de plug-in e classificações para ver se eles estão disponíveis.

  O comando **osgiCheck** aceita um ou mais conjuntos de classificações de serviço no formato: `-serviceRankings <service name>;<ranking>[,<serviceName>;<ranking>]`

Quando as classificações estão todas disponíveis, o método retorna com um nível de erro igual a 0. Se uma ou mais classificações estão indisponíveis, um erro de nível diferente de zero é configurado e uma tabela de todos os servidores que não incluem as classificações de serviço especificadas. Filtros adicionais podem ser usados para limitar a verificação de serviço para um subconjunto dos servidores disponíveis no domínio do eXtreme Scale.

Por exemplo, se a classificação ou o serviço especificado estiver ausente, a seguinte mensagem será exibida:

```
Server  OSGi Service Unavailable Rankings
------  ------------ --------------------
server1 loaderPlugin 3
server2 loaderPlugin 3
```

- Execute o comando **osgiUpdate** para atualizar a classificação de um ou mais plug-ins para todos os servidores em um único ObjectGrid e MapSet em uma única operação.

  O comando aceita um ou mais conjuntos de classificações de serviço no formato: `-serviceRankings <service name>;<ranking>[,<serviceName>;<ranking>] -g <grid name> -ms <mapset name>`

  Com este comando, é possível concluir as operações a seguir:

  – Verifique se os serviços especificados estão disponíveis para atualização em cada um dos servidores.
  – Altere o estado da grade para offline usando a interface StateManager. Consulte Gerenciando a Disponibilidade do ObjectGrid para obter mais informações. Este processo coloca a grade em modo quiesce e aguarda até que qualquer transação em execução tenha concluído e impede o início de qualquer nova transação. Este processo também sinaliza quaisquer plug-ins ObjectGridLifecycleListener e BackingMapLifecycleListener para descontinuar qualquer atividade transacional. Consulte Plug-ins para Fornecer Listeners de Eventos para obter informações sobre plug-ins do listener de eventos.
  – Atualize cada contêiner do eXtreme Scale em execução em uma estrutura OSGi para usar as novas versões de serviço.
  – Altere o estado da grade para online, permitindo que as transações continuem.

  O processo de atualização é idempotente, de forma que, se um cliente falhar ao concluir qualquer tarefa, isto resultará na operação sendo recuperada. Se um cliente for incapaz de executar a recuperação ou for interrompido durante o processo de atualização, o mesmo comando poderá ser emitido novamente e ele continuará na etapa apropriada.

  Se o cliente for incapaz de continuar, e o processo for reiniciado a partir de um outro cliente, use a opção `-force` para permitir que o cliente execute a atualização. O comando **xscmd.bat/xscmd.sh** impede que diversos clientes atualizem o mesmo conjunto de mapas simultaneamente. Para obter mais detalhes sobre o comando **osgiUpdate**, consulte Atualizando Serviços OSGi para Plug-ins do eXtreme Scale com **xscmd**.

Configurando Servidores com o OSGi Blueprint

É possível configurar os servidores de contêiner do WebSphere eXtreme Scale usando um arquivo XML do OSGi Blueprint, permitindo o empacotamento e o desenvolvimento simplificados de pacotes configuráveis do servidor autocontidos.

Antes de Iniciar

Este tópico assume que as seguintes tarefas foram concluídas:

- A estrutura do Eclipse Equinox OSGi foi instalada e iniciada com o contêiner de projeto do Eclipse Gemini ou do Apache Aries.
- O pacote configurável do servidor eXtreme Scale foi instalado e iniciado.
- O pacote configurável de plug-ins dinâmicos do eXtreme Scale foi criado.
- O arquivo XML do descritor do ObjectGrid do eXtreme Scale e o arquivo XML da política de implementação foram criados.

Sobre Esta Tarefa

Esta tarefa descreve como configurar um servidor do eXtreme Scale com um contêiner usando um arquivo XML do projeto. O resultado do procedimento é um pacote configurável do contêiner. Quando o pacote configurável do contêiner for iniciado, o pacote configurável do servidor eXtreme Scale controlará o pacote configurável, analisará o XML do servidor e iniciará um servidor e um contêiner.

Um pacote configurável do contêiner pode ser, opcionalmente, combinado com o aplicativo e os plug-ins do eXtreme Scale quando atualizações do plug-in dinâmico não são necessárias ou os plug-ins não suportam a atualização dinâmica.

Procedimento

1. Crie um arquivo XML do Blueprint com o namespace `objectgrid` incluído. É possível nomear o arquivo de qualquer jeito. No entanto, ele deve incluir o namespace do projeto:

```
<?xml version="1.0" encoding="UTF-8"?>

<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
           xmlns:objectgrid="http://www.ibm.com/schema/objectgrid"
           xsi:schemaLocation="http://www.ibm.com/schema/objectgrid
                http://www.ibm.com/schema/objectgrid/objectgrid.xsd">
...
</blueprint>
```

2. Inclua a definição de XML para o servidor eXtreme Scale com as propriedades de servidor apropriadas. Consulte o arquivo XML do descritor do Spring para obter detalhes sobre todas as propriedades de configuração disponíveis. Consulte o exemplo a seguir da definição de XML:

```
objectgrid:server
    id="xsServer"
   tracespec="ObjectGridOSGi=all=enabled"
   tracefile="logs/osgi/wxsserver/trace.log"
    jmxport="1199"
   listenerPort="2909">
    <objectgrid:catalog host="catserver1.mycompany.com" port="2809" />
    <objectgrid:catalog host="catserver2.mycompany.com" port="2809" />
</objectgrid:server>
```

3. Inclua a definição de XML para o contêiner do eXtreme Scale com a referência para a definição de servidor e os arquivos XML do descritor do ObjectGrid e de implementação do ObjectGrid integrados no pacote configurável; por exemplo:

```
<objectgrid:container id="container"
    objectgridxml="/META-INF/objectGrid.xml"
    deploymentxml="/META-INF/objectGridDeployment.xml"
    server="xsServer" />
```

4. Armazene o arquivo XML do Blueprint no pacote configurável do contêiner. O XML do Blueprint deve ser armazenado no diretório `OSGI-INF/blueprint` para que o contêiner do Blueprint seja localizado.

   Para armazenar o XML do Blueprint em um diretório diferente, você deve especificar o cabeçalho de manifesto Bundle-Blueprint; por exemplo:

```
Bundle-Blueprint: OSGI-INF/blueprint.xml
```

5. Empacote os arquivos em um arquivo JAR do pacote configurável único. Consulte o exemplo a seguir de uma hierarquia do diretório do pacote configurável:

```
MyBundle.jar
    /META-INF/manifest.mf
    /META-INF/objectGrid.xml
    /META-INF/objectGridDeployment.xml
    /OSGI-INF/blueprint/blueprint.xml
```

Resultados

Um pacote configurável do contêiner do eXtreme Scale agora está criado e pode ser instalado no Eclipse Equinox. Quando o pacote configurável do contêiner é iniciado, o ambiente de tempo de execução do servidor eXtreme Scale no pacote configurável do servidor eXtreme Scale irá iniciar automaticamente servidor do eXtreme Scale de singleton usando os parâmetros definidos no pacote configurável e um servidor de contêiner é iniciado. O pacote configurável pode ser interrompido e iniciado, o que resulta no contêiner parando e iniciando. O servidor é um singleton e não para quando o pacote configurável é iniciado pela primeira vez.

Capítulo 4. Amostras

Vários tutoriais, exemplos e amostras do WebSphere eXtreme Scale estão disponíveis.

Exemplos

Os seguintes tópicos ilustram os principais recursos do WebSphere eXtreme Scale.

- Exemplo da API do DataGrid
- Configurando Implementações Locais

Amostras de Comunidade

As seguintes amostras ilustram como usar o WebSphere eXtreme Scale em vários ambientes para exibir diferentes recursos do produto.

- **Estrutura de Serviço Assíncrono** - A estrutura de Serviço Assíncrono fornece uma malha de processamento escalável e tolerante a falhas para processamento assíncrono das mensagens. Para obter mais informações, incluindo como fazer download da amostra, consulte o Galeria de Amostras: Amostra da Estrutura de Serviço Assíncrono.
- **Segurança de autenticação do cliente** - Esta amostra descreve como configurar a autenticação que requer que o cliente forneça credenciais válidas antes que o servidor forneça acesso à qualquer grade. Para obter mais informações, incluindo como fazer download da amostra, consulte o Galeria de Amostras: Segurança de Autenticação do Cliente.
- **Criando mapas dinâmicos** - Essa amostra demonstra como criar mapas depois que a grade já tiver sido inicializada. Para o eXtreme Scale 7.0 e superior, é possível usar modelos para recuperar os mapas. Para obter mais informações, incluindo como fazer download da amostra, consulte o Galeria de Amostras: Criando Mapas Dinâmicos Após a Inicialização da Grade .
- **Replicação multimestre** - A amostra Introdução à Replicação Multimestre é fornecida para uma introdução rápida para a replicação multimestre (AP). Para obter mais informações, incluindo como fazer download da amostra, consulte o Galeria de Amostras: Amostra de Replicação Multimestre.
- **Consultas com a API EntityManager** - Esta amostra demonstra como usar as consultas em um mapa particionado distribuído com a API do EntityManager. Para obter mais informações, incluindo como fazer download da amostra, consulte o Galeria de Amostras: Executando Consultas em uma Grade Particionada usando a API do Gerenciador de Entidade.
- **Consultas paralelas com uma implementação ReduceGridAgent** -Demonstra como usar a API da Grade de Dados para executar uma consulta sobre cada partição na grade. Para obter mais informações, incluindo como fazer download da amostra, consulte a Galeria de Amostras: Executando Consultas em Paralelo usando um ReduceGridAgent.

Artigos com Tutoriais e Exemplos

Tabela 9. Artigos Disponíveis por Recurso

Artigo	Características
Criando aplicativos prontos para grade	API de ObjectMap, API de EntityManager, Consulta, Agentes, Java SE e EE, Estatísticas, Particionamento, Administração/Operações, Eclipse
Computação em estilo de grade e processamento de dados	API do EntityManager, Agentes
Criando uma alternativa de banco de dados escalável, resiliente e de alto desempenho	API do ObjectMap, Replicação, Particionamento, Administração/Operações, Eclipse
Melhorando xsadmin para WebSphere eXtreme Scale	Administração
Redbook: Guia do Usuário	Todos os tópicos

Teste Gratuito

Para começar a usar o WebSphere eXtreme Scale, faça download de uma versão gratuita para testar. É possível desenvolver aplicativos inovadores de alto desempenho ao estender o conceito de armazenamento em cache de dados usando os recursos avançados.

Download de Teste

Para fazer download de uma versão de teste gratuita do WebSphere eXtreme Scale, vá para Download de Teste do eXtreme Scale.

Após fazer download e descompactar a versão de teste do eXtreme Scale, navegue até o diretório `gettingstarted` e leia o arquivo `GETTINGSTARTED_README.txt`. Este tutorial permite iniciar o uso do eXtreme Scale, criar uma grade de dados em diversos servidores e executar alguns aplicativos simples para armazenar e recuperar dados em uma grade. Antes da implementação do eXtreme Scale em um ambiente de produção, há diversas opções a serem consideradas, incluindo a quantidade de servidores a serem usados, a quantidade de armazenamento em cada servidor e a replicação síncrona ou assíncrona.

Amostras de Arquivos de Propriedades

Os arquivos de propriedades do servidor contêm configurações para executar os servidores de catálogo e servidores de contêiner. É possível especificar um arquivo de propriedades do servidor para uma configuração do WebSphere Application Server ou independente. Os arquivos de propriedades do cliente contêm configurações para o cliente.

É possível usar as seguintes amostras de arquivos de propriedades que estão no diretório *wxs_install_root*`\properties` para criar seu arquivo de propriedades:

- `sampleServer.properties`
- `sampleClient.properties`

Amostra: Utilitário xsadmin

Com o utilitário **xsadmin**, é possível formatar e exibir informações textuais sobre sua topologia do WebSphere eXtreme Scale. O utilitário de amostra fornece um método para analisar e descobrir dados de implementação atuais e pode ser utilizado com uma base para criação de utilitários customizados.

Antes de Iniciar

- 7.1.1+ O utilitário **xsadmin** é fornecido como uma amostra de como é possível criar utilitários customizados para sua implementação. O utilitário **xscmd** é fornecido como um utilitário suportado para monitorar e administrar seu ambiente. Para obter informações adicionais, consulte Administrando com o Utilitário **xscmd**.
- Para o utilitário **xsadmin** exibir os resultados, sua topologia da grade de dados deverá ter sido criada. Os servidores de catálogos e servidores de contêiner devem ser iniciados. Consulte o Iniciando e Parando Servidores Independentes para obter informações adicionais.
- Verifique se a variável de ambiente *JAVA_HOME* está configurada para usar o ambiente de tempo de execução que foi instalado com o produto. Se estiver usando a versão de avaliação do produto, você deverá configurar a variável de ambiente *JAVA_HOME*.

Sobre Esta Tarefa

O utilitário **xsadmin** de amostra usa uma implementação de beans gerenciados (MBeans). Este aplicativo de monitoramento de amostra ativa rapidamente recursos de monitoramento integrados que podem ser estendidos usando as interfaces no pacote com.ibm.websphere.objectgrid.management. É possível examinar o código de origem do aplicativo **xsadmin** de amostra no arquivo *wxs_home*/samples/xsadmin.jar em uma instalação independente ou no arquivo *wxs_home*/xsadmin.jarem uma instalação do WebSphere Application Server.

É possível usar o utilitário **xsadmin** de amostra para visualizar o layout atual e o estado específico da grade de dados, como o conteúdo do mapa. Neste exemplo, o layout da grade de dados nesta tarefa consiste em uma única grade de dados *ObjectGridA* com um mapa *MapA* que pertence ao conjunto de mapas *MapSetA*. Este exemplo demonstra como é possível exibir todos os contêineres ativos dentro de uma grade de dados e imprimir métricas filtradas referentes ao tamanho do mapa *MapA*. Para visualizar todas as opções de comando possíveis, execute o utilitário **xsadmin** sem nenhum argumento ou com a opção **-help**.

Procedimento

1. Acesse o diretório bin.

   ```
   cd wxs_home/bin
   ```

2. Execute o utilitário **xsadmin**.
   - Para exibir a ajuda online, execute o comando a seguir:

     UNIX

     ```
     xsadmin.sh
     ```

     Windows

     ```
     xsadmin.bat
     ```

Você deve passar somente uma das opções listadas para o utilitário funcionar. Se nenhuma opção **-m** ou **-g** for especificada, o utilitário **xsadmin** imprime as informações para cada grade na topologia.

- Para ativar as estatísticas de todos os servidores, execute o seguinte comando:

  UNIX

  ```
  xsadmin.sh -g ObjectGridA -setstatsspec ALL=enabled
  ```

  Windows

  ```
  xsadmin.bat -g ObjectGridA -setstatsspec ALL=enabled
  ```

- Para exibir todos os contêineres on-line para uma grade, execute o seguinte comando:

  UNIX

  ```
  xsadmin.sh -g ObjectGridA -m MapSetA -containers
  ```

  Windows

  ```
  xsadmin.bat -g ObjectGridA -m MapSetA -containers
  ```

  Todas as informações do contêiner são exibidas. A seguir está um exemplo de saída:

  ```
  Connecting to Catalog service at localhost:1099


  *** Show all online containers for grid - ObjectGridA & mapset - MapSetA

  Host: 192.168.0.186
  Container: server1_C-0, Server:server1, Zone:DefaultZone
  Partition Shard Type
          0 Primary

  Num containers matching = 1
  Total known containers = 1
  Total known hosts = 1
  ```

  Atenção: Para obter essas informações quando o protocolo Segurança da Camada de Transporte/Secure Sockets Layer (TLS/SSL) estiver ativado, você deve iniciar os servidores de catálogos e de contêiner com o conjunto de portas do serviço JMX. Para configurar a porta de serviço JMX, é possível usar a opção **-JMXServicePort** no script **startOgServer** ou chamar o método setJMXServicePort na interface ServerProperties.

- Para conectar ao serviço de catálogo e exibir informações sobre MapA, execute o seguinte comando:

  UNIX

  ```
  xsadmin.sh -g ObjectGridA -m MapSetA -mapsizes -fm MapA
  ```

  Windows

  ```
  xsadmin.bat -g ObjectGridA -m MapSetA -mapsizes -fm MapA
  ```

  O tamanho do mapa especificado é exibido. A seguir está um exemplo de saída:

  ```
  Connecting to Catalog service at localhost:1099

  ****Displaying Results for Grid - ObjectGridA, MapSet - MapSetA*****
  ```

```
*** Listing Maps for server1 ***
 Nome do Mapa Partição Tamanho do Mapa Bytes Usados (B) Tipo de Shard
 MapA     0         0        0              Primário
```

- Para se conectar ao serviço de catálogo usando uma porta JMX específica e exibir informações sobre o mapa MapA, execute o seguinte comando:

  UNIX

  ```
  xsadmin.sh -g ObjectGridA -m MapSetA -mapsizes -fm MapA
    -ch CatalogMachine -p 6645
  ```

  Windows

  ```
  xsadmin.bat -g ObjectGridA -m MapSetA -mapsizes -fm MapA
    -ch CatalogMachine -p 6645
  ```

  O utilitário **xscmd** de amostra se conecta ao servidor MBean que está em execução em um servidor de catálogos. Um servidor de catálogos pode ser executado como um processo independente, como um processo do WebSphere Application Server ou integrado em um processo de aplicativo customizado. Utilize a opção **-ch** para especificar o nome do host do serviço de catálogo e a opção **-p** para especificar a porta de nomenclatura do serviço de catálogo.

  O tamanho do mapa especificado é exibido. A seguir está um exemplo de saída:

  ```
  Connecting to Catalog service at CatalogMachine:6645

  *****Displaying Results for Grid - ObjectGridA, MapSet - MapSetA*****


  *** Listing Maps for server1 ***
  Map Name: MapA  Partition #: 0  Map Size: 0  Shard Type: Primary
  Server Total: 0
  ```

- Para conectar-se a um serviço de catálogo hospedado em um processo do WebSphere Application Server, execute as etapas a seguir:

  A opção **-dmgr** é necessária ao conectar-se a um serviço de catálogo hospedado por qualquer processo do WebSphere Application Server ou cluster de processos. Utilize a opção **-ch** para especificar o nome do host se não `localhost` e a opção **-p** para substituir a porta de autoinicialização do serviço de catálogo, que utiliza o processo BOOTSTRAP_ADDRESS. A opção **-p** será necessária apenas se o BOOTSTRAP_ADDRESS não estiver configurado com o padrão de `9809`.

  Nota: A versão independente do WebSphere eXtreme Scale não pode ser usada para se conectar a um serviço de catálogo hospedado por um processo do WebSphere Application Server. Use o **xsadmin** que é um script incluído no diretório *was_root*`/bin`, que está disponível ao instalar o WebSphere eXtreme Scale no WebSphere Application Server ou no WebSphere Application Server Network Deployment.

  a. Navegue até o diretório bin do WebSphere Application Server:

     ```
     cd was_root/bin
     ```

  b. Ative o utilitário **xsadmin** usando o seguinte comando:

     UNIX

     ```
     xsadmin.sh -g ObjectGridA -m MapSetA -mapsizes -fm MapA -dmgr
     ```

     Windows

     ```
     xsadmin.bat -g ObjectGridA -m MapSetA -mapsizes -fm MapA -dmgr
     ```

  O tamanho do mapa especificado é exibido.

```
Connecting to Catalog service at localhost:9809

****Displaying Results for Grid - ObjectGridA, MapSet - MapSetA*****


*** Listing Maps for server1 ***
Map Name: MapA  Partition #: 0  Map Size: 0  Shard Type: Primary
Server Total: 0
```

- Para exibir o posicionamento configurado e de tempo de execução de sua configuração, execute um dos seguintes comandos:

  ```
  xsadmin -placementStatus
  xsadmin -placementStatus -g myOG -m myMapSet
  xsadmin -placementStatus -m myMapSet
  xsadmin -placementStatus -g myOG
  ```

  É possível definir o escopo do comando para exibir informações de posicionamento para a configuração inteira, para uma grade de dados única, para um único conjunto de mapas ou para uma combinação de uma grade de dados e conjunto de mapas. A seguir está um exemplo de saída:

```
**********Printing Placement Status for Grid - Grid, MapSet - mapSet**********

<objectGrid name="Grid" mapSetName="mapSet">
  <configuration>
    <attribute name="placementStrategy" value="FIXED_PARTITIONS"/>
    <attribute name="numInitialContainers" value="3"/>
    <attribute name="minSyncReplicas" value="0"/>
    <attribute name="developmentMode" value="true"/>
  </configuration>
  <runtime>
    <attribute name="numContainers" value="3"/>
    <attribute name="numMachines" value="1"/>
    <attribute name="numOutstandingWorkItems" value="0"/>
  </runtime>
</objectGrid>
```

Criando um perfil de configuração para o utilitário `xsadmin`

É possível salvar seus parâmetros especificados com frequência para o utilitário **xsadmin** em um arquivo de propriedades. Como resultado, as chamadas do utilitário **xsadmin** são mais curtas.

Antes de Iniciar

Crie uma implementação básica do WebSphere eXtreme Scale que inclui pelo menos um servidor de catálogos e pelo menos um servidor de contêiner. Para obter informações adicionais, consulte Script**startOgServer**.

Sobre Esta Tarefa

Consulte "Referência do Utilitário **xsadmin**" na página 207 para obter uma lista das propriedades que podem ser colocadas em um perfil de configuração para o utilitário **xsadmin**. Se você especificar um arquivo de propriedades e também um parâmetro correspondente como um argumento da linha de comandos, o argumento da linha de comandos substituirá o valor do arquivo de propriedades.

Procedimento

1. Crie um arquivo de propriedades do perfil de configuração. Este arquivo de propriedades deve conter quaisquer propriedades globais que deseja usar em todas as suas chamadas do comando **xsadmin**.

   Salve o arquivo de propriedades com qualquer nome escolhido. Por exemplo, o arquivo pode ser colocado no seguinte caminho: `/opt/ibm/WebSphere/wxs71/ObjectGrid/security/<my.properties>`.

Substitua <my.properties> pelo nome do seu arquivo. Por exemplo, as seguintes propriedades podem ser configuradas no seu arquivo:

- XSADMIN_TRUST_TYPE=jks
- XSADMIN_TRUST_PATH=/opt/ibm/WebSphere/wxs71/ObjectGrid/bin/security/key.jks
- XSADMIN_USERNAME=ogadmin

2. Execute o utilitário xsadmin com o arquivo de propriedades criado. Use o parâmetro **-profile** para indicar o local de seu arquivo de propriedades. Também é possível usar o parâmetro **-v** para exibir a saída detalhada.

```
./xsadmin.sh -l -v -password xsadmin -ssl -trustPass ogpass -profile
/opt/ibm/WebSphere/wxs71/ObjectGrid/security/<my.properties>
```

Referência do Utilitário xsadmin

É possível passar os argumentos para o utilitário **xsadmin** com dois métodos diferentes: com um argumento de linha de comandos ou com um arquivo de propriedades.

Argumentos xsadmin

É possível definir um arquivo de propriedades para o utilitário **xsadmin** com Versão 7.1 Fix 1 ou posterior. Ao criar um arquivo de propriedades, é possível salvar alguns dos argumentos usados frequentemente, como o nome do usuário. As propriedades que podem ser incluídas em um arquivo de propriedades na seguinte tabela. Se você especificar uma propriedade em um arquivo de propriedades e o argumento de linha de comandos equivalente, o valor do argumento de linha de comandos substituirá o valor do arquivo de propriedades.

Para obter informações adicionais sobre a definição de um arquivo de propriedades para o utilitário **xsadmin**, consulte "Criando um perfil de configuração para o utilitário **xsadmin**" na página 206.

Tabela 10. Argumentos para o Utilitário **xsadmin**

Argumento da Linha de Comandos	Nome da Propriedade Equivalente no Arquivo de Propriedades	Descrição e Valores Válidos
-bp	n/d	Indica a porta do listener. **Padrão:**2809
-ch	n/d	Indica o nome do host JMX para o servidor de catálogos. **Padrão:**localhost
-clear	n/d	Limpa o mapa especificado. **Permite os seguintes filtros:** -fm
-containers	n/d	Para cada grade de dados e conjunto de mapas, exibe uma lista de servidores de contêiner. **Permite os seguintes filtros:** -fnp
-continuous	n/d	Especifique este sinalizador se desejar resultados do tamanho de mapa contínuos para monitorar a grade de dados. Quando este comando é executado ando com o argumento **-mapsizes**, o tamanho do mapa será exibido a cada 20 segundos.
-coregroups	n/d	Exibe todos os grupos principais para o servidor de catálogos. Esse argumento é usado para diagnóstico avançado.
-dismissLink *<catalog_service_domain>*	n/d	Remove um link entre os 2 domínios de serviço de catálogo. Forneça o nome do domínio de serviço de catálogo estrangeiro ao qual se conectou anteriormente com o argumento **-establishLink**.

Tabela 10. Argumentos para o Utilitário **xsadmin** *(continuação)*

Argumento da Linha de Comandos	Nome da Propriedade Equivalente no Arquivo de Propriedades	Descrição e Valores Válidos
-dmgr	n/d	Indica se você está se conectando a um serviço de catálogo hospedado WebSphere Application Server. **Padrão:**`false`
-empties	n/d	Especifique este sinalizador se deseja mostrar os contêineres vazio na saída.
-establishLink *<foreign_domain_name> <host1:port1,host2:port2...>*	n/d	Conecta o domínio de serviço de catálogo a um domínio de serviço de catálogo estrangeiro. Use o seguinte formato: `-establishLink <foreign_domain_name> <host1:port1,host2:port2...>`. *foreign_domain_name* é o nome do domínio de serviço de catálogo estrangeiro e *host1:port1,host2:port2...* é uma lista separada por vírgula de nomes do host do servidor de catálogos e portas Object Request Broker (ORB) que ficam em execução neste domínio de serviço de catálogo.
-fc	n/d	Filtro apenas para este contêiner. Se estiver filtrando os servidores de contêiner em um ambiente do WebSphere Application Server Network Deployment, use o seguinte formato: *<cell_name>/<node_name>/<serverName_containerSuffix>* **Use com os argumentos a seguir: `-mapsizes`, `-teardown`,`-revisions`,`-getTraceSpec`,`-setTraceSpec`,`-getStatsSpec`,`-setStatsSpec`**
-fh	n/d	Filtros para apenas este host. **Use com os argumentos a seguir: `-mapsizes`, `-teardown`,`-revisions`,`-getTraceSpec`,`-setTraceSpec`,`-getStatsSpec`,`-setStatsSpec`,`-routetable`**
-fm	n/d	Filtra apenas esse mapa. **Use com os seguintes argumentos: `-clear`, `-mapsizes`**
-fnp	n/d	Filtra servidores que não possuem shards primários. **Use com os seguintes argumentos: `-containers`**
-fp	n/d	Filtro para apenas esta partição. **Use com os argumentos a seguir: `-mapsizes`, `-teardown`,`-revisions`,`-getTraceSpec`,`-setTraceSpec`,`-getStatsSpec`,`-setStatsSpec`,`-routetable`**
-fs	n/d	Filtros para apenas este servidor. Se estiver filtrando servidores de aplicativos em um ambiente WebSphere Application Server Network Deployment, utilize o seguinte formato: *<cell_name>/<node_name>/<server_name>* **Use com os argumentos a seguir: `-mapsizes`, `-teardown`,`-revisions`,`-getTraceSpec`,`-setTraceSpec`,`-getStatsSpec`,`-setStatsSpec`**
-fst	n/d	Filtro para apenas este tipo de shard. Especifique P para shards primários apenas, A para shards de réplica assíncrona apenas e S para shards de réplica síncrona apenas. **Use com os argumentos a seguir: `-mapsizes`, `-teardown`,`-revisions`,`-getTraceSpec`,`-setTraceSpec`,`-getStatsSpec`,`-setStatsSpec`**
-fz	n/d	Filtros para apenas esta zona. **Use com os argumentos a seguir: `-mapsizes`, `-teardown`,`-revisions`,`-getTraceSpec`,`-setTraceSpec`,`-getStatsSpec`,`-setStatsSpec`,`-routetable`**
-force	n/d	Força a ação que está no comando, desativando qualquer prompt preventivo. Esse argumento é útil para executar comandos em lote.
-g	n/d	Especifica o nome ObjectGrid.
-getstatsspec	n/d	Exibe a especificação de estatísticas atual. É possível configurar a especificação das estatísticas com o argumento **`-setstatsspec`**. **Permite os filtros a seguir:** `-fst -fc -fz -fs -fh -fp`
-getTraceSpec	n/d	Exibe a especificação de rastreio atual. É possível configurar a especificação de rastreio com o argumento **`-settracespec`**. **Permite os filtros a seguir:** `-fst -fc -fz -fs -fh -fp`
-h	n/d	Exibe a ajuda para o utilitário **`xsadmin`**, que inclui uma lista de argumentos.
-hosts	n/d	Exibe todos os hosts na configuração.

Tabela 10. Argumentos para o Utilitário **xsadmin** *(continuação)*

Argumento da Linha de Comandos	Nome da Propriedade Equivalente no Arquivo de Propriedades	Descrição e Valores Válidos
-jmxUrl	XSADMIN_JMX_URL	Especifica o endereço de um servidor de conector JMX API no seguinte formato: `service:jmx:`*`protocol`*`:`*`sap`*. As definições de variável *protocol* e *sap* a seguir: *protocol* Especifica o protocolo de transporte a ser usado para se conectar ao servidor de conector. *sap* Especifica o endereço no qual o servidor conector está localizado. Para obter mais informações sobre o formato da URL de serviço JMX, consulte Classe JMXServiceURL (SEJava 2 Platform 5.0).
-l	n/d	Exibe todas as grades de dados conhecidas e conjuntos de mapa.
-m	n/d	Especifica o nome do conjunto de mapas.
-mapsizes	n/d	Exibe o tamanho de cada mapa no servidor de catálogos para verificar se a distribuição de chave é uniforme sobre os shards. **Permite os seguintes filtros:** `-fm -fst -fc -fz -fs -fh -fp`
-mbeanservers	n/d	Exibe uma lista de todos os terminais do servidor MBean.
-overridequorum	n/d	Substitui a configuração de quorum para que os eventos do servidor de contêiner não sejam ignorados durante um cenário de falha do datacenter.
-password	XSADMIN_PASSWORD	Especifica a senha para efetuar login no utilitário **xsadmin**. Não especifique a senha no seu arquivo de propriedades se desejar que sua senha permaneça segura.
-p	n/d	Indica a porta JMX para o host do servidor de catálogos. **Padrão:** 1099 ou 9809 para um host WebSphere Application Server e 1099 para configurações independentes.
-placementStatus	n/d	Exibe o posicionamento configurado e o posicionamento de tempo de execução da sua configuração. É possível realizar escopo da saída para uma combinação das grades de dados e dos conjuntos de mapa, ou para a configuração inteira: • Configuração Inteira: `-placementStatus` • Para uma grade de dados específica: `-placementStatus -g` *`my_grid`* • Para um conjunto de mapas específico: `-placementStatus -m` *`my_mapset`* • Para um conjunto de mapas e grade de dados específicos: `-placementStatus -g` *`my_grid`* `-m` *`my_mapset`*
-primaries	n/d	Exibe uma lista dos shards primários.
-profile	n/d	Especifica o caminho completo para o arquivo de propriedades para o utilitário **xsadmin**.
-quorumstatus	n/d	Exibe o status de quorum para o serviço de catálogo.
-releaseShard <*container_server_name*> <*objectgrid_name*> <*map_set_name*> <*partition_name*>	n/d	Usado em conjunto com o argumento **-reserveShard**. O argumento **-releaseShard** deverá ser chamado depois que um shard for reservado e colocado. . O argumento **-releaseShard** chama o método ContainerMBean.release() .
-reserved	n/d	Usado com o argumento **-containers** para exibir apenas shards que foram reservados com o argumento **-reserveShard**.
-reserveShard <*container_server_name*> <*objectgrid_name*> <*map_set_name*> <*partition_name*>	n/d	Move o shard primário para o servidor de contêiner especificado. O método ContainerMBean.reserve() é chamado por esse argumento.
-resumeBalancing <*objectgrid_name*> <*map_set_name*>	n/d	Tenta equilibrar solicitações e permite futuras tentativas de re-equilíbrio para o ObjectGrid e para o conjunto de mapas especificado.

Tabela 10. Argumentos para o Utilitário **xsadmin** *(continuação)*

Argumento da Linha de Comandos	Nome da Propriedade Equivalente no Arquivo de Propriedades	Descrição e Valores Válidos
-revisions	n/d	Exibe identificadores de revisão para um domínio de serviço de catálogo, incluindo: cada grade de dados, número de partição, tipo de partição (primário ou réplica), domínio de serviço de catálogo, ID e o número de revisões de dados para cada shard específico. Esse argumento pode ser usado para determinar se uma réplica assíncrona ou domínio vinculado é capturado. Esse argumento chama o método ObjectGridMBean.getKnownRevisions(). **Permite os filtros a seguir:** `-fst -fc -fz -fs -fh -fp`
-routetable	n/d	Exibe o estado atual da grade de dados a partir de uma perspectiva de servidor do cliente. A tabela de rotas é a informação que um servidor do cliente do ObjectGrid usa para se comunicar com a grade de dados. Use a tabela de rota como um auxílio de diagnóstico quando estiver tentando identificar problemas de conexão ou exceções `TargetNotAvailable`. **Argumentos obrigatórios:** Em um ambiente independente, você deve especificar os parâmetros **-bp** e **-p** com esse argumento se não estiver usando os valores padrão para a porta listener de autoinicialização e a porta JMX para o host do servidor de catálogos. **Permite os seguintes filtros:** `-fz -fh -fp`
-settracespec <*trace_string*>	n/d	Ativa o rastreio nos servidores durante o tempo de execução. Consulte o seguinte exemplo: `-setTraceSpec "ObjectGridReplication=all=enabled"` Consulte Coletando Rastreio e Opções de Rastreio para obter informações adicionais sobre as sequências de rastreio que são possíveis de especificar. **Permite os filtros a seguir:** `-fst -fc -fz -fs -fh -fp`
-swapShardWithPrimary <*container_server_name*> <*objectgrid_name*> <*map_set_name*> <*partition_name*>	n/d	Troca o shard de réplica especificados a partir do servidor de contêiner especificado com o shard primário. Ao executar esse comando, é possível equilibrar manualmente os shards primários quando necessário.
-setstatsspec <*stats_spec*>	n/d	Ativa a reunião de estatísticas. Esse argumento chama os métodos DynamicServerMBean.setStatsSpec e DynamicServerMBean.getStatsSpec. Consulte a Classe StatsSpec para obter mais informações sobre os módulos estatísticos que podem ser monitorados. **Permite os seguintes filtros:** `-fm -fst -fc -fz -fs -fh -fp`
-suspendBalancing <*objectgrid_name*> <*map_set_name*>	n/d	Evita que tentativas futuras equilibrem o ObjectGrid e o conjunto de mapas especificados.
-ssl	n/d	Indica que o Secure Sockets Layer (SSL) está ativado.
-teardown	n/d	Para uma lista ou grupo de servidores de contêiner e catálogo. **Permite os filtros a seguir:** `-fst -fc -fz -fs -fh -fp` **Formato para fornecer uma lista de servidores:** *server_name_1*,*server_name_2* ... **Pare todos os servidores em uma zona, inclua o argumento -fz:** –fz <*zone_name*> **Pare todos os servidores em um host, inclua o argumento -fh:** –fh <*host_name*>
-triggerPlacement	n/d	Força o posicionamento de shard a ser executado, ignorando valor **numInitialContainers** configurado no arquivo XML de implementação. É possível usar este argumento quando estiver executando a manutenção em seu servidor para permitir que o posicionamento de shard continue em execução, mesmo se o valor **numInitialContainers** for menor que o valor configurado.
-trustPass	XSADMIN_TRUST_PASS	Especifica a senha para o truststore especificado.
-trustPath	XSADMIN_TRUST_PATH	Especifica um caminho para o arquivo de truststore. Exemplo: `etc/test/security/server.public`

Tabela 10. Argumentos para o Utilitário **xsadmin** *(continuação)*

Argumento da Linha de Comandos	Nome da Propriedade Equivalente no Arquivo de Propriedades	Descrição e Valores Válidos
-trustType	XSADMIN_TRUST_TYPE	Especifica o tipo de truststore. Valores válidos: JKS, JCEK, PKCS12, etc.
-unassigned	n/d	Exibe uma lista de shards que não podem ser colocados na grade de dados. Os shards não podem ser colocados quando o serviço de posicionamento tiver uma restrição que esteja evitando o posicionamento.
-username	XSADMIN_USERNAME	Especifica o nome de usuário para efetuar login no utilitário **xsadmin**.
-v	n/d	Ativa a ação da linha de comandos detalhada. Use este sinalizador se estiver usando variáveis de ambiente, um arquivo de propriedades ou ambos para especificar certos argumentos de linha de comandos e desejar visualizar seus valores. Consulte "Opção Detalhada para o Utilitário **xsadmin**" para obter informações adicionais.
-xml	n/d	Imprime a saída não filtrada a partir do método PlacementServiceMBean.listObjectGridPlacement(). Os outros Argumentos **xsadmin** filtram a saída deste método e organizam os dados em um formato mais consumível.

Opção Detalhada para o Utilitário xsadmin

É possível usar a opção detalhada **xsadmin** para resolução de problemas. Execute o comando `xsadmin -v` para listar todos os parâmetros configurados. A opção detalhada exibe todos os valores em todos os escopos, incluindo argumentos da linha de comandos, argumentos do arquivo de propriedades e os argumentos especificados para o ambiente. A seção `Argumentos Efetivos` inclui as configurações que estão sendo usadas no ambiente se você tiver especificado a mesma propriedade usando diversos escopos.

Exemplo de Opção Detalhada

Argumentos do comando xsadmin:

O texto a seguir é um exemplo da saída ao usar a opção detalhada a partir da linha de comandos após executar o seguinte comando com um valor de propriedades especificadas:

```
./xsadmin -l -v -username xsadmin -password xsadmin -ssl -trustPass ogpass
-profile /opt/ibm/WebSphere/wxs71/ObjectGrid/security/my.properties
```

Argumentos do arquivo de propriedades:

O conteúdo do arquivo `/opt/ibm/WebSphere/wxs71/ObjectGrid/security/my.properties` `properties` é o seguinte:

```
XSADMIN_TRUST_PASS=ogpass
XSADMIN_TRUST_TYPE=jks
XSADMIN_TRUST_PATH=/opt/ibm/WebSphere/wxs71/ObjectGrid/bin/security/key.jks
XSADMIN_USERNAME=ogadmin
XSADMIN_PASSWORD=ogpass
```

Resultados do comando:

Na seguinte saída a partir do comando **xsadmin** anterior, o texto que está em ***negrito e itálico*** indica as propriedades e os valores que são especificados na linha de comandos e no arquivo de propriedades. Na seção `Argumentos da linha de comandos efetivos`, observa-se que os argumentos especificados da linha de comandos substituem os valores no arquivo de propriedades.

```
Command line specified arguments
*********************************
XSADMIN_USERNAME=xsadmin
XSADMIN_PASSWORD=xsadmin
XSADMIN_TRUST_PATH=<unspecified>
XSADMIN_TRUST_TYPE=<unspecified>
XSADMIN_TRUST_PASS=ogpass
XSADMIN_PROFILE=/opt/ibm/WebSphere/wxs71/ObjectGrid/security/my.properties
XSADMIN_JMX_URL=<unspecified>
*********************************
Properties file specified arguments
***********************************
XSADMIN_USERNAME=ogadmin
XSADMIN_PASSWORD=ogpass
XSADMIN_TRUST_PATH=/opt/ibm/WebSphere/wxs71/ObjectGrid/bin/security/key.jks
XSADMIN_TRUST_TYPE=jks
XSADMIN_TRUST_PASS=ogproppass
XSADMIN_JMX_URL=<unspecified>
*********************************
Environment-specified arguments
*********************************
XSADMIN_USERNAME=<unspecified>
XSADMIN_PASSWORD=<unspecified>
XSADMIN_TRUST_PATH=<unspecified>
XSADMIN_TRUST_TYPE=<unspecified>
XSADMIN_TRUST_PASS=<unspecified>
XSADMIN_JMX_URL=<unspecified>
*********************************
Effective arguments
*********************************
XSADMIN_USERNAME=xsadmin
XSADMIN_PASSWORD=xsadmin
XSADMIN_TRUST_PATH=/opt/ibm/WebSphere/wxs71/ObjectGrid/bin/security/key.jks
XSADMIN_TRUST_TYPE=jks
XSADMIN_TRUST_PASS=ogpass
XSADMIN_PROFILE=/opt/ibm/WebSphere/wxs71/ObjectGrid/security/my.properties
XSADMIN_JMX_URL=<unspecified>
SSL authentication enabled: true
*********************************
Connecting to Catalog service at localhost:1099
*** Show all 'objectGrid:mapset' names
Grid Name  MapSet Name
accounting defaultMapSet
```

Atenção: A propriedade XSADMIN_PROFILE, embora seja exibida na saída detalhada, não é uma chave válida que possa ser especificada em um arquivo de propriedades. O valor desta propriedade na saída detalhada indica o valor da propriedade que está sendo usado, conforme indicado no argumento da linha de comandos **-profile**.

Saída Sem a Opção Detalhada

A seguir há um exemplo da mesma saída do comando sem a opção detalhada ativada:

```
./xsadmin -l -username xsadmin -password xsadmin -ssl -trustPass ogpass
-profile /opt/ibm/WebSphere/wxs71/ObjectGrid/security/my.properties

Connecting to Catalog service at localhost:1099
*** Show all 'objectGrid:mapset' names
Grid Name  MapSet Name
accounting defaultMapSet
```

Avisos

Referências nesta publicação a produtos, programas ou serviços IBM não significam que a IBM pretende torná-los disponíveis em todos os países onde opera. Qualquer referência a produtos, programas ou serviços IBM não significa que apenas produtos, programas ou serviços IBM possam ser utilizados. Qualquer produto, programa ou serviço funcionalmente equivalente, que não infrinja nenhum direito de propriedade intelectual da IBM poderá ser utilizado em substituição a este produto, programa ou serviço. A avaliação e verificação da operação em conjunto com outros produtos, exceto aqueles expressamente designados pela IBM, são de inteira responsabilidade do Cliente.

A IBM pode ter patentes ou solicitações de patentes pendentes relativas a assuntos tratados nesta publicação. O fornecimento desta publicação não garante ao Cliente nenhum direito sobre tais patentes. Pedidos de licença devem ser enviados, por escrito, para:

Gerência de Relações Comerciais e Industriais da IBM Brasil
Av. Pasteur, 138-146
Botafogo
Rio de Janeiro, RJ
CEP 22290-240

Licenciados deste programa que desejam obter mais informações sobre este assunto com objetivo de permitir: (i) a troca de informações entre programas criados independentemente e outros programas (incluindo este) e (ii) a utilização mútua das informações trocadas, devem entrar em contato com:

Gerência de Relações Comerciais e Industriais da IBM Brasil
Av. Pasteur, 138-146,
Botafogo
Rio de Janeiro, RJ
CEP 22290-240
Brasil

Tais informações podem estar disponíveis, sujeitas a termos e condições apropriados, incluindo em alguns casos, o pagamento de uma taxa.

Marcas Registradas

Os termos a seguir são marcas registradas da IBM Corporation nos Estados Unidos e/ou em outros países:

- AIX
- CICS
- Cloudscape
- DB2
- Domino
- IBM
- Lotus
- RACF
- Redbooks
- Tivoli
- WebSphere
- z/OS

Java e todas as marcas registradas baseadas em Java são marcas registradas da Sun Microsystems, Inc. nos Estados Unidos e/ou em outros países.

LINUX é uma marca registrada de Linus Torvalds nos Estados Unidos e/ou em outros países.

Microsoft, Windows, Windows NT e o logotipo do Windows são marcas registradas da Microsoft Corporation nos Estados Unidos e/ou em outros países.

UNIX é uma marca registrada do The Open Group nos Estados Unidos e em outros países.

Outros nomes de empresas, produtos e serviços podem ser marcas registradas ou marcas de serviços de terceiros.

Índice Remissivo

S

T

U

V

W

Z

Impresso no Brasil