IBM WebSphere eXtreme Scale Versión 7.1.1
Versión 7 Release 1

Visión general del producto

21 de noviembre de 2011

Esta edición se aplica a la versión 7, release 1, modificación 1 de WebSphere eXtreme Scale y a todos los releases y modificaciones posteriores hasta que se indique lo contrario en nuevas ediciones.

Contenido

Figuras . . . . . **v**

Tablas . . . . . **vii**

Acerca de la *Visión general del producto* **ix**

Capítulo 1. Visión general del producto **1**

WebSphere eXtreme Scalevisión general 1
Novedades de la versión 7.1.1 4
Notas del release 6
Requisitos de hardware y software 7
Convenios de directorio 8
Visión general técnica de WebSphere eXtreme Scale 10
Visión general de memoria caché 11
Arquitectura de memoria caché: correlaciones, contenedores, clientes y catálogos 11
Zonas 17
Desalojadores. 21
Visión general de la infraestructura OSGi . . . 24
Visión general de integración de memoria caché . . 26
Plug-in de memoria caché de nivel 2 (L2) JPA . . 26
Gestión de sesiones HTTP 33
Proveedor de memoria caché dinámica 35
Integración de base de datos: almacenamiento en memoria caché de grabación diferida, en línea y complementaria 48
Memoria caché escasa y completa 49
Memoria caché complementaria 50
Memoria caché en línea 50
Almacenamiento en memoria caché de grabación diferida. 53
Cargadores 57
Precarga de datos y calentamiento. 59
Técnicas de sincronización de base de datos . . 61
Invalidación de datos 62
Índices 64
Cargadores JPA 66
Visión general de serialización 68
Serialización mediante Java 70
Plug-in ObjectTransformer 71
Serialización mediante plug-ins DataSerializer . . 75
Visión general de la escalabilidad 76
Cuadrículas de datos, particiones y fragmentos 76
Particionamiento. 78
Colocación y particiones 80
Transacciones de partición única y transacciones entre cuadrículas de datos 84
Escalado en unidades o contenedores. 90
Visión general de disponibilidad 92
Alta disponibilidad 92
Réplicas y fragmentos 109
Visión general del proceso de transacciones . . 120
Visión general de seguridad 135
Visión general de los servicios de datos REST 137

Capítulo 2. Planificación. 141

Planificación de la topología 141
Almacenamiento local de memoria caché en memoria 142
Memoria caché local replicada de igual. . . . 143
Memoria caché incorporada 145
Memoria caché distribuida 146
Integración de base de datos: almacenamiento en memoria caché de grabación diferida, en línea y complementaria 148
Planificación de topologías de varios centros de datos 167
Interoperatividad con otros productos WebSphere 180

Capítulo 3. Escenarios 181

Utilización de un entorno OSGi para desarrollar y ejecutar plug-ins de eXtreme Scale 181
Visión general de la infraestructura OSGi . . . 181
Instalación de la infraestructura OSGi de Eclipse Equinox con Eclipse Gemini para clientes y servidores 183
Creación y ejecución de plug-ins dinámicos de eXtreme Scale para su uso en un entorno OSGi . 186
Ejecución de contenedores de eXtreme Scale con plug-ins dinámicos en un entorno OSGi . . . 194

Capítulo 4. Ejemplos 205

Prueba gratuita. 206
Archivos de propiedades de ejemplo 206
Ejemplo: Programa de utilidad **xsadmin**. 207
Creación de un perfil de configuración para el programa de utilidad **xsadmin** 210
Referencia del programa de utilidad **xsadmin** 211
Opción verbose del programa de utilidad **xsadmin** 215

Avisos 219

Marcas registradas 221

Índice. 223

Figuras

1. Topología de nivel superior 3
2. Servicio de catálogo 12
3. Dominio de servicio de catálogo 13
4. Servidor de contenedor 13
5. Partición 14
6. Fragmento 14
7. ObjectGrid 15
8. Correlación 15
9. Conjuntos de correlaciones 15
10. Topologías posibles 17
11. Primarios y réplicas en las zonas 18
12. Topología interno del dominio JPA 28
13. Topología incorporada JPA 29
14. Topología incorporada con particiones JPA 30
15. Topología remota JPA 32
16. Topología de gestión de sesiones HTTP con una configuración de contenedor remoto . . . 34
17. ObjectGrid como un almacenamiento intermedio de base de datos 48
18. ObjectGrid como una memoria caché secundaria 49
19. Memoria caché complementaria 50
20. Memoria caché en línea 51
21. Almacenamiento en memoria caché de lectura directa 52
22. Almacenamiento en memoria caché de grabación directa 52
23. Almacenamiento en memoria caché de grabación diferida 53
24. Almacenamiento en memoria caché de grabación diferida 54
25. Cargador 58
26. Plug-in Loader 60
27. Cargador de clientes 61
28. Renovación periódica 62
29. Arquitectura del cargador JPA 67
30. Vía de acceso de comunicación entre un fragmento primario y fragmentos de réplica . 110
31. Colocación de un conjunto de correlaciones de ObjectGrid con una política de despliegue de 3 particiones con un valor minSyncReplicas de 1, un valor maxSyncReplicas de 1 y un valor maxAsyncReplicas de 1 113
32. Colocación de ejemplo de un conjunto de correlaciones ObjectGrid para la partición partition0. La política del despliegue tiene un valor minSyncReplicas de 1, un valor maxSyncReplicas de 2 y un valor maxAsyncReplicas de 1. 117
33. El contenedor del fragmento primario falla. 117
34. El fragmento de réplica síncrona en el contenedor 2 de ObjectGrid pasa a ser el fragmento primario. 118
35. La máquina B contiene el fragmento primario. En función de cómo se establece la modalidad de reparación automática y la disponibilidad de los contenedores, un nuevo fragmento de réplica síncrona se podría colocar o no en una máquina. 118
36. Microsoft WCF Data Services 137
37. Servicio de datos REST de WebSphere eXtreme Scale 138
38. Escenario de memoria caché en memoria local 142
39. La memoria caché duplicada por un igual con los cambios que se propagan con JMS . . . 143
40. La memoria caché duplicada por un igual con los cambios propagados con el High Availability Manager. 144
41. Memoria caché incorporada 145
42. Memoria caché distribuida 147
43. Memoria caché cercana 147
44. ObjectGrid como un almacenamiento intermedio de base de datos 149
45. ObjectGrid como una memoria caché secundaria 149
46. Memoria caché complementaria 151
47. Memoria caché en línea 152
48. Almacenamiento en memoria caché de lectura directa 153
49. Almacenamiento en memoria caché de grabación directa 153
50. Almacenamiento en memoria caché de grabación diferida 154
51. Almacenamiento en memoria caché de grabación diferida 155
52. Cargador 159
53. Plug-in Loader 161
54. Cargador de clientes 162
55. Renovación periódica 163
56. Proceso de Eclipse Equinox para incluir toda la configuración y los metadatos en un paquete OSGi 197
57. Proceso de Eclipse Equinox para especificar la configuración y los metadatos fuera de un paquete OSGi 197

Tablas

1. Comparación de características 38
2. Integración de tecnología sin fisuras 39
3. Interfaces de programación 40
4. Resumen del descubrimiento de anomalías y la recuperación 95
5. Valor de estado y respuesta 97
6. Secuencia de confirmación del fragmento primario 99
7. Proceso de confirmación síncrona 99
8. Enfoques de arbitraje 175
9. Artículos disponibles por característica 206
10. Argumentos del programa de utilidad **xsadmin** 211

Acerca de la *Visión general del producto*

El conjunto de documentación de WebSphere eXtreme Scale incluye tres volúmenes que proporcionan la información necesaria para utilizar, programar y administrar el producto WebSphere eXtreme Scale.

Biblioteca de WebSphere eXtreme Scale

La biblioteca de WebSphere eXtreme Scale contiene las siguientes publicaciones:

- El *Visión general del producto* contiene una vista de nivel superior de los conceptos de WebSphere eXtreme Scale, incluidos casos de ejemplo y guías de aprendizaje.
- *Guía de instalación* describe cómo instalar topologías comunes de WebSphere eXtreme Scale.
- La *Guía de administración* contiene la información necesaria para los administradores del sistema, incluido cómo planificar despliegues de aplicaciones, planificar la capacidad, instalar y configurar el producto, iniciar y detener servidores, supervisar el entorno y proteger el entorno.
- La *Guía de programación* contiene información dirigida a los desarrolladores de aplicaciones que indica cómo desarrollar aplicaciones para WebSphere eXtreme Scale utilizando la información de API incluida.

Para descargar las publicaciones, vaya a la página de la biblioteca WebSphere eXtreme Scale.

También puede acceder a la misma información en esta biblioteca en el Information Center de WebSphere eXtreme Scale Versión 7.1.1.

Utilización fuera de línea de los manuales

Todos los manuales de la biblioteca de WebSphere eXtreme Scale contienen enlaces al Information Center, con el siguiente URL raíz: `http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r1m1`. Estos enlaces le llevan directamente a la información relacionada. Sin embargo, si está trabajando fuera de línea y se encuentra con uno de estos enlaces, puede buscar el título del enlace en los otros manuales de la biblioteca. La documentación de la API, el glosario y los mensajes de referencia no están disponibles en los manuales en formato PDF.

Quién debe utilizar esta publicación

Esta publicación va dirigida a cualquier usuario que le interese obtener conocimientos sobre WebSphere eXtreme Scale.

Cómo obtener actualizaciones de esta publicación

Puede obtener actualizaciones para esta publicación descargando la versión más reciente desde la página de la biblioteca de WebSphere eXtreme Scale.

Envío de comentarios

Póngase en contacto con el equipo de documentación. ¿Ha encontrado lo que necesita? ¿Ha sido la información precisa y completa? Envíe sus comentarios sobre

esta documentación mediante correo electrónico a wasdoc@us.ibm.com.

Capítulo 1. Visión general del producto

WebSphere eXtreme Scale es una cuadrícula de datos elástica, escalable y en memoria. La cuadrícula de datos dinámicamente almacena en memoria caché, particiona, replica y gestiona la lógica empresarial y los datos de aplicación entre varios servidores. WebSphere eXtreme Scale realiza volúmenes masivos de proceso de transacciones con un alta eficacia y una escalabilidad lineal. Con WebSphere eXtreme Scale, también puede obtener calidades de servicio como, por ejemplo, integridad transaccional, alta disponibilidad y tiempos de respuesta predecibles.

WebSphere eXtreme Scalevisión general

WebSphere eXtreme Scale es una cuadrícula de datos elástica, escalable y en memoria. La cuadrícula de datos dinámicamente almacena en memoria caché, particiona, replica y gestiona la lógica empresarial y los datos de aplicación entre varios servidores. WebSphere eXtreme Scale realiza volúmenes masivos de proceso de transacciones con un alta eficacia y una escalabilidad lineal. Con WebSphere eXtreme Scale, también puede obtener calidades de servicio como, por ejemplo, integridad transaccional, alta disponibilidad y tiempos de respuesta predecibles.

WebSphere eXtreme Scale se puede utilizar de distintas formas. Puede utilizar el producto como una memoria caché muy potente, como un espacio de proceso de base de datos en memoria para gestionar el estado de aplicación, o para crear aplicaciones XTP (Extreme Transaction Processing). Estas prestaciones XTP incluyen una infraestructura de la aplicación para dar soporte a las aplicaciones críticas para la empresa más exigentes.

Escalabilidad elástica

Es posible escalabilidad elástica mediante la utilización del almacenamiento distribuido en memoria caché de objetos. Con la escalabilidad flexible, la cuadrícula de datos se supervisa y se gestiona sola. La cuadrícula de datos puede añadir o eliminar servidores de la topología, lo que aumenta o disminuye la memoria, el rendimiento de la red y la capacidad de proceso, según se requiera. Cuando se inicia un proceso de escalado, la capacidad se añade a la cuadrícula de datos mientras esta se ejecuta, sin necesidad de reiniciar. Por el contrario, el proceso de reducción elimina inmediatamente la capacidad. La cuadrícula de datos también se auto-arregla recuperándose automáticamente de anomalías.

WebSphere eXtreme Scale frente a una base de datos en memoria

WebSphere eXtreme Scale no se puede considerar una base de datos en memoria real. Una base de datos en memoria es demasiado simple para manejar algunas de las complejidades que puede gestionar WebSphere eXtreme Scale. Si una base de datos en memoria tiene un servidor que falla, esta no puede reparar el problema. Una anomalía puede ser desastrosa si todo el entorno se encuentra en ese servidor.

Para enfrentarse al problema de este tipo de anomalía, eXtreme Scale divide el conjunto de datos especificado en particiones, que son equivalentes a tres esquemas de árbol limitado. Los esquemas de árbol limitado describen la relación entre entidades. Cuando se utilizan particiones, las relaciones de entidad deben

modelar una estructura de datos de árbol. En esta estructura, la cabeza del árbol es la entidad raíz y es la única entidad que se particiona. Todos los demás hijos de la entidad raíz se almacenan en la misma partición que la entidad raíz. Cada partición existe como una copia primaria o fragmento. Una partición contienen también fragmentos replica para hacer una copia de seguridad de los datos. Un base de datos en memoria no puede proporcionar esta función porque no está estructurada ni es dinámica de esta forma. Con una base de datos en memoria, debe implementar las operaciones que WebSphere eXtreme Scale realiza automáticamente. Puede ejecutar operaciones de SQL en bases de datos en memoria, lo que mejora la velocidad de proceso en comparación con las bases de datos que no están en memoria. WebSphere eXtreme Scale tiene su propio lenguaje de consulta en lugar de soporte de SQL. Este lenguaje de consulta es más elástico, permite particionamiento de datos y proporciona recuperación fiable de anomalías.

WebSphere eXtreme Scale con bases de datos

Con la característica de memoria caché de grabación diferida, WebSphere eXtreme Scale puede actuar como una memoria caché frontal para una base de datos. Gracias al uso de esta memoria caché frontal, el rendimiento aumenta mientras se reduce la carga de la base de datos y los conflictos. WebSphere eXtreme Scale proporciona una escalada horizontal y una escalada vertical a un coste de proceso predecible.

La imagen siguiente muestra que en un entorno de memoria caché coherente distribuido, los clientes de eXtreme Scale envían y reciben datos de la cuadrícula de datos. La cuadrícula de datos se puede sincronizar automáticamente con un almacén de datos de fondo. La memoria caché es coherente porque todos los clientes ven los mismos datos en la memoria caché. Cada fragmento de datos se almacena exactamente en un servidor grabable en la memoria caché. Tener una sola copia de cada fragmento de datos evita tener copias inútiles de registros que podrían contener distintas versiones de los datos. Una memoria caché coherente contiene más datos a medida que se añaden más servidores a la cuadrícula de datos, y se amplía de forma lineal, a medida que la cuadrícula de datos crece en tamaño. De forma opcional, también se pueden duplicar los datos para la tolerancia al error adicional.

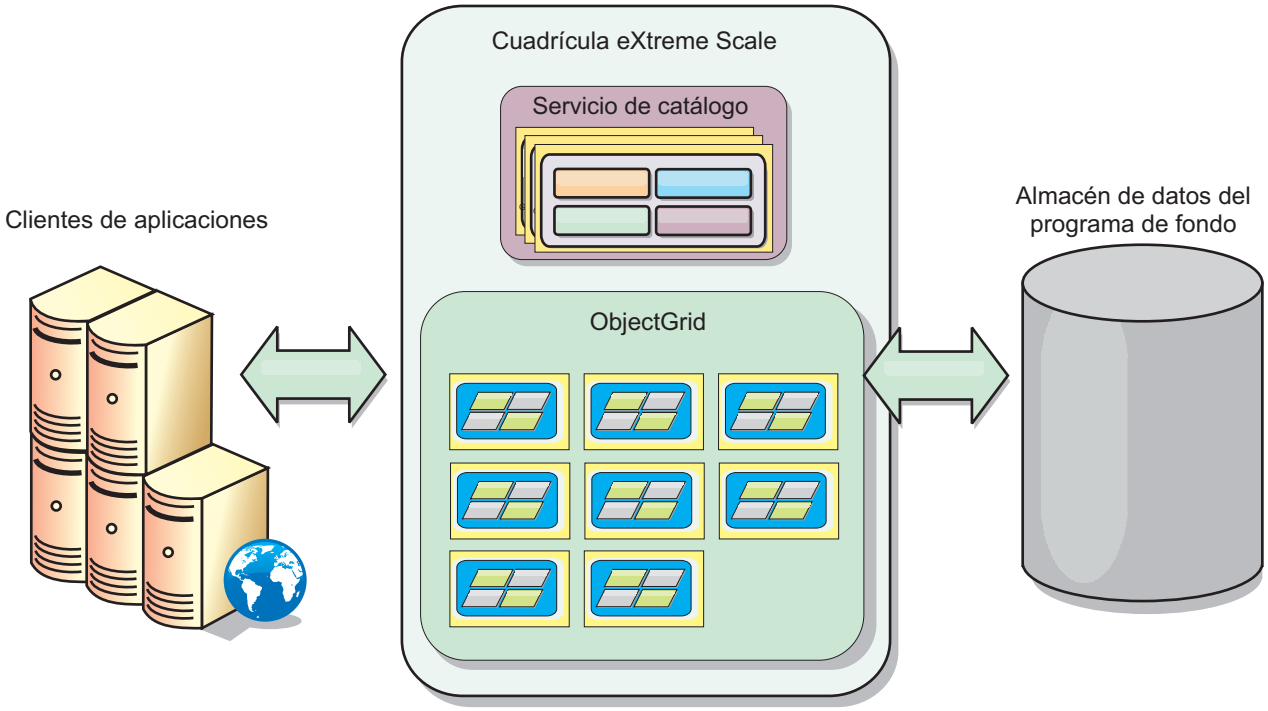


Figura 1. Topología de nivel superior

WebSphere eXtreme Scale tiene servidores, denominados *servidores de contenedor*, que proporcionan su cuadrícula de datos en memoria. Estos servidores pueden ejecutarse en WebSphere Application Server o en máquinas virtuales Java Java Standard Edition (J2SE) individuales. Se puede ejecutar más de un servidor de contenedor en un único servidor físico. Como resultado, la cuadrícula de datos en memoria puede ser grande. La cuadrícula de datos no está limitada por, ni repercute sobre, la memoria ni el espacio de dirección de la aplicación o del servidor de aplicaciones. La memoria puede ser la suma de la memoria de varios cientos, o miles, de máquinas virtuales Java, en ejecución en varios servidores físicos distintos.

Como un espacio de proceso de base de datos en memoria, se puede realizar copia de seguridad de WebSphere eXtreme Scale mediante disco, base de datos, o ambos.

Mientras que eXtreme Scale proporciona varias API Java, muchos casos de uso no requieren ninguna programación de usuario, solo la configuración y el despliegue de la infraestructura WebSphere.

Visión general de la cuadrícula de datos

La interfaz de programación más simple de eXtreme Scale es la interfaz ObjectMap, que es una interfaz de correlación simple que incluye: un método map.put(key,value) para poner un valor en la memoria caché y un método map.get(key) para recuperar posteriormente el valor.

El paradigma de cuadrícula de datos básico es un par de clave-valor, donde la cuadrícula de datos almacena valores (objetos Java), con una clave asociada (otro objeto Java). La clave se utiliza posteriormente para recuperar el valor. En eXtreme Scale, una correlación consta de entradas de estos pares clave-valor.

WebSphere eXtreme Scale ofrece una serie de configuraciones de cuadrícula de datos, desde una única, una memoria caché local simple a una memoria caché distribuida grande, utilizando varias máquinas virtuales Java o servidores.

Además de almacenar objetos Java simples, puede almacenar objetos con relaciones. Puede utilizar un lenguaje de consulta que sea como SQL, con sentencias `SELECT ... FROM ... WHERE` para recuperar estos objetos. Por ejemplo, un objeto order (pedido) puede tener un objeto customer (cliente) y varios objetos item (artículo) asociados a éste. WebSphere eXtreme Scale soporta las relaciones de uno a uno, de uno a varios y de varios a varios.

WebSphere eXtreme Scale también soporta una interfaz de programación EntityManager, para almacenar entidades de la memoria caché. Esta interfaz de programación es similar a las entidades en Java Enterprise Edition. Las relaciones de entidad se pueden descubrir automáticamente desde un archivo XML de descriptor de entidades o anotaciones en las clases Java. Puede recuperar una entidad de la memoria caché mediante la clave primaria utilizando el método find en la interfaz EntityManager. Las entidades pueden persistir o eliminarse de la cuadrícula de datos dentro de un límite de transacción.

Considere un ejemplo distribuido donde la clave es un simple nombre alfabético. La memoria caché se podría dividir en cuatro particiones mediante clave: la partición 1 para las claves que empiezan por A-E, la partición 2 para las claves que empiezan por F-L, y así sucesivamente. Para la disponibilidad, una partición tiene un fragmento primario y un fragmento de réplica. Los cambios de los datos de la memoria caché se realizan en el fragmento primario, y se replican en el fragmento de réplica. Configura el número de servidores que contienen los datos de la cuadrícula de datos, y eXtreme Scale distribuye los datos en fragmentos entre estas instancias de servidor. Para la disponibilidad, los fragmentos de réplica se colocan en servidor físicos aparte de los fragmentos primarios.

WebSphere eXtreme Scale utiliza un servicio de catálogo para localizar el fragmento primario para cada clave. Maneja el traspaso de fragmentos entre servidores eXtreme Scale cuando los servidores físicos fallan y posteriormente se recuperan. Por ejemplo, si el servidor que contiene un fragmento de réplica falla, eXtreme Scale asigna un nuevo fragmento de réplica. Si un servidor que contiene un fragmento primario falla, el fragmento de réplica pasa a ser el fragmento primario. Como anteriormente, se construye un nuevo fragmento de réplica.

Novedades de la versión 7.1.1

WebSphere eXtreme Scale incluye muchas nuevas características en la Versión 7.1.1. Utilice este tema para obtener información sobre las actualizaciones más recientes del producto.

Plug-ins DataSerializer

Cuando los clientes y los servidores intercambian información o cuando los servidores replican datos de un servidor a otro, los datos se deben convertir o serializae, para poderse transmitir a través de la red. En releases anteriores, utilizaba la serialización predeterminada de Java o el plug-in ObjectTransformer para serializar los datos. En este release puede utilizar los plug-ins DataSerializer para describir eficazmente el formato de serialización, o matriz de bytes, a WebSphere eXtreme Scale de modo que el producto pueda interactuar con la matriz de bytes sin necesitar un formato de objeto específico. Más información...

Infraestructura OSGi

Mediante la infraestructura OSGi, puede exponer los plug-ins como servicios OSGi para que el tiempo de ejecución de eXtreme Scale los pueda utilizar. Además, puede iniciar los clientes y servidores eXtreme Scale en un contenedor OSGi, lo que le permite añadir y actualizar dinámicamente plug-ins eXtreme Scale en el entorno de ejecución. Más información...

Mejora del rendimiento del proveedor de memoria caché dinámica

Se ha mejorado el proceso de invalidación en el proveedor de memoria caché dinámica de WebSphere eXtreme Scale. Las solicitudes de invalidación se procesan asíncronamente y en lote cuando el parámetro **wait** del método invalidate(key, wait) se establece en un valor de `false`. Esto mejora considerablemente el rendimiento. Más información...

Modificación del comportamiento predeterminado de la colocación

En releases anteriores, cuando se iniciaba un nuevo servidor de contenedor en la cuadrícula de datos, la colocación de fragmentos en ese servidor de contenedor se iniciaba inmediatamente. Esta colocación inmediata resultaba en una utilización alta del procesador en los servidores que contenían los nuevos servidores de contenedor. El comportamiento predeterminado se ha modificado para establecer un retardo de 15000 milisegundos o 15 segundos antes de que se produzca la colocación. Puede cambiar el intervalo de colocación con la propiedad del servidor **placementDeferralInterval**. Más información...

Topología interna de dominio para configuraciones de plug-in de memoria caché de nivel 2 (L2) JPA (Java Persistence API)

Configurando una topología interna de dominio en la memoria caché L2 JPA, se coloca un fragmento primario en cada servidor de contenedor de la configuración. Cada fragmento primario contiene todo el contenido de la partición. Utilizando esta configuración, puede aumentar el rendimiento porque los clientes pueden acceder localmente a los datos y cualquier fragmento primario puede escribir en la cuadrícula de datos. Más información...

Programa de utilidad `xscmd`

El programa de utilidad **xscmd** es la nueva versión soportada del programa de utilidad **xsadmin**. El programa de utilidad **xsadmin** se incluía como un ejemplo no soportado en releases anteriores. Más información...

Herramienta para generar informes de análisis de registro

Con la nueva herramienta **xsloganalyzer** puede generar informes a partir de los archivos de registro que pueden ayudarle a analizar el rendimiento del entorno y solucionar los problemas. Más información...

IBM eXtremeIO e IBM eXtremeMemory

Configurando eXtremeMemory, puede almacenar objetos en memoria nativa en lugar de en un almacenamiento dinámico Java. La configuración de eXtremeMemory habilita eXtremeIO, un nuevo mecanismo de transporte. Si mueve

los objetos fuera del almacenamiento dinámico de Java, podrá evitar las pausas de recogida de basura, lo que hará que el rendimiento sea más constante y los tiempos de respuesta sean predecibles. Más información...

Soporte de WebSphere Application Server Versión 8

WebSphere eXtreme Scale Versión 7.1.1 se puede instalar ahora en WebSphere Application Server y WebSphere Application Server Network Deployment Versión 8. Más información...

Notas del release

Se proporcionan enlaces al sitio web de soporte del producto, a la documentación del producto y a las últimas actualizaciones, limitaciones y problemas conocidos.

- "Acceso a las actualizaciones, limitaciones y problemas conocidos más recientemente"
- "Acceso a los requisitos de software y del sistema"
- "Acceso a documentación del producto"
- "Acceso al sitio web de soporte de producto"
- "Cómo ponerse en contacto con el servicio de soporte de software de IBM" en la página 7

Acceso a las actualizaciones, limitaciones y problemas conocidos más recientemente

Las notas de release están disponibles en el sitio de soporte del producto como notas técnicas. Para ver una lista de todas las notas técnicas para WebSphere eXtreme Scale, vaya a la página web de soporte.Al pulsar los enlaces proporcionados aquí se realiza una búsqueda de las notas del release de interés en la página web de soporte, que se devolverán en una lista.

- 7.1.1+ Para ver una lista de las notas de release para la versión 7.1.1, vaya a la página web de soporte.
- Para ver una lista de las notas del release para la versión 7.1, visite la página web de soporte.
- Para ver una lista de las notas del release para la versión 7.0, visite la página web de soporte.
- Para ver una lista de las notas del release para la versión 6.1, vaya a la página wiki de las notas del release.

Acceso a los requisitos de software y del sistema

Los requisitos de hardware y software aparecen documentados en las páginas siguientes:

- Requisitos de sistema detallados

Acceso a documentación del producto

Para obtener todo el conjunto de información, vaya a la página de la biblioteca.

Acceso al sitio web de soporte de producto

Para buscar las últimas notas técnicas, las descargas, los arreglos e información adicional relacionada con el soporte, vaya al Portal de soporte.

Cómo ponerse en contacto con el servicio de soporte de software de IBM®

Si encuentra un problema con el producto, primero intente llevar a cabo las siguientes acciones:

- Siga los pasos descritos en la documentación del producto
- Busque la documentación relacionada en la ayuda en línea
- Busque los mensajes de error en la consulta de mensajes

Si no puede resolver el problema con ninguno de los métodos citados anteriormente, póngase en contacto con el servicio de IBM.

Requisitos de hardware y software

Examine una visión general de requisitos de hardware y de sistema operativo. Aunque no es necesario que utilice un nivel específico de hardware o sistema operativo para WebSphere eXtreme Scale, están disponibles opciones de hardware y software soportadas formalmente en la página Systems Requirements (Requisitos de sistema) del sitio de soporte del producto. Si existe un conflicto entre el Information Center y la página de requisitos de sistema, tiene prioridad la información del sitio web. La información de requisitos previos en el centro de información sólo se proporciona por comodidad.

Consulte la página Requisitos del sistema para ver el conjunto oficial de requisitos de hardware y software.

No es necesario que instale y despliegue eXtreme Scale en un nivel específico de sistema operativo. Cada instalación de Java Platform, Standard Edition (Java SE) y Java Platform, Enterprise Edition (Java EE) necesita diferentes niveles de sistema operativo o arreglos.

Puede instalar y desplegar el producto en los entornos de Java EE y Java SE. También puede empaquetar el componente de cliente con las aplicaciones Java EE directamente si integrarse con WebSphere Application Server. WebSphere eXtreme Scale soporta Java SE 5 o posteriores y WebSphere Application Server Versión 6.1 y posteriores.

Requisitos de hardware

WebSphere eXtreme Scale no requiere un nivel específico de hardware. Los requisitos de hardware dependen del hardware soportado para la instalación de Java Platform, Standard Edition que utiliza para ejecutar WebSphere eXtreme Scale. Si utiliza eXtreme Scale con WebSphere Application Server u otra implementación de Java Platform, Enterprise Edition, los requisitos de hardware de estas plataformas son suficientes para WebSphere eXtreme Scale.

Requisitos de sistema operativo

- **Sin la consola web**

  eXtreme Scale no requiere un nivel específico de sistema operativo. Cada implementación de Java SE y Java EE requiere niveles o arreglos distintos de sistema operativo para problemas que se han descubierto durante la comprobación de la implementación de Java. Los niveles necesarios para estas implementaciones son suficientes para eXtreme Scale.
- **Con la consola web**

Los requisitos siguientes se aplican a cada sistema operativo si se utiliza la consola:

- Linux: JVM de 32 bits o 64 bits
- Linux PPC: sólo JVM de 32 bits
- Windows: sólo JVM de 32 bits
- AIX: sólo JVM de 32 bits

Requisitos del navegador web

La consola web da soporte a los siguientes navegadores web:

- Mozilla Firefox, versión 3.5.x y posteriores
- Mozilla Firefox, versión 3.6.x y posteriores
- Microsoft Internet Explorer, versión 7 o 8

Requisitos de WebSphere Application Server

- WebSphere Application Server Versión 6.1.0.39 o posterior
- WebSphere Application Server Versión 7.0.0.19 o posterior
- WebSphere Application Server Versión 8.0.0.1 o posterior

Consulte los Arreglos recomendados para WebSphere Application Server si desea más información.

Requisitos de otros servidores de aplicaciones

Otras implementaciones de Java EE pueden utilizar el tiempo de ejecución de eXtreme Scale como una instancia local o como un cliente para los servidores eXtreme Scale. Para implementar Java SE, debe utilizar la versión 5 o posterior.

Convenios de directorio

Se utilizan los siguientes convenios de directorio en toda la documentación para hacer referencia a directorios como por ejemplo *raíz_instalación_wxs* e *inicio_wxs*. Accede a estos directorios durante distintos escenarios, incluido durante la instalación y la utilización de las herramientas de línea de mandatos.

raíz_intal_wxs
El directorio *raíz_instalación_wxs* es el directorio raíz donde se instalan los archivos del producto WebSphere eXtreme Scale. El directorio *raíz_instalación_wxs* puede ser el directorio en el que se extrae el archivado de prueba o el directorio en el que se instala el producto WebSphere eXtreme Scale.

- Ejemplo al extraer la prueba:

  Ejemplo: `/opt/IBM/WebSphere/eXtremeScale`
- Ejemplo cuando se instala WebSphere eXtreme Scale en un directorio autónomo:

  Ejemplo: `/opt/IBM/eXtremeScale`
- Ejemplo cuando se integra WebSphere eXtreme Scale con WebSphere Application Server:

  Ejemplo: `/opt/IBM/WebSphere/AppServer`

inicio_wxs
El directorio *inicio_wxs* es el directorio raíz de los componentes, ejemplos y bibliotecas del producto WebSphere eXtreme Scale. Este directorio es el mismo

que el directorio *raíz_instalación_wxs* cuando se ha extraído la versión de prueba. Para instalaciones autónomas, el directorio *inicio_wxs* es el subdirectorio `ObjectGrid` del directorio *raíz_instalación_wxs*. Para instalaciones integradas con WebSphere Application Server, este directorio es el directorio `optionalLibraries/ObjectGrid` del directorio *raíz_instalación_wxs*.

- Ejemplo al extraer la prueba:

  Ejemplo: `/opt/IBM/WebSphere/eXtremeScale`
- Ejemplo cuando se instala WebSphere eXtreme Scale en un directorio autónomo:

  Ejemplo: `/opt/IBM/eXtremeScale/ObjectGrid`
- Ejemplo cuando se integra WebSphere eXtreme Scale con WebSphere Application Server:

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

raíz_was
El directorio *raíz_was* es el directorio raíz de una instalación de WebSphere Application Server:

Ejemplo: `/opt/IBM/WebSphere/AppServer`

inicio_servicioRest
El directorio *inicio_servicioRest* es el directorio en el que se encuentran las bibliotecas y los ejemplos del servicio de datos REST de WebSphere eXtreme Scale. Este directorio se denomina `restservice` y es un subdirectorio del directorio *inicio_wxs*.

- Ejemplo para despliegues autónomos:

  Ejemplo: `/opt/IBM/WebSphere/eXtremeScale/ObjectGrid/restservice`
- Ejemplo para despliegues integrados de WebSphere Application Server:

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

raíz_tomcat
raíz_tomcat es el directorio raíz de la instalación de Apache Tomcat.

Ejemplo: `/opt/tomcat5.5`

raíz_wasce
raíz_wasce es el directorio raíz de la instalación de WebSphere Application Server Community Edition.

Ejemplo: `/opt/IBM/WebSphere/AppServerCE`

inicio_java
inicio_java es el directorio raíz de una instalación de Java Runtime Environment (JRE).

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

inicio_samples
inicio_samples es el directorio en el que extrae los archivos de ejemplo que se utilizan para las guías de aprendizaje.

Ejemplo: `/wxs-samples/`

raíz_dvd
El directorio *raíz_dvd* es el directorio raíz del DVD que contiene el producto.

Ejemplo: `raíz_dvd/docs/`

raíz_equinox
El directorio *raíz_equinox* es el directorio raíz de la instalación de infraestructura OSGi de Eclipse Equinox.

Ejemplo:`/opt/equinox`

inicio_usuario
El directorio *inicio_usuario* es la ubicación donde se almacenan los archivos de usuario, por ejemplo los perfiles de seguridad.

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

UNIX `/home/`*`nombre_usuario`*

Visión general técnica de WebSphere eXtreme Scale

WebSphere eXtreme Scale es una cuadrícula de datos elástica, escalable y en memoria. Coloca en la memoria caché, realiza particiones y réplicas y gestiona de forma dinámica los datos de aplicaciones y la lógica empresarial entre varios servidores.

Puesto que WebSphere eXtreme Scale no es una base de datos en memoria, debe considerar los requisitos de configuración específicos. El primer paso para desplegar una cuadrícula de datos es iniciar un grupo principal y un servicio de catálogo, que actuará como coordinador para todas las demás máquinas virtuales Java que participan en la cuadrícula de datos y gestionan la información de configuración. Los procesos de WebSphere eXtreme Scale se inician con unas sencillas invocaciones de script de mandato desde la línea de mandatos.

El paso siguiente es iniciar los procesos de servidor WebSphere eXtreme Scale para la cuadrícula de datos para almacenar y recuperar datos. Cuando se inician los servidores, estos se registran automáticamente en el grupo principal y el servicio de catálogo, lo que les permite cooperar en el suministro de servicios de cuadrícula de datos. Más servidores aumentan tanto la capacidad como la fiabilidad de la cuadrícula de datos.

Una cuadrícula de datos local es una cuadrícula sencilla y de instancia única donde todos los datos están en esta cuadrícula. Para utilizar eficazmente WebSphere eXtreme Scale como un espacio de proceso de base de datos en memoria, puede configurar y desplegar una cuadrícula de datos distribuida. Los datos de la cuadrícula distribuida se extienden sobre los distintos servidores eXtreme Scale que los contienen, de forma que cada servidor sólo contiene algunos de los datos, llamados partición.

Un parámetro de configuración de cuadrícula de datos distribuida de claves es el número de particiones de la cuadrícula. Los datos de cuadrícula se particionan en este número de subconjuntos, cada uno de los cuales recibe el nombre de partición. El servicio de catálogo localiza la partición para un dato determinado basado en su clave. El número de particiones afecta directamente a la capacidad y escalabilidad de la cuadrícula de datos. Un servidor puede contener una o más particiones de cuadrícula de datos. Por lo tanto, el espacio de la memoria del servidor limita el tamaño de una partición. Por el contrario, aumentar el número de particiones aumenta la capacidad de la cuadrícula de datos. La capacidad máxima de la cuadrícula de datos es el número de particiones por el tamaño de memoria utilizable de cada servidor. Un servidor puede ser una JVM, pero se puede definir el servidor eXtreme Scale para que se ajuste a su entorno de despliegue.

Los datos de una partición se almacenan en un fragmento. Para la disponibilidad, una cuadrícula de datos se puede configurar con réplicas, que pueden ser síncronas o asíncronas. Los cambios en los datos de cuadrícula se realizan en el fragmento primario y se duplican en los fragmentos duplicados. La memoria total que utiliza o requiere la cuadrícula de datos es por lo tanto el tamaño de la cuadrícula de datos por (1 (para el primario) + el número de réplicas).

WebSphere eXtreme Scale distribuye los fragmentos de una cuadrícula de datos entre el número de servidores que forman la cuadrícula. Estos servidores pueden estar en el mismo servidor físico o en servidores físicos distintos. Para la disponibilidad, los fragmentos de réplica se colocan en servidor físicos aparte de los fragmentos primarios.

WebSphere eXtreme Scale supervisa el estado de sus servidores y mueve los fragmentos durante la anomalía o recuperación de servidores físicos o fragmentos. Por ejemplo, si el servidor que contiene un fragmento de réplica falla, WebSphere eXtreme Scale asigna un nuevo fragmento de réplica, y replica los datos del fragmento primario a la nueva réplica. Si un servidor contiene un fragmento primario faisl, el fragmento de réplica pasa a ser el fragmento primario y se construye un nuevo fragmento de réplica. Si inicia un servidor adicional para la cuadrícula de datos, los fragmentos se equilibran entre todos los servidores. Este reequilibrado se denomina escalado horizontal. De forma similar, para escalado vertical, podría detener uno de los servidores para reducir los recursos utilizados por una cuadrícula de datos. Como resultado, los fragmentos se equilibran en los servidores restantes.

Visión general de memoria caché

WebSphere eXtreme Scale puede funcionar como un espacio de proceso de base de datos en memoria, que puede utilizar para proporcionar el almacenamiento en memoria caché en línea para un programa de fondo de la base de datos o para funcionar como una memoria caché secundaria. El almacenamiento en memoria caché en línea utiliza eXtreme Scale como el medio principal para interactuar con los datos. Cuando eXtreme Scale se utiliza como memoria caché complementaria, el programa de fondo se utiliza junto con la cuadrícula de datos. En esta sección se describen distintos conceptos y escenarios de almacenamiento en memoria caché y se explican las topologías disponibles para desplegar una cuadrícula de datos.

Arquitectura de memoria caché: correlaciones, contenedores, clientes y catálogos

Con WebSphere eXtreme Scale, la arquitectura puede utilizar el almacenamiento en memoria caché de datos en memoria local o el almacenamiento en memoria caché de datos de cliente-servidor distribuido.

Para poder funcionar, WebSphere eXtreme Scale necesita una mínima infraestructura adicional. La infraestructura se compone de scripts que instalan, inician y detienen una aplicación Java Platform, Enterprise Edition en un servidor. Los datos colocados en memoria caché se almacenan en el servidor eXtreme Scale, y los clientes se conectan de forma remota al servidor.

Las memorias caché distribuidas ofrecen un mejor rendimiento, disponibilidad y escalabilidad y se puede configurar mediante topologías dinámicas, en los que los servidores se equilibran automáticamente. También puede añadir servidores

adicionales sin reiniciar los servidores eXtreme Scale existentes. Puede crear despliegues sencillos o grandes despliegues con terabytes en los que son necesarios miles de servidores.

Servicio de catálogo

El servicio de catálogo alberga lógica que debería estar inactiva durante un estado fijo y tiene poca influencia en escalabilidad. El servicio de catálogo se crea para dar servicio a cientos de contenedores que pasan a estar disponibles de forma simultánea y servicios de ejecuciones para gestionar los contenedores.

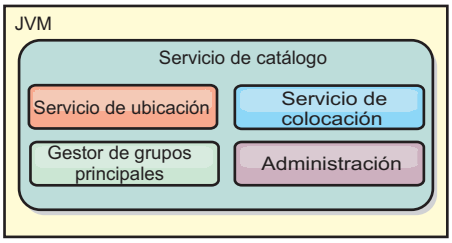


Figura 2. Servicio de catálogo

El catálogo se ocupa de los servicios siguientes:

Servicio de ubicación
El servicio de ubicación proporciona localidad para los clientes que buscan contenedores que alojan aplicaciones y contenedores que buscan registrar aplicaciones alojadas con el servicio de colocación. El servicio de ubicación se ejecuta en todos los miembros de la cuadrícula para escalar hacia fuera esta función.

Servicio de colocación
El servicio de colocación es el sistema nervioso central de la cuadrícula y es el responsable de asignar fragmentos individuales al contenedor host. El servicio de colocación se ejecuta como uno de los N servicios elegidos en el clúster. Dado que se utiliza la política uno de N, siempre hay exactamente una instancia del servicio de colocación en ejecución. Si se detuviera esa instancia, tomaría el control otro proceso. Todos los estados del servicio de catálogo se replican en todos los servidores que alojan el servicio de catálogo para favorecer la redundancia.

Gestor de grupos principales
El gestor de grupos principales gestiona el agrupamiento de igual para la supervisión de salud, organiza los contenedores en pequeños grupos de servidores y federa automáticamente los grupos de servidores. Cuando un contenedor se pone en contacto en primer lugar con el servicio de catálogo, el contenedor espera a ser asignado a un grupo nuevo o existente de varias máquinas virtuales Java (JVM). Cada grupo de máquinas virtuales Java supervisa la disponibilidad de cada uno de sus miembros a través de las pulsaciones. Uno de los miembros del grupo transmite información de disponibilidad del servicio de catálogo para reaccionar ante anomalías mediante la reasignación y el reenvío de rutas.

Administración
Las cuatro fases que componen la administración del entorno de WebSphere eXtreme Scale son planificación, despliegue, gestión y supervisión.

Para favorecer la disponibilidad, configure un dominio de servicio de catálogo. Un dominio de servicio de catálogo está formado por varias máquinas virtuales Java, incluida una JVM maestra y una serie de máquinas virtuales Java de copia de seguridad.

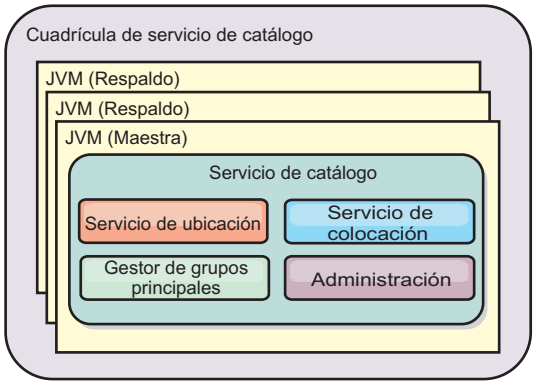


Figura 3. Dominio de servicio de catálogo

Servidores de contenedor, particiones y fragmentos

El servidor de contenedor almacena los datos de la aplicación para la cuadrícula de datos. Estos datos se dividen normalmente en partes, denominadas particiones, que alojan varios servidores de contenedor. Cada servidor de contenedor a su vez aloja un subconjunto de los datos completos. Una JVM puede alojar uno o más contenedores y cada servidor de contenedor puede alojar varios fragmentos.

Recuerde: Planifique el tamaño de almacenamiento dinámico de los servidores de contenedor, que alojan todos los datos. Configure los valores de almacenamiento dinámico según corresponda.

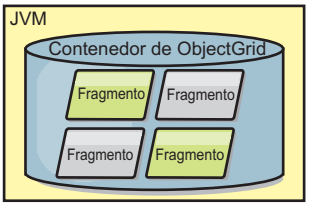


Figura 4. Servidor de contenedor

Las particiones alojan un subconjunto de los datos en la cuadrícula. WebSphere eXtreme Scale coloca automáticamente varias particiones en un único servidor de contenedor y propaga las particiones a medida que pasan a estar disponibles más servidores de contenedor.

Importante: Elija cuidadosamente el número de particiones antes del despliegue final, ya que el número de particiones no se puede cambiar de forma dinámica. Se utiliza un mecanismo hash para localizar las particiones en la red y eXtreme Scale no puede volver a realizar hash de todo el conjunto de datos, una vez que se haya desplegado. Como regla general, se puede sobrestimar el número de particiones

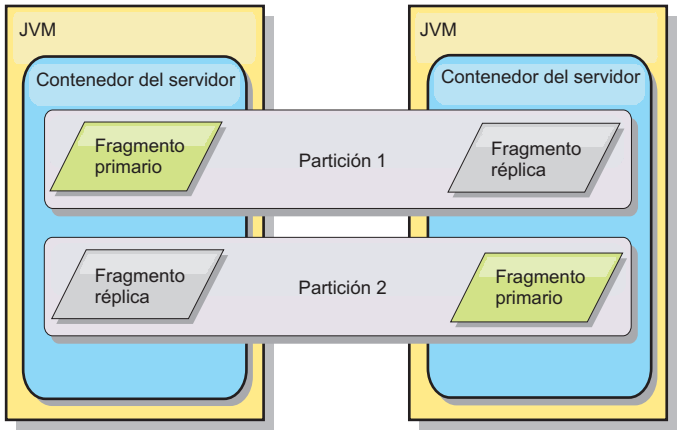


Figura 5. Partición

Los fragmentos son instancias de particiones y tienen uno de los roles siguientes: primario o réplica. El fragmento primario y sus réplicas conforman la manifestación física de la partición. Cada partición tiene varios fragmentos que cada uno de éstos incluye todos los datos contenidos en dicha partición. Un fragmento es el primario y las demás son réplicas, que son copias redundantes de los datos del fragmento primario. Un fragmento primario es la única instancia de partición que permite que las transacciones se graben en la memoria caché. Un fragmento réplica es una instancia "duplicada" de la partición. Recibe actualizaciones de forma síncrona o asíncrona del fragmento primario. El fragmento réplica sólo permite a las transacciones leer de la memoria caché. Las réplicas nunca se alojan en el mismo servidor de contenedor como fragmento primario y normalmente no se alojan en la misma máquina que el fragmento primario.

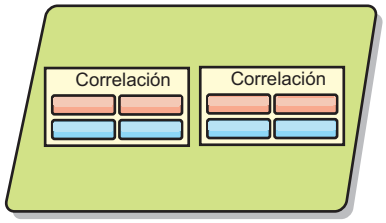


Figura 6. Fragmento

Para aumentar la disponibilidad de los datos, o para aumentar las garantías de persistencia, replique los datos. No obstante, la réplica supone coste en la transacción y cambia rendimiento por disponibilidad. Con eXtreme Scale, puede controlar el coste, ya que se admiten la réplica síncrona y asíncrona, así como modelos de réplica híbridos que usan modalidades de réplica síncrona y asíncrona. Un fragmento réplica síncrona recibe actualizaciones como parte de la transacción del fragmento primario para garantizar la coherencia de los datos. Una réplica síncrona puede doblar el tiempo de respuesta porque la transacción debe confirmar en el fragmento primario y la réplica síncrona antes de que se complete la transacción. Un fragmento réplica asíncrona recibe actualizaciones después de la confirmación de la transacción para limitar el impacto en el rendimiento, pero puede haber pérdida de datos ya que la réplica asíncrona puede estar varias transacciones por detrás del fragmento primario.

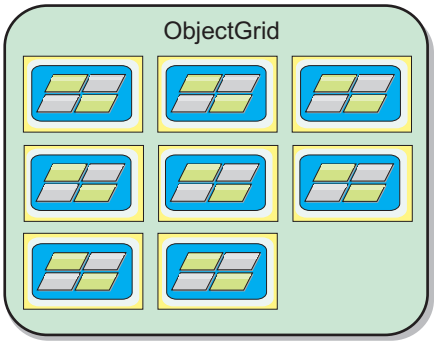


Figura 7. ObjectGrid

Correlaciones

Una correlación es un contenedor de pares de clave-valor, que permite a una aplicación almacenar un valor indexado por una clave. Las correlaciones soportan los índices que se pueden añadir a atributos de índice en la clave o el valor. Estos índices son utilizados automáticamente por el tiempo de ejecución de la consulta para determinar el método más eficaz para ejecutar una consulta.

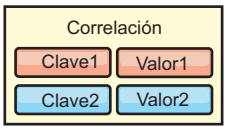


Figura 8. Correlación

Un conjunto de correlaciones es una colección de correlaciones con un algoritmo de particionamiento común. Los datos de las correlaciones se replican en función de la política definida en el conjunto de correlaciones. Un conjunto de relaciones sólo se utiliza en topologías distribuidas y no es necesario en topologías locales.

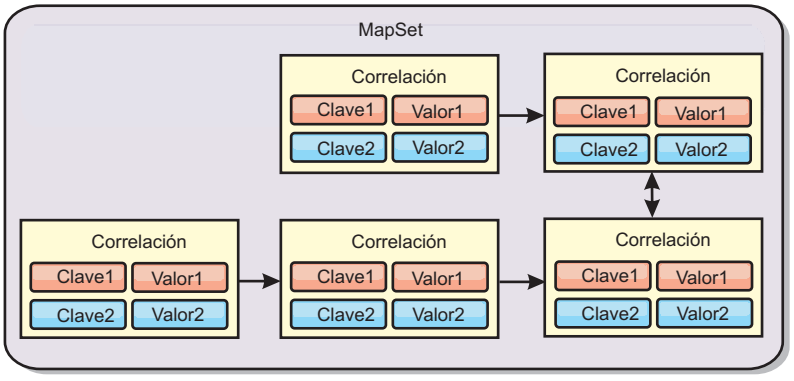


Figura 9. Conjuntos de correlaciones

Un conjunto de correlaciones puede tener asociado un esquema. Un esquema son los metadatos que describen las relaciones entre cada correlación, cuando se utilizan tipos Object homogéneos, o entidad.

WebSphere eXtreme Scale puede almacenar objetos Java serializables en cada una de las correlaciones utilizando la API ObjectMap. Un esquema se puede definir en las correlaciones para identificar la relación entre los objetos de las correlaciones donde cada correlación incluye objetos de un único tipo. La definición de un esquema para las correlaciones es obligatoria para consultar el contenido de los objetos de la correlación. WebSphere eXtreme Scale puede tener varios esquemas de correlación definidos.

WebSphere eXtreme Scale también puede almacenar entidades mediante la API EntityManager. Cada entidad se asocia con una correlación. El esquema de un conjunto de correlaciones de entidad se descubre automáticamente utilizando un archivo XML de descriptor de entidad o clases Java anotadas. Cada entidad tiene un conjunto de atributos clave y un conjunto de atributos no clave. Una entidad también puede tener relaciones con otras entidades. WebSphere eXtreme Scale admite relaciones de una a una, de una a varias, de varias a una, y de varias a varias. Cada entidad se correlaciona físicamente con una única correlación del conjunto de correlaciones. Las entidades permiten que las aplicaciones tengan fácilmente gráficos de objetos complejos que abarquen varias correlaciones. Una topología distribuida puede tener varios esquemas de entidad.

Para obtener más información, consulte Almacenamiento en memoria caché de objetos y sus relaciones (API EntityManager).

Clientes

Los clientes se conectan a un servicio de catálogo, recuperan una descripción de la topología del servidor y se comunican directamente con cada servidor según precisen. Cuando la topología del servidor cambia porque se añaden servidores nuevos o porque han fallado servidores existentes, el servicio de catálogo dinámico direcciona el cliente al servidor apropiado que alberga los datos. Los clientes deben examinar las claves de los datos de la aplicación para determinar a qué partición direccionar la petición. Los clientes pueden leer los datos de diversas particiones en una única transacción. No obstante, los clientes pueden actualizar sólo una única partición en una transacción. Después de que el cliente actualice algunas entradas, la transacción del cliente debe usar esa partición para las actualizaciones.

Las combinaciones posibles de despliegue son las siguientes:

- Existe un servicio de catálogo en su propia cuadrícula de máquinas virtuales Java. Se puede utilizar un único servicio de catálogo para gestionar diversos clientes o servidores de eXtreme Scale.
- Se puede iniciar un contenedor en una JVM por sí mismo o se puede cargar en una JVM arbitraria con otros contenedores para instancias de ObjectGrid distintas.
- Un cliente puede existir en cualquier JVM y comunicarse con una o más instancias de ObjectGrid. También puede existir un cliente en la misma JVM que la de un contenedor.

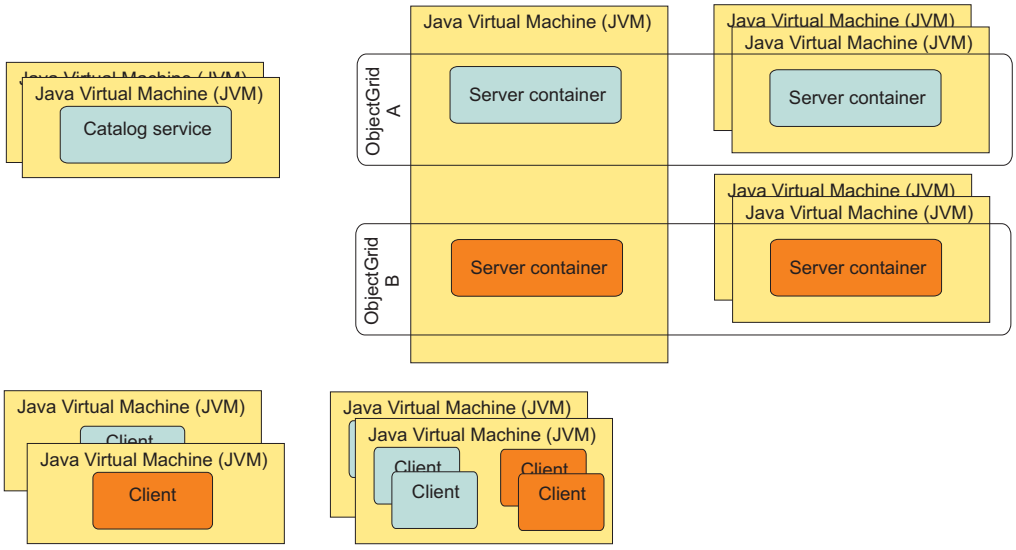


Figura 10. Topologías posibles

Zonas

Las zonas le proporcionan control sobre la colocación de fragmentos. Las zonas son agrupaciones lógicas definidas por el usuario de servidores físicos. A continuación se muestran ejemplos de distintos tipos de zonas: distintos servidores blade, chasis de servidores blade, plantas de un edificio, edificios o distintas ubicaciones geográficas en un entorno de varios centros de datos. Otro caso de uso es en un entorno virtualizado donde muchas instancias de servidor, cada una de ellas con una dirección IP exclusiva, se ejecutan en el mismo servidor físico.

Zonas definidas entre centros de datos

El ejemplo y caso de uso clásico de las zonas es cuando se tienen dos centros de datos geográficamente dispersos. Los centros de datos dispersos distribuyen la cuadrícula de datos entre distintas ubicaciones de recuperación de anomalía de centro de datos. Por ejemplo, es posible que desee asegurarse de que tiene un conjunto completo de fragmentos de réplica asíncrona para la cuadrícula de datos en un centro de datos remoto. Con esta estrategia, puede realizar la recuperación de la anomalía del centro de datos local de forma transparente, sin pérdida de datos. Los propios centros de datos tienen redes de latencia baja y velocidad alta. Sin embargo, la comunicación entre un centro de datos y otro tiene latencia más alta. Las réplicas síncronas se utilizan en cada centro de datos donde la baja latencia minimiza el impacto de la réplica en los tiempos de respuesta. La utilización de réplica asíncrona reduce el impacto sobre el tiempo de respuesta. La distancia geográfica proporciona disponibilidad en caso de anomalía de centro de datos local.

En el ejemplo siguiente, los fragmentos primarios de la zona de Chicago tienen réplicas en la zona de Londres. Los fragmentos primarios de la zona de Londres tienen réplicas en la zona de Chicago.

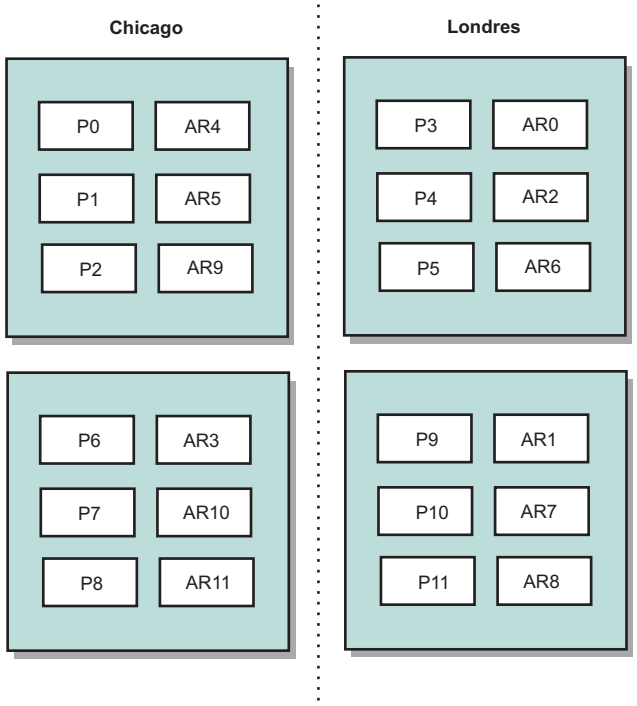


Figura 11. Primarios y réplicas en las zonas

Tres valores de configuración de eXtreme Scale controlan la colocación de fragmentos:

- Establecer el archivo de despliegue
- Agrupar contenedores
- Especificar reglas

Las secciones siguientes explican las distintas opciones, presentadas de forma flexible en orden de complejidad creciente.

Inhabilitar modalidad de desarrollo

En el archivo XML de despliegue, establezca: `developmentMode="false"`.

Este sencillo paso activa la primera política de colocación de fragmentos de eXtreme Scale.

Para obtener más información sobre el archivo XML, consulte Archivo XML de descriptor de política de despliegue .

Política 1: los fragmentos de la misma partición se colocan en servidores físicos distintos.

Considere un ejemplo sencillo de una cuadrícula de datos con un fragmento de réplica. Con esta política, los fragmentos primario y de réplica de cada partición se encuentran en distintos servidores físicos. Si falla un único servidor físico, no se producirá pérdida de datos. Los fragmentos primario o de réplica de cada partición se encuentran en servidores físicos distintos que no han fallado, o ambos se encuentran en algún otro servidor físico que no ha fallado.

La alta disponibilidad y simplicidad de esta política la convierte en el valor más eficaz para todos los entornos de producción. En muchos casos, la aplicación de esta política es el único paso necesario para controlar de forma efectiva la colocación de fragmentos en el entorno.

Al aplicar esta política, se define un servidor físico mediante una dirección IP. Los fragmentos se colocan en servidores de contenedor. Los servidores de contenedor tienen una dirección IP, por ejemplo, el parámetro **-listenerHost** en el script **startOgServer**. Varios servidores de contenedor pueden tener la misma dirección IP.

Dado que un servidor físico tiene varias direcciones IP, tenga en cuenta el paso siguiente para tener más control del entorno.

Definir zonas para agrupar servidores de contenedor

Los servidores de contenedor se asignan a zonas con el parámetro **-zone** en el script startOgServer. En un entorno de WebSphere Application Server, las zonas se definen mediante grupos de nodos con un formato de nombre específico: `ReplicationZone<Zona>`. De esta forma, puede elegir el nombre y la pertenencia de las zonas. Para obtener más información, consulte Definición de zonas para servidores de contenedor.

Política 2: se colocan fragmentos de la misma partición en zonas distintas.

Considere ampliar el ejemplo de una cuadrícula de datos con un fragmento de réplica desplegándolo en dos centros de datos. Defina cada centro de datos como una zona independiente. Utilice un nombre de zona de DC1 para los servidores de contenedor en el primer centro de datos, y de DC2 para los servidores de contenedor en el segundo centro de datos. Con esta política, los fragmentos primario y de réplica de cada partición estarían en distintos centros de datos. Si un centro de datos falla, no se producirá pérdida de datos. Para cada partición, su fragmento primario o de réplica está en el otro centro de datos.

Con esta política, puede controlar la colocación de fragmentos definiendo zonas. Puede elegir la agrupación o el límite físico o lógico que le interese. A continuación, elija un nombre de zona exclusivo para cada grupo, e inicie los servidores de contenedor en cada una de las zonas con el nombre de la zona correspondiente. Por lo tanto, eXtreme Scale coloca fragmentos de forma que los fragmentos de la misma partición se coloquen en zonas distintas.

Especificar reglas de zona

El nivel más preciso de control sobre la colocación de fragmentos se consigue utilizando reglas de zona. Las reglas de zona se especifican en el elemento zoneMetadata del XML de descriptor de política de despliegue de eXtreme Scale. Una regla de zona define un conjunto de zonas en el que se colocan los fragmentos. Un elemento shardMapping asigna un fragmento a una regla de zona. El atributo de fragmento del elemento shardMapping especifica el tipo de fragmento:

- P especifica el fragmento primario
- S especifica fragmentos de réplica síncrona
- A especifica fragmentos de réplica asíncrona.

Si existe más de una réplica síncrona o asíncrona, debe proporcionar elementos shardMapping del tipo de fragmento correspondiente. El atributo

exclusivePlacement del elemento zoneRule determina la colocación de fragmentos en la misma partición en zonas. Los valores del atributo exclusivePlacement son los siguientes:

- `true` (un fragmento no se puede colocar en la misma zona que otro fragmento de la misma partición).

  Recuerde: En el caso "true", debe tener como mínimo tantas zonas en la regla como fragmentos que la utilicen. De esta forma se asegura de que cada fragmento puede estar en su propia zona.
- `false` (los fragmentos de la misma partición se pueden colocar en la misma zona.

El valor predeterminado es true.

Para obtener más información, consulte Ejemplo: Definiciones de zona en el archivo XML de descriptor de política de despliegue.

Casos de uso ampliados

A continuación se muestran varios casos de uso de estrategias de colocación de fragmentos:

Actualizaciones rotativas

Considere un escenario en el que desea aplicar actualizaciones rotativas en los servidores físicos, incluido mantenimiento que requiere reinicio del despliegue. En este ejemplo, supongamos que tiene una cuadrícula de datos que se distribuye en 20 servidores físicos, definidos con una réplica síncrona. Desea concluir 10 de los servidores físicos simultáneamente para el mantenimiento.

Cuando concluye grupos de 10 servidores físicos, ninguna partición tiene simultáneamente sus fragmentos primario y de réplica en los servidores que está concluyendo. De lo contrario, perderá los datos de esa partición.

La solución más fácil es definir una tercera zona. En lugar de dos zonas de 10 servidores físicos, utilice tres zonas, dos con siete servidores físicos y una con seis. La distribución de los datos en más zonas permite una mejor migración tras error para la disponibilidad.

En lugar de definir otra zona, el otro enfoque es añadir una réplica.

Actualización de eXtreme Scale

Cuando se actualiza el software de eXtreme Scale de forma rotativa con cuadrículas de datos que contienen datos activos, tenga en cuenta los elementos siguientes. La versión del software del servicio de catálogo debe ser mayor o igual que las versiones del software del servidor de contenedor. En primer lugar, actualice todos los servidores de catálogo con una estrategia rotativa. Lea más sobre la actualización del despliegue en el tema Actualización de servidores eXtreme Scale.

Modificación del modelo de datos

Un problema relacionado es cómo modificar el modelo de datos o esquema de objetos almacenados en la cuadrícula de datos sin causar tiempo de inactividad. Causaría una interrupción modificar el modelo de datos deteniendo la cuadrícula

de datos y reiniciando con las clases de modelo de datos actualizadas en la classpath del servidor de contenedor y recargando la cuadrícula de datos. Una alternativa sería iniciar una nueva cuadrícula de datos con el nuevo esquema, copiar los datos de la cuadrícula de datos antigua a la nueva cuadrícula de datos y a continuación concluir la cuadrícula de datos antigua.

Cada uno de estos procesos causa interrupción y produce tiempo de inactividad. Para modificar el modelo de datos sin tiempo de inactividad, almacene el objeto en uno de los formatos siguientes:

- Utilice XML como valor
- Utilice un blob creado con Google protobuf
- Utilice JavaScript Object Notation (JSON)

Escriba serializadores para pasar de un POJO (plain old Java object) a uno de estos formatos fácilmente en el lado del cliente. Los cambios de esquema serán más fáciles.

Virtualización

Cloud computing y la virtualización son populares tecnologías emergentes. De forma predeterminada, eXtreme Scale asegura que dos fragmentos de la misma partición nunca se colocan en la misma dirección IP, tal como se describe en la Política 1. Cuando realiza el despliegue en imágenes virtuales, como por ejemplo VMware, muchas instancias de servidor, cada una con una dirección IP exclusiva, se pueden ejecutar en el mismo servidor físico. Para asegurarse de que las réplicas solo se puedan colocar en servidores físicos distintos, puede utilizar zonas para solucionar el problema. Agrupe los servidores físicos en zonas y utilice reglas de colocación para mantener fragmentos primarios y de réplica en zonas distintas.

Zonas para redes de área amplia

Es posible que desee desplegar una única cuadrícula de datos de eXtreme Scale en varios edificios o centros de datos con conexiones de red más lentas. Las conexiones de red más lentas llevan a un ancho de banda más bajo y a conexiones de latencia más alta. La posibilidad de particiones de red también aumenta en esta modalidad debido a la congestión de la red y otros factores.

Para tratar estos riesgos, el servicio de catálogo de eXtreme Scale organiza los servidores de contenedor en grupos principales que intercambian pulsaciones para detectar anomalías de servidor de contenedor. Estos grupos principales no se distribuyen en varias zonas. Un líder de cada grupo principal envía la información de pertenencia al servicio de catálogo. El servicio de catálogo comprueba si hay anomalías notificadas antes de responder a la información de pertenencia mediante las pulsaciones al servidor de contenedor en cuestión. Si el servicio de catálogo ve una detección de anomalía falsa, este no realiza ninguna acción. La partición de grupo principal se recupera rápidamente. El servicio de catálogo también envía pulsaciones a líderes de grupo principal periódicamente a velocidad lenta para manejar el caso de aislamiento de grupo principal.

Desalojadores

Los desalojadores eliminan datos de la cuadrícula de datos. Puede establecer un desalojador basado en el tiempo o, debido a que los desalojadores están asociados con BackingMaps, utilizar la interfaz BackingMap para especificar el desalojador conectable.

Desalojador predeterminado

Se crea un desalojador TTL predeterminado con cada correlación de respaldo. El desalojador predeterminado elimina las entradas en función de un concepto de tiempo de vida. Este comportamiento lo define el atributo ttlType, que tiene los tipos siguientes:

Ninguno

Especifica que las entradas nunca caducan y, por lo tanto, nunca se eliminan de la correlación.

Hora de creación (CREATION_TIME)

Especifica que las entradas se desalojan en función de cuándo se crearon.

Si utiliza el ttlType CREATION_TIME, el desalojador desaloja una entrada cuando su hora de creación es igual a su valor de atributo TimeToLive, que se establece en milisegundos en la configuración de la aplicación. Si estableció el valor del atributo TimeToLive en 10 segundos, la entrada se desalojará automáticamente 10 segundos después de su inserción.

Es importante establecer con cuidado este valor de CREATION_TIME. El desalojador funciona mejor cuando existen cantidades razonablemente altas de adiciones a la memoria caché que sólo se usan durante un tiempo establecido. Con esta estrategia, todo lo que se crea se eliminará después de transcurrido un tiempo establecido.

El ttlType CREATION_TIME resulta de utilidad en casos como renovar la cotización de bolsa cada 20 minutos o menos. Suponga que una aplicación web obtiene cotizaciones de bolsa. La obtención de las cotizaciones más recientes no es clave. En este caso, las cotizaciones de bolsa se almacenan en la memoria caché en ObjectGrid durante 20 minutos. Transcurridos los 20 minutos, las entradas de la correlación ObjectGrid caducan y se desalojan. Cada 20 minutos aproximadamente, la correlación ObjectGrid utiliza el plug-in Loader para renovar los datos de la correlación con datos renovados de la base de datos. La base de datos se actualiza cada 20 minutos con las cotizaciones de bolsa más recientes.

Hora del último acceso (LAST_ACCESED_TIME)

Especifica que las entradas se desalojan en función de la última vez que se accedió a ellas, ya sea porque se leyeron o se actualizaron.

Última actualización (LAST_UPDATE_TIME)

Especifica que las entradas se desalojan en función de cuándo se actualizaron por última vez.

Si utiliza LAST_UPDATE_TIME ttlType, establezca el valor de TimeToLive en un número más bajo que si estuviera utilizando CREATION_TIME, porque el valor de TimeToLive de las entradas se restablece cada vez que se obtiene acceso. En otras palabras, si el atributo TimeToLive es igual a 15 y una entrada ha existido 14 segundos y, acto seguido, se accede a ella, no caduca hasta transcurridos otros 15 segundos. Si establece TimeToLive en un número relativamente alto, puede que muchas entradas nunca se desalojen. No obstante, si establece el valor en un número aproximado a 15 segundos, las entradas se eliminarán si no se accede a ellas con mucha frecuencia.

Los ttlType LAST_ACCESS_TIME o LAST_UPDATE_TIME resultan de utilidad en casos como mantener los datos de sesión de un cliente, con una

correlación ObjectGrid. Los datos de sesión deben destruirse si el cliente no utiliza los datos de sesión durante un período de tiempo. Por ejemplo, los datos de sesión exceden el tiempo de espera si después de 30 minutos no se produce actividad en el cliente. En este caso, el uso de un tipo TTL de LAST_ACCESS_TIME o LAST_UPDATE_TIME con el atributo TimeToLive establecido en 30 minutos resulta adecuado para esta aplicación.

También puede escribir sus propios desalojadores. Para obtener más información, consulte Cómo escribir un desalojador personalizado .

Desalojador conectable

El desalojador TTL predeterminado utiliza una política de desalojo basada en la hora, y el número de entradas en BackingMap no tiene ningún efecto en la hora de caducidad de una entrada. Puede utilizar un desalojo conectable opcional para desalojar entradas basándose en el número de entradas que existían en lugar de basarse en la hora.

Los siguientes desalojadores conectables opcionales proporcionan algunos algoritmos utilizados habitualmente para decidir qué entradas se van a desalojar cuando una BackingMap aumenta de tamaño por encima de ciertos límites.

- LRUEvictor es un desalojador que utiliza un algoritmo de menos usada recientemente (LRU) para decidir qué entradas se van a desalojar cuando la BackingMap exceda un número máximo de entradas.
- LFUEvictor es un desalojador que utiliza un algoritmo de usada menos frecuentemente (LFU) para decidir qué entradas se van a desalojar cuando la BackingMap exceda un número máximo de entradas.

BackingMap informa a un desalojador de la creación, modificación o eliminación de entradas en una transacción. BackingMap hace un seguimiento de estas entradas y elige cuándo desalojar una o más entradas de la instancia de BackingMap.

Una instancia de BackingMap no tiene información de configuración para un tamaño máximo. Por el contrario, las propiedades de desalojador se establecen para controlar el comportamiento del desalojador. Tanto LRUEvictor como LFUEvictor tienen una propiedad de tamaño máximo que se utiliza para que el desalojador empiece a desalojar entradas después de que se supere el tamaño máximo. Como el desalojador TTL, es posible que los desalojadores LRU y LFU no desalojen inmediatamente una entrada cuando se alcance el número máximo de entradas para minimizar el impacto en el rendimiento.

Si los algoritmos de desalojo LRU o LFU no son adecuados para una determinada aplicación, puede escribir sus propios desalojadores para crear la estrategia de desalojo.

Desalojo basado en memoria

Importante: El desalojo basado en memoria sólo está soportado en Java Platform, Enterprise Edition versión 5 o posterior.

Todos los desalojadores incorporados dan soporte al desalojo basado en memoria que se puede habilitar en la interfaz BackingMap estableciendo el atributo evictionTriggers de BackingMap en `MEMORY_USAGE_THRESHOLD`. Para obtener más

información sobre cómo establecer el atributo evictionTriggers en BackingMap, consulte Interfaz BackingMap y Archivo XML de descriptor ObjectGrid.

El desalojo basado en memoria se basa en el umbral de uso del almacenamiento dinámico. Cuando el desalojo basado en memoria está habilitado en BackingMap y BackingMap tiene cualquier desalojo incorporado, el umbral de uso se establece en un porcentaje predeterminado de la memoria total si el umbral no se ha establecido previamente.

Cuando utilice el desalojo basado en memoria, deberá configurar el umbral de recogida de basura en el mismo valor que su uso del almacenamiento dinámico de destino. Por ejemplo, el umbral de desalojo basado en memoria se establece en el 50 por ciento y el umbral de recogida de basura está al 70 por ciento del nivel predeterminado, el uso del almacenamiento dinámico puede llegar hasta el 70 por ciento. Este aumento del uso del almacenamiento dinámico se produce porque el desalojo basado en memoria sólo se desencadena después de un ciclo de recogida de basura.

Para cambiar el porcentaje del umbral de uso predeterminado, establezca la propiedad memoryThresholdPercentage en el archivo de propiedades del contenedor y servidor para el proceso de servidor eXtreme Scale. Para establecer el umbral de uso de destino en un proceso de cliente, puede utilizar MemoryPoolMXBean.

El algoritmo de desalojo basado en memoria utilizado por WebSphere eXtreme Scale es sensible al comportamiento del algoritmo de recogida de basura que se utiliza. El mejor algoritmo para el desalojo basado en memoria es el recopilador de rendimiento predeterminado de IBM. Los algoritmos de recogida de basura de generación puede provocar un comportamiento no deseado y, por lo tanto, no debe utilizar estos algoritmos con el desalojo basado en memoria.

Para cambiar el porcentaje del umbral de uso, establezca la propiedad memoryThresholdPercentage en los archivos de propiedad de contenedor y servidor para los procesos de servidor de eXtreme Scale.

Durante la ejecución, si el uso de memoria excede el umbral del uso de destino, los desalojadores basados en memoria empiezan a desalojar entradas e intentan mantener el uso de la memoria por debajo del umbral de uso de destino. Sin embargo, no existe ninguna garantía de que la velocidad del desalojo sea lo suficientemente rápida para evitar un posible error de memoria, si el tiempo de ejecución del sistema sigue consumiendo memoria rápidamente.

Visión general de la infraestructura OSGi

OSGi define un sistema de módulo dinámico para Java. La plataforma de servicio OSGi tiene una arquitectura por capas, y está diseñada para ejecutarse en diversos perfiles Java estándar. Puede iniciar servidores y clientes de WebSphere eXtreme Scale en un contenedor OSGi.

Ventajas de la ejecución de aplicaciones en el contenedor OSGi

El soporte OSGi de WebSphere eXtreme Scale le permite desplegar el producto en la infraestructura OSGi de Eclipse Equinox. Anteriormente, si deseaba actualizar los plug-ins utilizados por eXtreme Scale, tenía que reiniciar la máquina virtual Java (JVM) para aplicar las nuevas versiones de los plug-ins. Con la prestación de actualización dinámica que proporciona la infraestructura OSGi, ahora puede

actualizar las clases de plug-in sin reiniciar la JVM. Estos plug-ins los exportan los paquetes de usuario como servicios. WebSphere eXtreme Scale accede al servicio o a los servicios buscándolos en el registro OSGi.

Los contenedores de eXtreme Scale se pueden configurar para que se inicien de forma más fácil y dinámica utilizando el servicio de administración de configuración OSGi o con OSGi Blueprint. Si desea desplegar una cuadrícula de datos nueva con la estrategia de colocación, puede hacerlo creando una configuración OSGi o desplegando un paquete con archivos XML de descriptor de eXtreme Scale. Con el soporte de OSGi, los paquetes de aplicaciones que contienen eXtreme Scale se pueden instalar, iniciar, detener, actualizar y desinstalar sin reiniciar todo el sistema. Con esta posibilidad, puede actualizar la aplicación sin interrumpir la cuadrícula de datos.

Se pueden configurar beans y servicios de plug-in con ámbitos de fragmento personalizados, lo que permite opciones de integración sofisticadas opciones con otros servicios que se ejecutan en la cuadrícula de datos. Cada plug-in puede utilizar clasificaciones OSGi Blueprint para verificar que cada instancia del plug-in está activada en la versión correcta. Se proporcionan un bean gestionado por OSGi (MBean) y el programa de utilidad **xscmd**, que permiten consultar los servicios OSGi de plug-in de eXtreme Scale y sus clasificaciones.

Esta prestación permite a los administradores reconocer rápidamente los errores potenciales de configuración y administración y actualizar las clasificaciones de servicio de plug-in utilizadas por eXtreme Scale.

Paquetes OSGi

Para interactuar con los plug-ins y desplegarlos en la infraestructura OSGi, debe utilizar *paquetes*. En la plataforma de servicio OSGi, un paquete es un archivo de archivado Java (JAR) que contiene código Java, recursos y un manifiesto que describe el paquete y sus dependencias. El paquete es la unidad de despliegue de una aplicación. El producto eXtreme Scale da soporte a los siguientes tipos de paquete:

Paquete de servidor
El paquete de servidor es el archivo `objectgrid.jar`, se instala con la instalación de servidor autónomo de eXtreme Scale, es necesario para ejecutar servidores eXtreme Scale y también se puede utilizar para ejecutar clientes de eXtreme Scale o cachés locales en memoria. El ID de paquete para el archivo `objectgrid.jar` es com.ibm.websphere.xs.server_`<versión>`, donde la versión tiene el formato: `<Versión>.<Release>.<Modificación>`. Por ejemplo, el paquete de servidor para eXtreme Scale versión 7.1.1 es com.ibm.websphere.xs.server_7.1.1.

Paquete de cliente
El paquete de cliente es el archivo `ogclient.jar`, se instala con las instalaciones autónomas y de cliente de eXtreme Scale y se utiliza para ejecutar clientes de eXtreme Scale o cachés locales en memoria. El ID de paquete para el archivo `ogclient.jar` es com.ibm.websphere.xs.client_`versión`, donde la versión tiene el formato: `<Versión>.<Release>.<Modificación>`. Por ejemplo, el paquete de cliente para eXtreme Scale versión 7.1.1 es com.ibm.websphere.xs.client_7.1.1.

Limitaciones

No puede reiniciar el paquete de eXtreme Scale porque no puede reiniciar el Intermediario para solicitudes de objetos (ORB). Para reiniciar el servidor eXtreme Scale, debe reiniciar la infraestructura OSGi.

Visión general de integración de memoria caché

El elemento decisivo que proporciona a WebSphere eXtreme Scale la capacidad de ejecutarse con tal versatilidad y fiabilidad es su aplicación de conceptos de colocación en memoria caché para optimizar la persistencia y la recogida de datos en prácticamente cualquier entorno de despliegue.

Plug-in de memoria caché de nivel 2 (L2) JPA

WebSphere eXtreme Scale incluye los plug-ins de memoria caché de nivel (L2) para los proveedores OpenJPA e Hibernate Java Persistence API (JPA). Cuando se utiliza uno de estos plug-ins, la aplicación utiliza la API JPA. Se introduce una cuadrícula de datos entre la aplicación y la base de datos, lo cual mejora los tiempos de respuesta.

EL uso de eXtreme Scale como un proveedor de memoria caché de nivel 2 aumenta el rendimiento al leer y consultar datos y reduce la carga de la base de datos. WebSphere eXtreme Scale tiene ventajas sobre las implementaciones de memoria caché incorporadas porque la memoria caché se duplica automáticamente entre todos los procesos. Cuando un cliente almacena en memoria caché un valor, todos los demás clientes son capaces de utilizar el valor almacenado en memoria caché que está localmente en la memoria.

Puede configurar la topología y propiedades del proveedor de memoria caché L2 en el archivo `persistence.xml`. Para obtener más información sobre cómo configurar estas propiedades, consulte Propiedades de configuración de la memoria caché JPA.

Consejo: El plug-in de memoria caché L2 JPA requiere una aplicación que utilice las API JPA. Si desea utilizar las API WebSphere eXtreme Scale para acceder a un origen de datos JPA, utilice el cargador JPA. Para obtener más información, consulte “Cargadores JPA” en la página 66.

Consideraciones acerca de la topología de memoria caché L2 JPA

Los factores siguientes incluyen en el tipo de topología que se debe configurar:

1. **¿Qué volumen de datos espera que se almacenen en memoria caché?**
   - Si los datos pueden colocarse en un solo almacenamiento dinámico de JVM, utilice la “Topología incorporada” en la página 28 o la “Topología interna del dominio” en la página 27.
   - Si los datos no pueden colocarse en un solo almacenamiento dinámico de JVM, utilice la “Topología incorporada con particiones” en la página 29 o la “Topología remota” en la página 31
2. **¿Cuál es la proporción esperada de lectura respecto a grabación?**

   La proporción de lectura respecto a grabación afecta al rendimiento de la memoria caché L2. Cada topología maneja las operaciones de lectura y grabación de formas diferentes.
   - “Topología incorporada” en la página 28: lectura local, grabación remota

- "Topología interna del dominio": lectura local, grabación local
- "Topología incorporada con particiones" en la página 29: Particionada: lectura remota, grabación remota
- "Topología remota" en la página 31: lectura remota, grabación remota.

Las aplicaciones que son principalmente de sólo lectura deben utilizar topologías incorporadas e internas de dominio cuando sea posible. Las aplicaciones que realizan más grabación deben utilizar topologías internas de dominio.

3. **¿Cuál es el porcentaje de datos consultados respecto a encontrados por una clave?**

   Cuando están habilitadas, las operaciones de consulta utilizan la memoria caché de consulta de JPA. Habilite la memoria caché de consulta de JPA sólo para aplicaciones con una alta proporción de lectura respecto a grabación, por ejemplo cuando se aproxime al 99% de operaciones de lectura. Si utiliza JPA la memoria caché de consulta de JPA, debe utilizar la "Topología incorporada" en la página 28 o la "Topología interna del dominio".

   La operación de búsqueda por clave capta una entidad de destino si la entidad de destino no tiene ninguna relación. Si la entidad de destino tiene relaciones con el tipo de captación EAGER, se captan estas relaciones junto con la entidad de destino. En la memoria caché de datos JPA, la captación de estas relaciones hace unos pocos aciertos de caché obtengan todos los datos de relación.

4. **¿Cuál es el nivel de obsolescencia tolerado de los datos?**

   En un sistema con pocas JVM, existe la latencia de réplica de datos para operaciones de grabación. El objetivo de la memoria caché es mantener una vista de datos sincronizados final en todas las JVM. Cuando se utiliza la topología interna del dominio, existe un retardo de réplica de datos para las operaciones de grabación. Las aplicaciones que utilizan esta topología deben ser capaces de tolerar lecturas obsoletas y grabaciones simultáneas que pueden sobrescribir los datos.

7.1.1+ Topología interna del dominio

Con una topología interna del dominio, los fragmentos primarios se colocan en cada uno de los servidores de contenedor de la topología. Estos fragmentos internos contiene el conjunto completo de datos para la partición. Cualquiera de estos fragmentos primarios también puede completar operaciones de grabación de memoria caché. Esta configuración elimina el cuello de botella en la topología incorporada donde todas las operaciones de grabación de memoria caché deben pasar por un único fragmento primario.

En una topología interna del dominio, no se crean fragmentos de réplica, incluso si tiene definidas réplicas en los archivos de configuración. Cada fragmento primario redundante contiene una copia completa de los datos, de forma que cada fragmento primario también se puede considerar un fragmento de réplica. Esta configuración utiliza una partición única, similar a la topología incorporada.

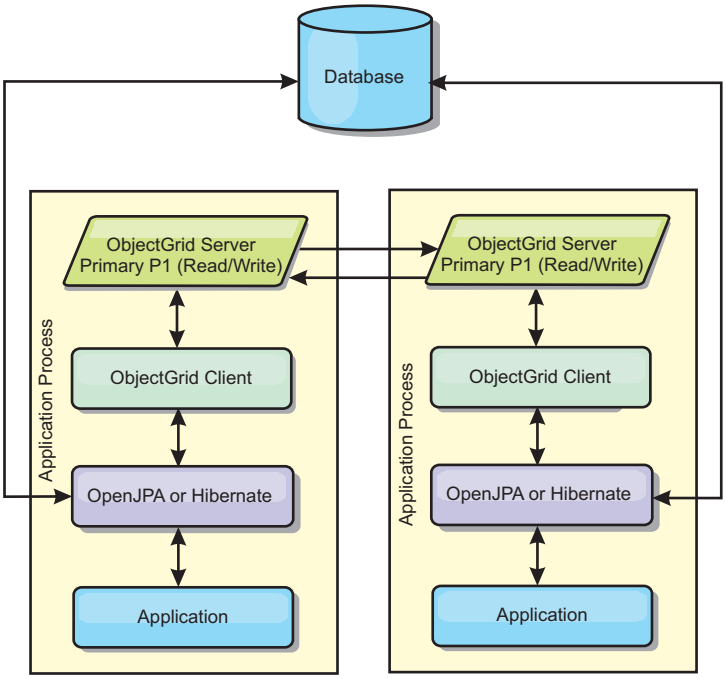


Figura 12. Topología interno del dominio JPA

Propiedades de configuración de memoria caché JPA relacionadas para la topología interna del dominio:

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

Ventajas:

- Las lecturas y actualizaciones de memoria caché son locales.
- La configuración es sencilla.

Limitaciones:

- Esta topología es más adecuada cuando los servidores de contenedor pueden contener todo el conjunto de datos de partición.
- Los fragmentos de réplica, incluso si se han configurado, nunca se colocan ya que cada uno de los servidores de contenedor aloja un fragmento primario. Sin embargo, todos los fragmentos primarios se replican con los otros fragmentos primarios, así que estos fragmentos primarios pasan a ser réplicas entre ellos.

Topología incorporada

Consejo: Considere utilizar la topología interna del dominio para obtener el mejor rendimiento.

Una topología incorporada crea un servidor de contenedor en el espacio de proceso de cada aplicación. OpenJPA e Hibernate leen directamente con la copia en memoria de la memoria caché y escriben en todas las demás copias. Puede mejorar el rendimiento de la escritura utilizando la réplica asíncrona. El rendimiento de esta topología predeterminada es mejor cuando el volumen de datos en caché es lo suficientemente pequeño para caber en un solo proceso. Con una topología incorporada, cree una única partición para los datos.

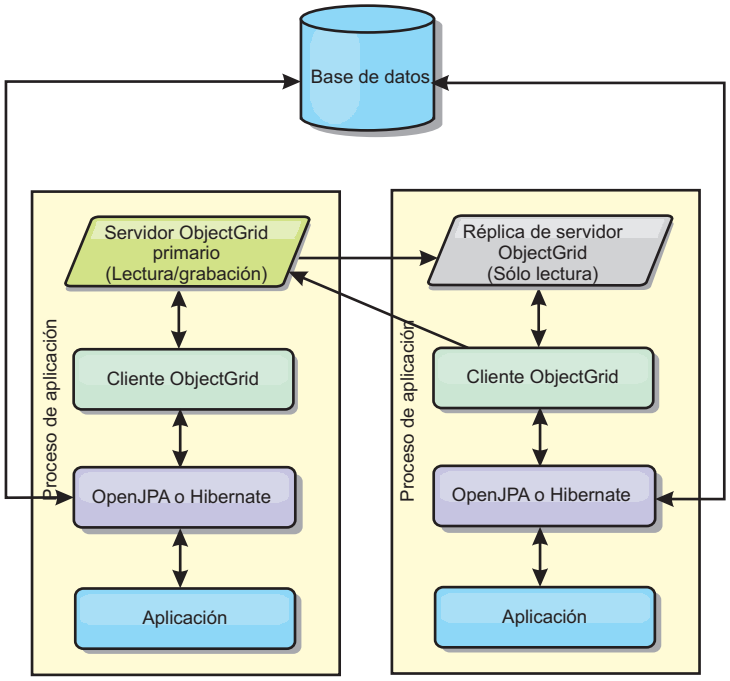


Figura 13. Topología incorporada JPA

Propiedades de configuración de memoria caché JPA relacionadas para la topología incorporada:

```
ObjectGridName=nombre_objectgrid,ObjectGridType=EMBEDDED,MaxNumberOfReplicas=núm_réplicas,ReplicaMode=SYNC | ASYNC | NONE
```

Ventajas:

- Todas las lecturas de memoria caché son acceso locales rápidos.
- La configuración es sencilla.

Limitaciones:

- El volumen de los datos se limita al tamaño del proceso.
- Todas las actualizaciones de memoria caché se envían a través del fragmento primario, lo que crea un cuello de botella.

Topología incorporada con particiones

Consejo: Considere utilizar la topología interna del dominio para obtener el mejor rendimiento.

PRECAUCIÓN:
No utilice la memoria caché de consulta de JPA con una topología particionada incorporada. La memoria caché de consulta almacena los resultados de consulta que son una colección de claves de entidad. La memoria caché de consulta va a la memoria caché de datos para captar todos los datos de entidad. Dado que la memoria caché de datos se divide entre múltiples procesos, estas llamadas adicionales pueden anular las ventajas de la memoria caché L2.

Cuando los datos en memoria caché son demasiado voluminosos para caber en un solo proceso, puede utilizar la topología incorporada particionada. Esta topología divide los datos en varios procesos. Los datos se dividen entre los fragmentos primarios, de modo que cada fragmento primario contiene un subconjunto de los

datos. Aún puede utilizar esta opción cuando la latencia de base de datos sea alta.

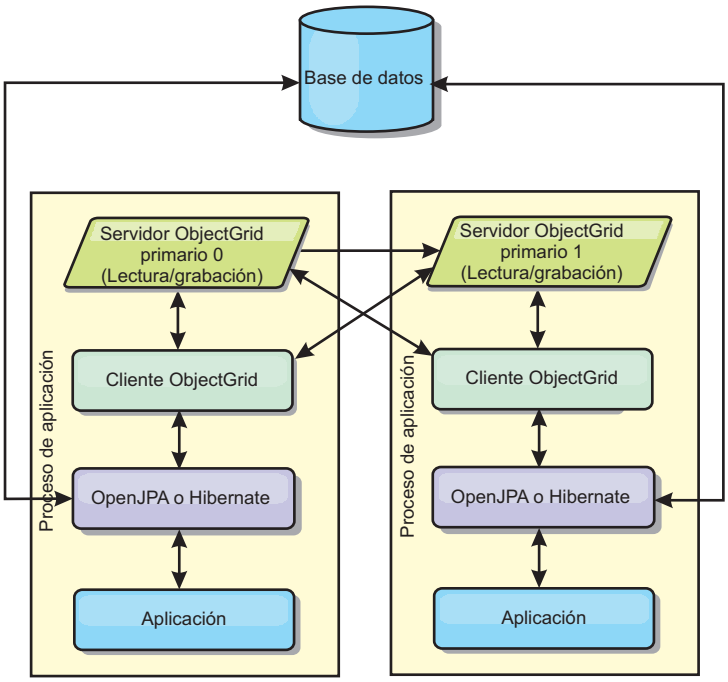


Figura 14. Topología incorporada con particiones JPA

Propiedades de configuración de memoria caché JPA relacionadas para la topología particionada incorporada:

```
ObjectGridName=nombre_objectgrid,ObjectGridType=EMBEDDED_PARTITION,ReplicaMode=SYNC | ASYNC | NONE,
NumberOfPartitions=núm_particiones,ReplicaReadEnabled=TRUE | FALSE
```

Ventajas:

- Almacena grandes cantidades de datos.
- La configuración es sencilla.
- Las actualizaciones de la memoria caché se reparten en diversos procesos.

Limitación:

- La mayoría de las lecturas y actualizaciones de la memoria caché son remotas.

Por ejemplo, para almacenar en memoria caché 10 GB de datos con un máximo de 1 GB por JVM, se necesitan 10 Máquinas virtuales Java. Por lo tanto, el número de particiones debe establecerse en 10 o más. De manera ideal, el número de particiones se debe establecer en un número primo donde cada fragmento almacena una cantidad razonable de memoria. Normalmente, el valor numberOfPartitions es igual al número de Máquinas virtuales Java. Con este valor, cada JVM almacena una partición. Si habilita las réplicas, debe aumentar el número de Máquinas virtuales Java en el sistema. De lo contrario, cada JVM también almacena una partición de réplica, que consume tanta memoria como una partición primaria.

Lea la información sobre el dimensionamiento de la memoria y el cálculo del número de particiones en la *Guía de administración* para maximizar el rendimiento de su configuración elegida.

Por ejemplo, en un sistema con cuatro Máquinas virtuales Java, y el valor del parámetro numberOfPartitions de 4, cada JVM aloja una partición primaria. Una operación de lectura tiene un 25 por ciento de posibilidades de captar datos desde una partición disponible localmente, que es mucho más rápido que obtener los datos de una JVM remota. Si una operación de lectura como, por ejemplo, ejecutar una consulta, debe captar una colección de datos que implican 4 particiones de manera uniforme, el 75 por ciento de las llamadas son remotas y el otro 25 por ciento son locales. Si el valor ReplicaMode se establece en `SYNC` o `ASYNC` y el valor ReplicaReadEnabled está establecido en `true`, se crean las cuatro particiones de réplica y se expanden a lo largo de las cuatro Máquinas virtuales Java. Cada JVM aloja una partición primaria y una partición de réplica. La probabilidad de que la operación de lectura se ejecute de forma local aumenta a un 50 por ciento. La operación de lectura que capta una colección de datos que implican cuatro particiones de manera uniforme tiene un 50 por ciento de llamadas remotas y un 50 por ciento de llamadas locales. Las llamadas locales son mucho más rápidas que las memorias remotas. Siempre que se producen llamadas remotas, baja el rendimiento.

Topología remota

PRECAUCIÓN:
No utilice la memoria caché de consulta de JPA con una topología remota. La memoria caché de consulta almacena los resultados de consulta que son una colección de claves de entidad. La memoria caché de consulta va a la memoria caché de datos para captar todos los datos de entidad. Dado que la memoria caché de datos es remota, estas llamadas adicionales pueden anular las ventajas de la memoria caché L2.

Consejo: Considere utilizar la topología interna del dominio para obtener el mejor rendimiento.

Una topología remota almacena todos los datos almacenados en la memoria caché en uno o más procesos separados, reduciendo el uso de la memoria de los procesos de la aplicación. Puede aprovechar la distribución de los datos en distintos procesos desplegando una cuadrícula de datos particionada y replicada de eXtreme Scale. A diferencia de las configuraciones incorporadas, e incorporadas particionadas, descritas en las secciones anteriores, si desea gestionar la cuadrícula de datos remota debe hacerlo de forma independiente de la aplicación y del proveedor JPA.

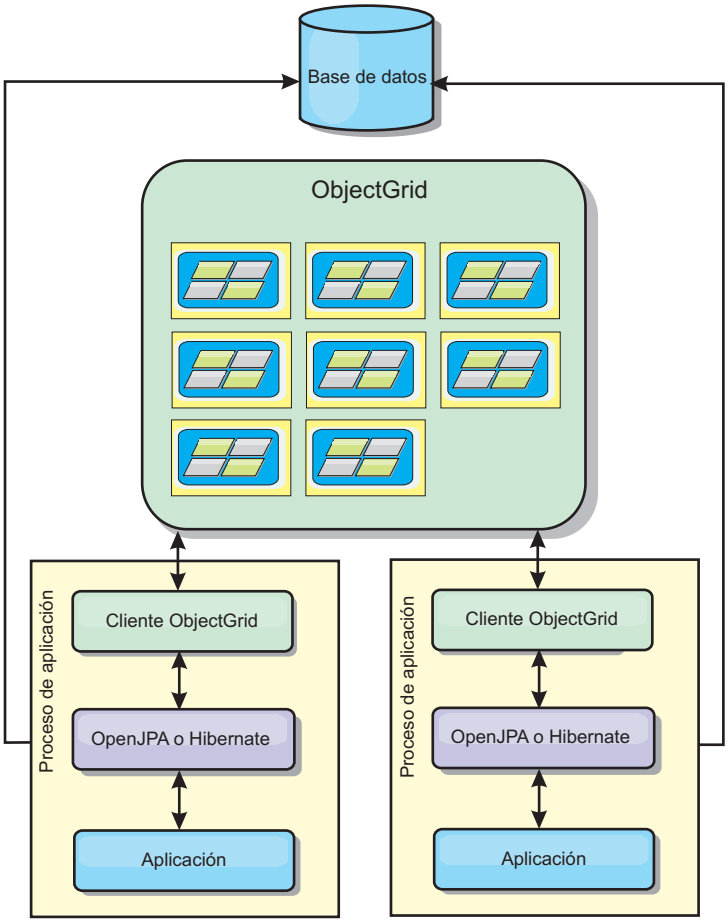


Figura 15. Topología remota JPA

Propiedades de configuración de memoria caché JPA relacionadas para la topología remota:

`ObjectGridName=`*`nombre_objectgrid`*`,ObjectGridType=REMOTE`

El tipo de ObjectGrid REMOTE no requiere ningún valor de propiedad porque el ObjectGrid y la política de despliegue están definidos independientemente de la aplicación JPA. El plug-in de memoria caché JPA se conecta remotamente a un ObjectGrid remoto.

Puesto que toda la interacción con ObjectGrid es remota, esta topología tiene el rendimiento más lento entre todos los tipos ObjectGrid.

Ventajas:

- Almacena grandes cantidades de datos.
- El proceso de aplicaciones está libre de los datos en memoria caché.
- Las actualizaciones de la memoria caché se reparten en diversos procesos.
- Opciones de configuración flexibles.

Limitación:

- Todas las lecturas y actualizaciones de la memoria caché son remotas.

Gestión de sesiones HTTP

El gestor de réplica de sesiones que se envía con WebSphere eXtreme Scale puede funcionar con el gestor de sesiones predeterminado en el servidor de aplicaciones. Los datos de sesión se replican de un proceso a otro proceso para soportar la alta disponibilidad de datos de sesión de usuario.

Características

El gestor de sesiones se ha diseñado para poderse ejecutar en cualquier contenedor de Java Platform, Enterprise Edition Versión 5 o posterior. Dado que el gestor de sesiones no tiene dependencias en las API de WebSphere, puede admitir varias versiones de WebSphere Application Server así como de entornos de servidor de aplicaciones de proveedores.

El gestor de sesiones HTTP proporciona funciones de réplica de sesiones para una aplicación asociada. El gestor de réplica de sesión funciona con el gestor de sesiones para el contenedor web. Juntos, el gestor de sesiones y el contenedor web crean sesiones HTTP y gestionan los ciclos de vida de las sesiones HTTP que están asociadas con la aplicación. Estas actividades de gestión de ciclo de vida incluyen: la invalidación de sesiones basadas en un tiempo de espera o una llamada a un servlet explícito o JavaServer Pages (JSP) y la invocación de escuchas de sesión asociados a la sesión o a la aplicación web. El gestor de sesiones continúa las sesiones en una cuadrícula de datos totalmente replicada, en clúster y particionada. El uso del gestor de sesiones de WebSphere eXtreme Scale permite al gestor de sesiones proporcionar soporte de migración tras error de sesiones HTTP cuando los servidores de aplicaciones concluyen o finalizan inesperadamente. El gestor de sesiones también puede funcionar en entornos que no admitan afinidad, si la afinidad no la aplica un nivel de equilibrador de carga que distribuya solicitudes al nivel de servidor de aplicaciones.

Escenarios de uso

El gestor de sesiones se puede utilizar en los siguientes escenarios:

- En entornos que utilizan servidores de aplicaciones de versiones diferentes de WebSphere Application Server, por ejemplo en un escenario de migración.
- En despliegues que utilizan servidores de aplicaciones de proveedores diferentes. Por ejemplo, una aplicación que se desarrolla en servidores de aplicaciones de código abierto y que se aloja en WebSphere Application Server. Otro ejemplo es una aplicación que se promociona de la fase intermedia a la de producción. Es posible realizar una migración sin interrupciones de estas versiones del servidor de aplicación mientras todas las sesiones HTTP están activas y dan servicio.
- En entornos que requieren que el usuario continúe las sesiones con niveles de calidad de servicio (QoS) más altos. La disponibilidad de sesión se garantiza mejor durante la sustitución por anomalía de servidor que los niveles de QoS de WebSphere Application Server predeterminados.
- En un entorno donde la afinidad de sesiones no se puede garantizar o en entornos en los que la afinidad la mantiene un equilibrador de carga de proveedor. Con un equilibrador de carga de proveedor, el mecanismo de afinidad se debe personalizar para dicho equilibrador de carga.
- En cualquier entorno para descargar el proceso necesario para la gestión de sesiones y el almacenamiento en un proceso Java externo.
- En varias células que permiten la sustitución por anomalía de las sesiones entre las células.

- En múltiples centros de datos o múltiples zonas.

Cómo funciona el gestor de sesiones

El gestor de réplica de sesiones utiliza un escucha de sesión para escuchar los cambios de los datos de sesión. El gestor de réplica de sesiones continúa los datos de sesión en una instancia de ObjectGrid de forma local o remota. Puede añadir el escucha de sesión y el filtro de servlet en todos los módulos web de la aplicación con las herramientas que se proporcionan con WebSphere eXtreme Scale. También puede añadir estos escuchas y filtros de forma manual al descriptor de despliegue web de la aplicación.

Este gestor de réplica de sesión funciona con cada gestor de sesión de contenedor web de proveedor para replicar los datos de sesión en las máquinas virtuales Java. Cuando el servidor original queda inactivo, los usuarios pueden recuperar datos de sesión de otros servidores.

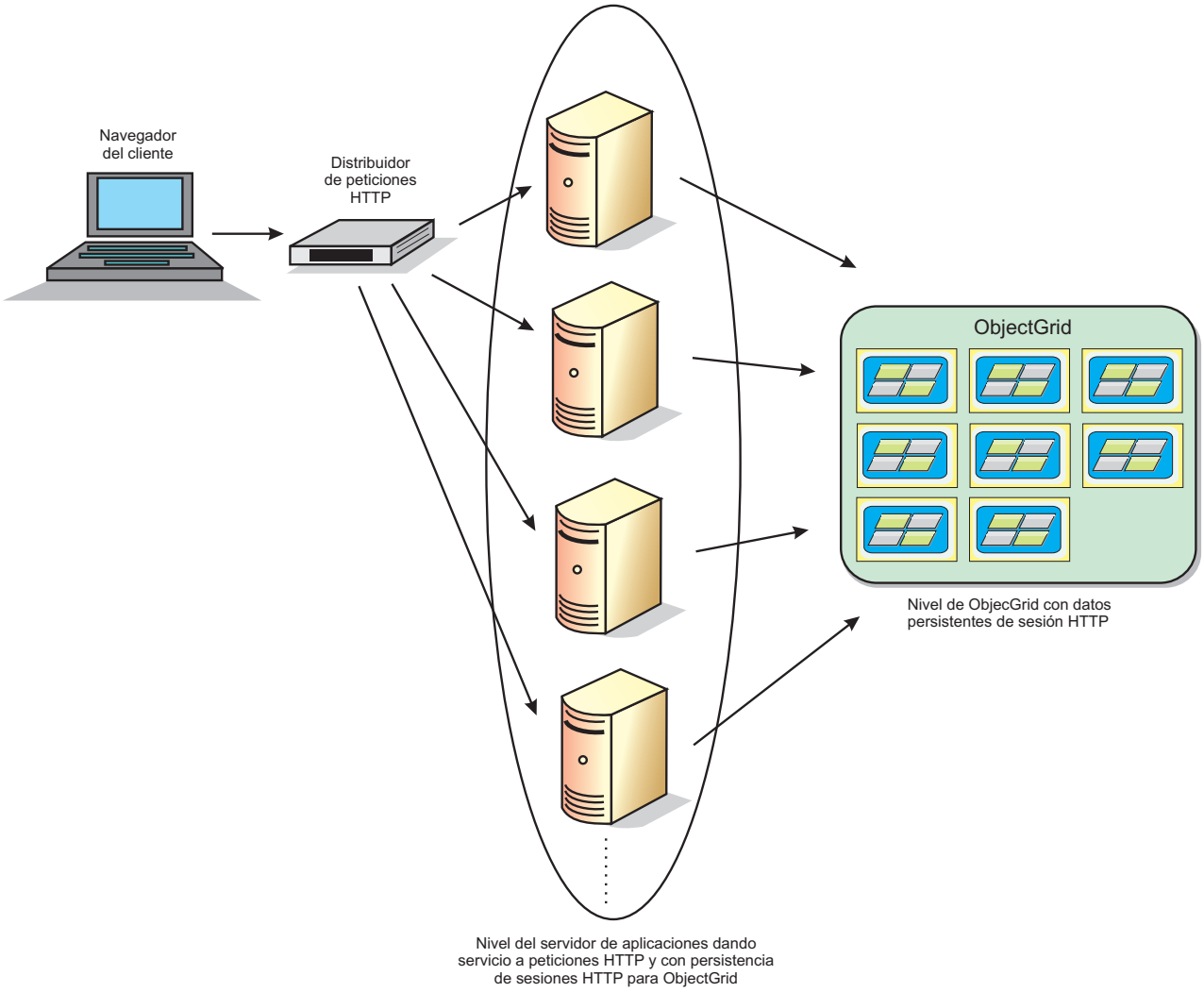


Figura 16. Topología de gestión de sesiones HTTP con una configuración de contenedor remoto

Topologías de despliegue

El gestor de sesiones puede configurarse mediante el uso de dos escenarios de despliegue dinámico diferentes:

Servidores de contenedor de eXtreme Scale incorporados conectados a red
En este escenario, los servidores eXtreme Scale se colocan en los mismos procesos que los servlets. El gestor de sesiones se puede comunicar directamente con la instancia de ObjectGrid local, lo que evita costosos retrasos de red. Este escenario es preferible al ejecutar con afinidad y el rendimiento es crítico.

Servidores de contenedor de eXtreme Scale remotos conectados a red
En este escenario, los servidores eXtreme Scale ejecutan en procesos externos desde el proceso en el que se ejecutan los servlets. El gestor de sesiones se comunica con una cuadrícula de servidor de eXtreme Scale remoto. Este caso de ejemplo es preferible cuando el nivel del contenedor web no tiene la memoria para almacenar los datos de sesión. Los datos de sesión se descargan a un nivel independiente, lo que produce un uso de memoria más bajo en el nivel de contenedor web. Se produce una latencia más alta porque los datos están en una ubicación remota.

Inicio del contenedor incorporado genérico

eXtreme Scale inicia automáticamente un contenedor de ObjectGrid incorporado dentro de cualquier proceso de servidor de aplicaciones cuando el contenedor web inicializa el escucha de sesión o el filtro de servlet, si la propiedad objectGridType se establece en `EMBEDDED`. Consulte Parámetros de inicialización del contexto del servlet si desea más detalles.

No es necesario empaquetar un archivo `ObjectGrid.xml` y un archivo `objectGridDeployment.xml` en el archivo WAR o EAR de aplicación web. Los archivos `ObjectGrid.xml` y `objectGridDeployment.xml` predeterminados están empaquetados en el archivo JAR de producto. De forma predeterminada, se crean correlaciones dinámicas para distintos contextos de aplicaciones web. Se siguen admitiendo las correlaciones estáticas de eXtreme Scale.

Este enfoque para iniciar contenedores ObjectGrid incorporados se aplica a cualquier tipo de servidor de aplicaciones. Los enfoques relacionados con un componente WebSphere Application Server o WebSphere Application Server Community Edition GBean están en desuso.

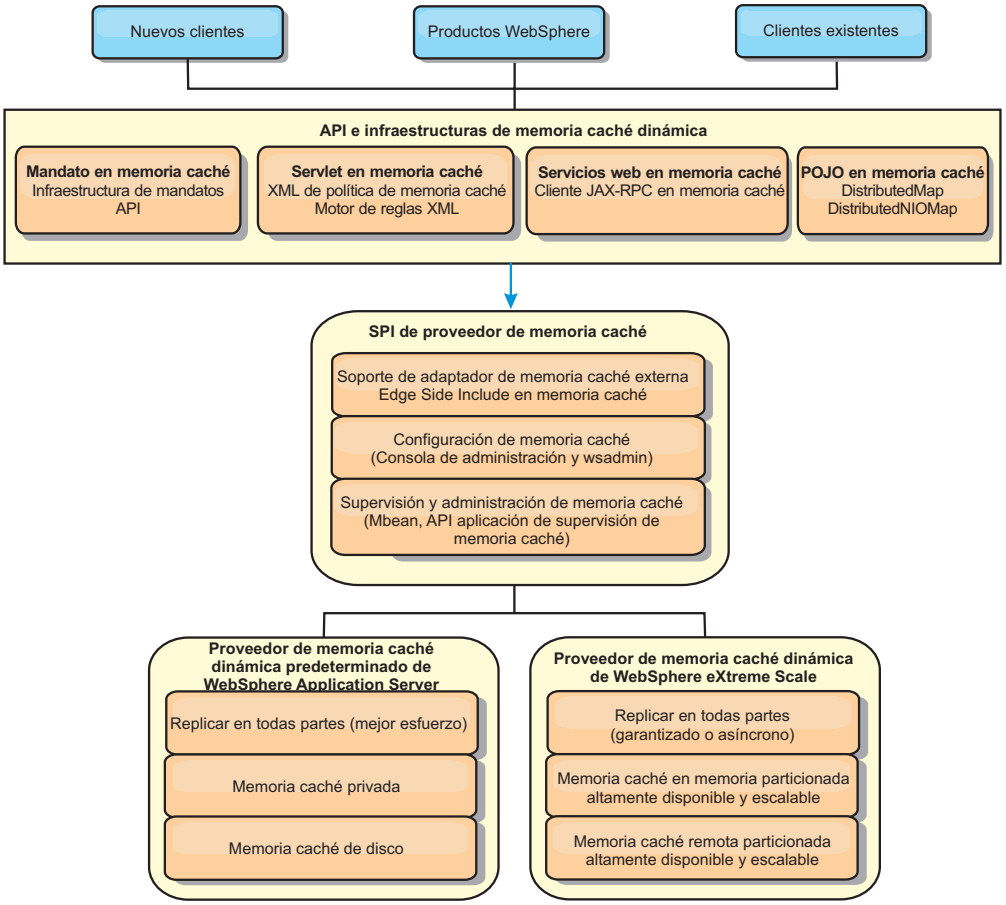
Proveedor de memoria caché dinámica

La API de memoria caché dinámica está disponible para las aplicaciones Java EE desplegadas en WebSphere Application Server. Se puede sacar el máximo partido del proveedor de memoria caché dinámica para almacenar en la memoria caché los datos empresariales, el HTML generado o para sincronizar los datos de la memoria caché en la célula utilizando el servicio de duplicación de datos (DRS).

Visión general

Previamente, el único proveedor de servicios para la API de memoria caché dinámica era el motor de la memoria caché dinámica incorporada en WebSphere Application Server. Los clientes pueden utilizar la interfaz del proveedor de servicios de memoria caché dinámica en WebSphere Application Server para conectarse a la memoria caché dinámica de eXtreme Scale. Configurando esta

prestación, podrá habilitar aplicaciones escritas con la API de memoria caché dinámica o aplicaciones que utilizan la memoria caché de nivel de contenedor (como, por ejemplo, servlets) para sacar el máximo partido de las características y capacidades de rendimiento de WebSphere eXtreme Scale.



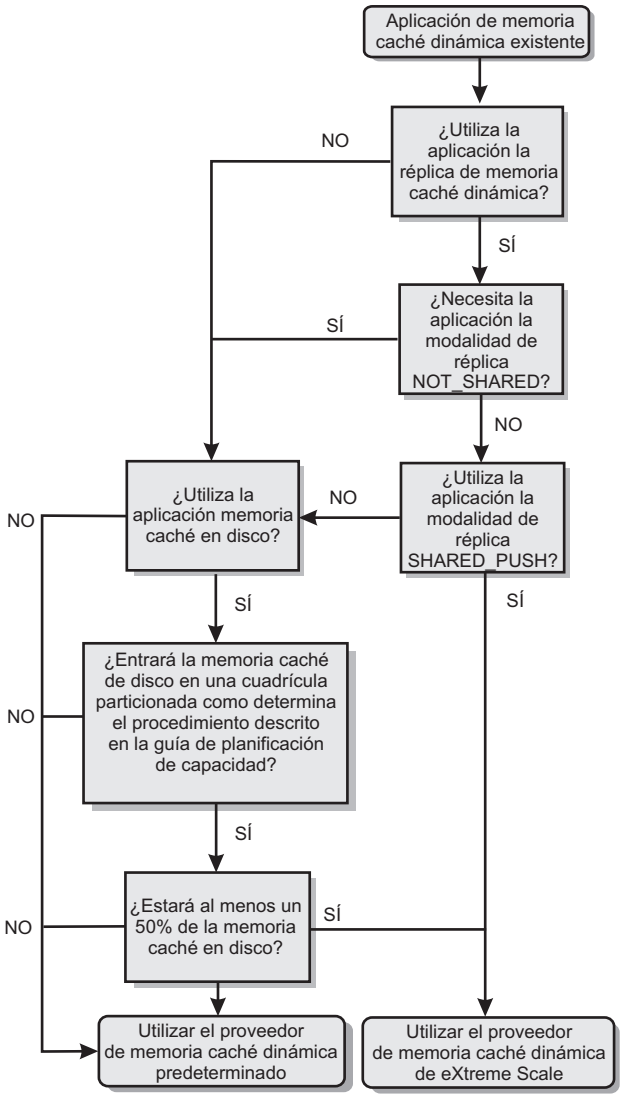
Puede instalar y configurar el proveedor de memoria caché dinámico tal como se describe en Configuración del proveedor de memoria caché dinámica para WebSphere eXtreme Scale.

Decidir cómo sacar partido de WebSphere eXtreme Scale

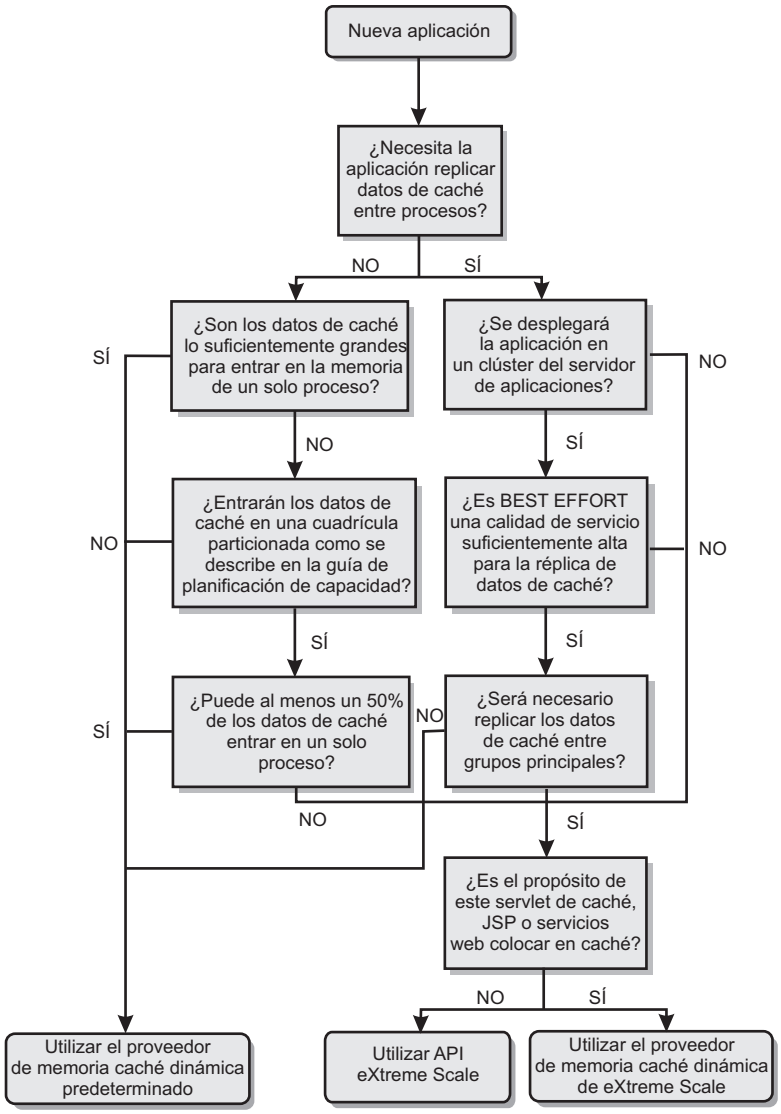
Las características disponibles en WebSphere eXtreme Scale aumentan de forma significativa las capacidades distribuidas de la API de memoria caché dinámica más allá de lo que ofrece el motor de memoria caché dinámica predeterminado y el servicio de duplicación de datos. Con eXtreme Scale, puede crear memorias caché que se distribuyen verdaderamente entre varios servidores, en lugar de sólo duplicarlas y sincronizarlas entre los servidores. Además, las memorias caché de eXtreme Scale son transaccionales y están disponibles, lo que asegura que cada servidor ve los mismos contenidos para el servicio de memoria caché dinámica. WebSphere eXtreme Scale ofrece una mayor calidad de servicio para la réplica de memoria caché que DRS.

Sin embargo, estas ventajas no implican que el proveedor de memoria caché dinámica de eXtreme Scale sea la opción correcta para cada aplicación. Utilice los árboles de decisiones y la matriz de comparaciones de característica siguiente para determinar qué tecnología se adapta mejora a la aplicación.

Árbol de decisiones para migrar las aplicaciones de la memoria caché dinámica existente



Árbol de decisiones para seleccionar un proveedor de memoria caché para las nuevas aplicaciones



Comparación de características

Tabla 1. Comparación de características

Características de memoria caché	Proveedor predeterminado	Proveedor de eXtreme Scale	API de eXtreme Scale
Memoria caché local en memoria	x	x	x
Memoria caché distribuida	Incorporado	Incorporado, particionado-incorporado y particionado-remoto	Varios
Ampliable de forma lineal		x	x
Réplica fiable (síncrona)		ORB	ORB

Tabla 1. Comparación de características (continuación)

Características de memoria caché	Proveedor predeterminado	Proveedor de eXtreme Scale	API de eXtreme Scale
Desbordamiento de disco	x		
Desalojo	LRU/TTL/basado en almacenamiento dinámico	LRU/TTL (por partición)	Varios
Invalidación	x	x	x
Relaciones	ID de dependencia, plantillas,	ID de dependencia, plantillas,	x
Búsquedas sin clave			Consulte e índice
Integración de fondo			Cargadores
Transaccional		Implícita	x
Almacenamiento basado en clave	x	x	x
Sucesos y receptores	x	x	x
Integración de WebSphere Application Server	Sólo una única célula	Varias células	Célula independiente
Soporte de Java Standard Edition		x	x
Supervisión y estadísticas	x	x	x
Seguridad	x	x	x

Tabla 2. Integración de tecnología sin fisuras

Características de memoria caché	Proveedor predeterminado	Proveedor de eXtreme Scale	API de eXtreme Scale
Colocación en memoria caché de los resultados del servlet/JSP de WebSphere Application Server	V5.1+	V6.1.0.25+	
Colocación en memoria caché del resultado de WebSphere Application Server Web Services (JAX-RPC)	V5.1+	V6.1.0.25+	
Colocación en memoria caché de la sesión HTTP			x
Proveedor de memoria caché para OpenJPA e Hibernate			x

Tabla 2. Integración de tecnología sin fisuras (continuación)

Sincronización de la base de datos utilizando OpenJPA e Hibernate			x

Tabla 3. Interfaces de programación

Características de memoria caché	Proveedor predeterminado	Proveedor de eXtreme Scale	API de eXtreme Scale
API basada en mandato	API de infraestructura de mandatos	API de infraestructura de mandatos	API de DataGrid
API basada en correlación	API DistributedMap	API DistributedMap	API ObjectMap
API EntityManager			x

Para ver una descripción más detallada sobre cómo funcionan las memorias caché distribuidas de eXtreme Scale, consulte la información sobre la configuración de despliegue en la *Guía de administración*.

Nota: Una memoria caché distribuida de eXtreme Scale sólo puede almacenar entradas en las que la clave y el valor ambos implementan la interfaz java.io.Serializable.

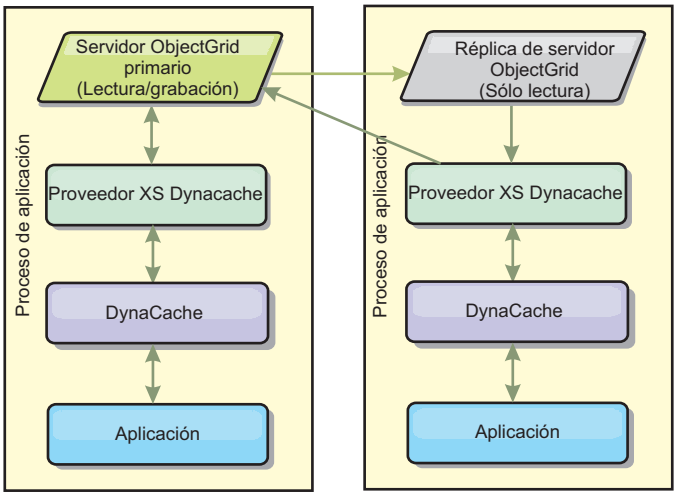
Tipos de topología

Un servicio de memoria caché dinámica creado con el proveedor eXtreme Scale se puede desplegar en cualquiera de las tres topologías disponibles, lo que le permite adaptar la memoria caché específicamente al rendimiento, los recursos y las necesidades administrativas. Estas topologías son: incorporado, particionado incorporado y remoto.

Topología incorporada

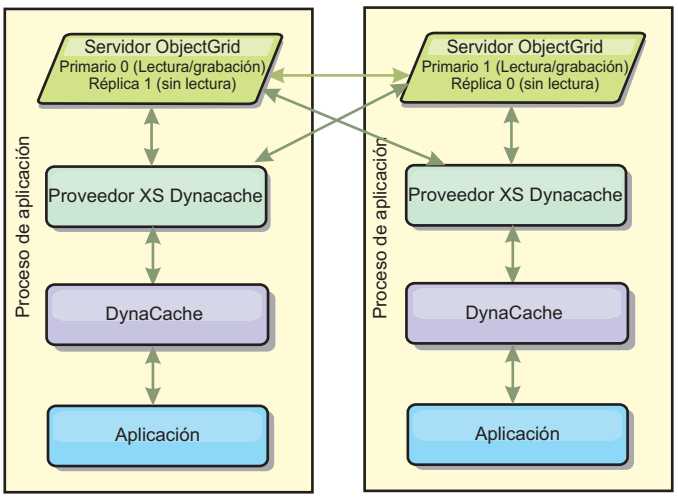
La topología incorporada es similar a la memoria caché dinámica predeterminada y al proveedor DRS. Las instancias de memoria caché distribuida creadas con la topología incorporada conservan una copia de la memoria caché dentro de cada proceso eXtreme Scale que accede al servicio de memoria caché dinámica, lo que permite que todas las operaciones de lectura se produzcan de forma local. Todas las operaciones de escritura pasan por un proceso de único servidor, en el que se gestionan los bloqueos transaccionales, antes de duplicarse el resto de los servidores. Consecuentemente, esta topología es mejor para las cargas de trabajo donde las operaciones de memoria caché-lectura superan en número a las operaciones de memoria caché-escritura.

Con la topología incorporada, las entradas de memoria caché nuevas o actualizadas no son visibles de forma inmediata en cada proceso de servidor único. Una entrada de caché no será visible, incluso para el servidor que lo ha generado, hasta que se propague a través de los servicios de duplicación asíncrona de WebSphere eXtreme Scale. Estos servicios funcionan tan rápido como lo permita el hardware, pero sigue habiendo un pequeño retardo. La topología incorporada se muestra en la siguiente imagen:



Topología incorporada con particiones

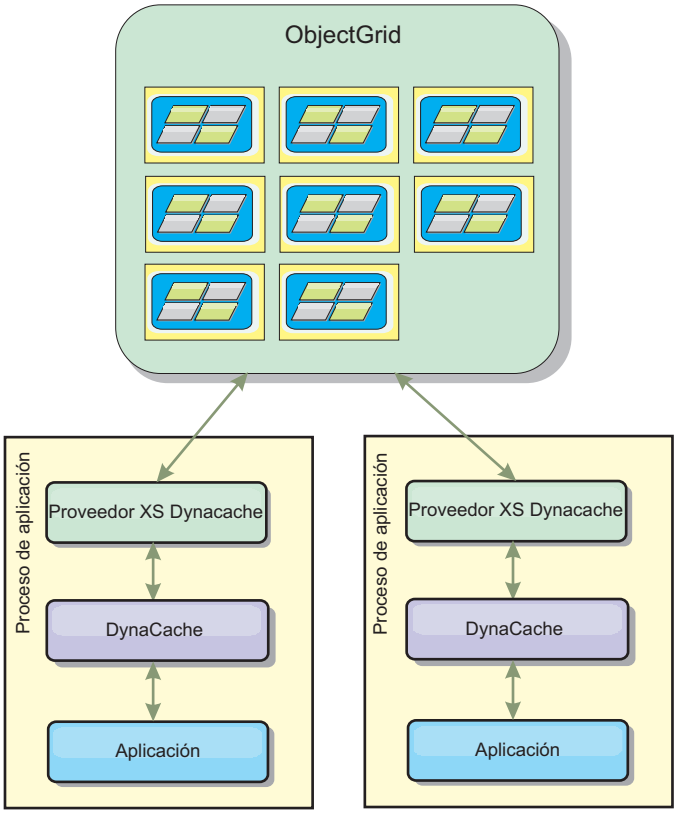
Para las cargas de trabajo donde se producen las escrituras de memoria caché tan a menudo o con más frecuencia que las lecturas, se recomiendan las topologías incorporadas con particiones o las remotas. La topología incorporada con particiones conserva todos los datos de la memoria caché dentro de los procesos WebSphere Application Server que acceden a la memoria caché. Sin embargo, cada proceso sólo almacena una parte de los datos de la memoria caché. Todas las lecturas y escrituras de los datos situados en esta "partición" pasan por el proceso, lo que significa que la mayoría de las solicitudes para la memoria caché se cumplirán con una llamada a procedimiento remoto. Esto genera una mayor latencia para las operaciones de lectura que la topología incorporada, pero la capacidad de la memoria caché distribuida para manejar las operaciones de lectura y escritura se ampliará de forma lineal con el número de procesos WebSphere Application Server que acceden a la memoria caché. Además, con esta topología, el tamaño máximo de la memoria caché no está vinculado al tamaño de un único proceso WebSphere. Puesto que cada proceso sólo alberga una parte de la memoria caché, el tamaño máximo de la memoria caché pasa a ser el tamaño agregado de todos los procesos, menos la sobrecarga del proceso. La topología incorporada con particiones se muestra en la siguiente imagen:



Por ejemplo, suponga que tiene una cuadrícula de procesos de servidor con 256 megabytes de almacenamiento dinámico libre para alojar el servicio de memoria caché dinámica. El proveedor de la memoria caché dinámica predeterminada y el proveedor eXtreme Scale que utiliza la topología incorporada se deben limitar ambos a un tamaño de memoria caché en memoria de 256 megabytes menos la sobrecarga. Consulte la sección Planificación de capacidad y Alta disponibilidad más adelante en este documento. El proveedor eXtreme Scale que utiliza la topología incorporada con particiones se limitará a un tamaño de memoria caché de un gigabyte menos la sobrecarga. De esta forma, el proveedor WebSphere eXtreme Scale posibilita tener servicios de memoria caché dinámica en memoria mayores que el tamaño de un proceso de servidor único. El proveedor de la memoria caché dinámica predeterminada se basa en el uso de una memoria caché de disco para permitir a las instancias de memoria caché crecer más allá del tamaño de un proceso único. En muchas situaciones, el proveedor WebSphere eXtreme Scale puede eliminar la necesidad de una memoria caché de disco y los caros sistemas de almacenamiento en disco que hacen que funcione.

Topología remota

La topología remota también se puede utilizar para eliminar la necesidad de una memoria caché de disco. La única diferencia entre las topologías remota e incorporada con particiones es que todos los datos de la memoria caché se almacenan fuera de los procesos WebSphere Application Server cuando se utiliza la topología remota. WebSphere eXtreme Scale soporta los procesos de contenedor autónomo para los datos de memoria caché. Estos procesos de contenedor tienen una sobrecarga menor que un proceso WebSphere Application Server y, además, no están limitados a utilizar una máquina virtual Java (JVM) determinada. Por ejemplo, un proceso WebSphere Application Server de 32 bits que está accediendo a los datos de un servicio de memoria caché dinámica podría estar situado en un proceso de contenedor eXtreme Scale que se ejecuta en una JVM de 64 bits. Esto permite a los usuarios sacar el máximo partido de la capacidad de memoria aumentada de los procesos de 64 bits para la colocación en memoria caché, sin generar la sobrecarga adicional de 64 bits para los procesos de servidor de aplicaciones. La topología remota se muestra en la siguiente imagen:



Compresión de datos

La compresión es otra característica de rendimiento ofrecida por el proveedor de memoria caché dinámica WebSphere eXtreme Scale que puede ayudar a los usuarios a gestionar la sobrecarga de la memoria caché. El proveedor de la memoria caché dinámica predeterminada no permite la compresión de los datos almacenados en la memoria caché. Con el proveedor eXtreme Scale, esto ahora es posible. La compresión de memoria caché mediante el algoritmo de deflate se puede habilitar en cualquiera de las tres topologías distribuidas. Habilitar la compresión aumentará la sobrecarga para las operaciones de lectura y escritura, pero aumentará drásticamente la densidad de la memoria caché para las aplicaciones, como la colocación en memoria caché de servlet y JSP.

Almacenamiento local de memoria caché en memoria

El proveedor de memoria caché dinámica de WebSphere eXtreme Scale se puede utilizar también para recuperar las instancias de memoria caché dinámica que tienen la **réplica inhabilitada**. Al igual que el proveedor de memoria caché dinámica predeterminado, estas memorias caché pueden almacenar datos no serializables. También pueden ofrecer un mejor rendimiento que el proveedor de memoria caché dinámica en grandes servidores de varios procesadores porque la vía de acceso del código eXtreme Scale está diseñado para maximizar la concurrencia de la memoria caché en memoria.

Diferencias funcionales del motor de memoria caché dinámica y eXtreme Scale

En el caso de las memorias de caché en memoria donde la réplica está inhabilitada, no debería haber ninguna diferencia funcional apreciable entre las memorias caché

respaldadas por el proveedor de la memoria caché dinámica predeterminada y WebSphere eXtreme Scale. Los usuarios no deberían advertir ninguna diferencia funcional entre las dos memorias caché, excepto que las memorias caché respaldadas por WebSphere eXtreme Scale no soportan las estadísticas ni la descarga de disco, ni las operaciones relacionadas con el tamaño de la memoria caché en memoria.

En el caso de las memorias caché donde está habilita la réplica, no habrá ninguna diferencia apreciable en los resultados devueltos por las mayoría de las llamadas de la API de memoria caché dinámica, independientemente de si el cliente está utilizando el proveedor de memoria caché dinámica o el proveedor de memoria caché dinámica eXtreme Scale. Para algunas operaciones, no podrá emular el comportamiento del motor de memoria caché dinámica utilizando eXtreme Scale.

Estadísticas de la memoria caché dinámica

Se informa de las estadísticas de memoria caché dinámica a través de la aplicación CacheMonitor o el MBean de memoria caché dinámica. Cuando se utiliza el proveedor de memoria caché dinámica eXtreme Scale, se informará de las estadísticas a través de estas interfaces, pero el contexto de los valores estadísticos serán diferentes.

Si se comparte una instancia de memoria caché dinámica entre tres servidores llamados A, B y C, el objeto de las estadísticas de memoria caché dinámica sólo devuelve las estadísticas para la copia de la memoria caché en el servidor que realizó la llamada. Si se recuperan las estadísticas en el servidor A, sólo reflejan la actividad en el servidor A.

Con eXtreme Scale, sólo hay una memoria caché distribuida compartida entre todos los servidores, de forma que no es posible rastrear las mayoría de las estadísticas en una base de servidor-por-servidor, como lo hace el proveedor de la memoria caché dinámica predeterminada. A continuación, aparece una lista de las estadísticas generadas por la API de estadísticas de memoria caché y lo que representan cuando se utiliza el proveedor de memoria caché dinámica WebSphere eXtreme Scale. Del mismo modo que el proveedor predeterminado, estas estadísticas no se sincronizan y, por lo tanto, pueden variar hasta el 10% para las cargas de trabajo concurrentes.

- **Coincidencias en la memoria caché** : las coincidencias de la memoria caché se rastrean por servidor. Si el tráfico en el servidor A genera 10 coincidencias de memoria caché y el tráfico en el servidor B genera 20 coincidencias de memoria caché, las estadísticas de la memoria caché informarán de 10 coincidencias de memoria caché en el servidor A y 20 coincidencias de memoria caché en el servidor B.
- **Faltas de coincidencia de la memoria caché**: las faltas de coincidencia de la memoria caché se rastrean por servidor simplemente como las coincidencias de memoria caché.
- **Entradas de la memoria caché**: esta estadística informa del número de entradas de memoria caché en la memoria caché distribuida. Todos los servidores que acceden a la memoria caché informarán del mismo valor para esta estadística y dicho valor será el número total de entradas de memoria caché en la memoria de todos los servidores.
- **Tamaño de la memoria caché en MB**: esta métrica se admite solo para memorias caché que utilizan las topologías remota, incorporada o embedded_partitioned. Notifica el número de megabytes del espacio de almacenamiento dinámico Java consumido por la memoria caché, en toda la cuadrícula. Esta estadística notifica

el uso de almacenamiento dinámico solo para las particiones primarias; debe tener en cuenta las réplicas. Dado que el valor predeterminado de las topologías remotas y embedded_partitioned en una réplica asíncrona, doble este número para obtener el consumo real de memoria de la memoria caché.

- **Supresiones de memoria caché**: esta estadística informa del número total de entradas eliminadas de la memoria caché por un método cualquiera y es un valor de agregado para toda la memoria caché distribuida. Si el tráfico en el servidor A genera 10 invalidaciones y el tráfico en el servidor B genera 20 invalidaciones, el valor en ambos servidores será 30.
- **Supresiones de la memoria caché menos utilizada recientemente (LRU)**: esta estadística es un agregado, al igual que las supresiones de memoria caché. Rastrea el número de entradas que se eliminaron para mantener la memoria caché debajo de su tamaño máximo.
- **Invalidaciones de tiempo de espera**: también se trata de una estadística agregada y rastrea el número de entradas que se eliminaron porque excedieron el tiempo de espera.
- **Invalidaciones explícitas** : también es una estadística agregada, rastrea el número de entradas que se eliminaron con la invalidación directa mediante clave, ID de dependencia o plantilla.
- **Stats ampliados** : el proveedor de memoria caché dinámica de eXtreme Scale exporta las siguientes series de clave stat ampliada.
  - **com.ibm.websphere.xs.dynacache.remote_hits**: el número total de coincidencias de memoria caché en el contenedor eXtreme Scale. Se trata de una estadística agregada y su valor en la correlación de stats ampliados es `long`.
  - **com.ibm.websphere.xs.dynacache.remote_misses**: el número total de faltas de coincidencia de la memoria caché rastreadas en el contenedor eXtreme Scale. Una estadística agregada, su valor en la correlación de stats ampliados es `long`.

Informando de las estadísticas de restablecimiento

El proveedor de memoria caché dinámica le permite restablecer las estadísticas de memoria caché. Con el proveedor predeterminado, la operación restablecer sólo borra las estadísticas en el servidor afectado. El proveedor de memoria caché dinámica eXtreme Scale rastrea la mayoría de sus datos estadísticos en los contenedores de la memoria caché remota. Estos datos no se borran ni modifican cuando se restablecen las estadísticas. En lugar de esto, el comportamiento de la memoria caché dinámica predeterminada se simula en el cliente informando de la diferencia entre el valor actual de una estadística determinada y el valor de dicha estadística la última que se llamó a la operación restablecer en dicho servidor.

Por ejemplo, si el tráfico en el servidor A genera 10 supresiones de memoria caché, las estadísticas en el servidor A y en el servidor B informarán de 10 supresiones. Ahora, si las estadísticas en el servidor B se restablecen y el tráfico en el servidor A genera 10 supresiones adicionales, las estadísticas en el servidor A informarán de 20 supresiones y los stats en el servidor B informarán de 10 supresiones.

Sucesos de la memoria caché dinámica

La API de memoria caché dinámica permite a los usuarios registrar escuchas de sucesos. Cuando se utiliza eXtreme Scale como el proveedor de memoria caché dinámica, los escuchas de sucesos funcionan como se esperaba para las memorias caché locales.

Para las memorias caché distribuidas, el comportamiento del sucesos dependerá de la topología que se utiliza. Para las memorias caché que utilizan la topología incorporada, los sucesos se generarán en el servidor que maneja las operaciones de escritura, también conocidas como el fragmento primario. Esto significa que sólo un servidor recibirá notificaciones de suceso, pero tendrá todas las notificaciones de suceso esperadas normalmente del proveedor de memoria caché dinámica. Puesto que WebSphere eXtreme Scale elige el fragmento primario en el tiempo de ejecución, no es posible garantizar que un proceso de servidor determinado reciba siempre estos sucesos.

Las memorias caché incorporadas con particiones generarán sucesos en cualquier servidor que aloje una partición de la memoria caché. Si una memoria caché tiene 11 particiones y cada servidor de una cuadrícula de 11 servidores de WebSphere Application Server Network Deployment aloja una de estas particiones, cada servidor recibirá los sucesos de la memoria caché dinámica para las entradas de memoria caché que aloja. Ningún proceso de servidor único verá toda la información de todos los sucesos, a menos que las 11 particiones estuvieran alojadas en dicho proceso de servidor. Del mismo modo que la topología incorporada, no es posible garantizar que un proceso de servidor determinado vaya a recibir un conjunto determinado de sucesos o cualquier suceso.

Las memorias caché que utilizan la topología remota no soportan los sucesos de memoria caché dinámica.

Llamadas de MBean

El proveedor de la memoria caché dinámica WebSphere eXtreme Scale no soporta la memoria caché de disco. Cualquier llamada de MBean relacionadas con la memoria caché de disco no funcionará.

Correlación de políticas de duplicación de memoria caché dinámica

El proveedor de memoria caché dinámica incorporada de WebSphere Application Server soporta varias políticas de duplicación de memoria caché. Estas políticas se pueden configurar de forma global o en cada entrada de memoria caché. Consulte la documentación de la memoria caché dinámica si desea una descripción de de estas políticas de duplicación.

El proveedor de memoria caché dinámica eXtreme Scale no permite estas políticas directamente. Las características de duplicación de una memoria caché se determinan a través del tipo de topología distribuida de eXtreme Scale configurado y se aplica a todos los valores colocados en dicha memoria caché, independientemente de la política de duplicación establecida por el servicio de memoria caché dinámica en la entrada. A continuación, aparece una lista de todas las políticas de duplicación soportadas por el servicio de memoria caché dinámica e ilustra que topología eXtreme Scale proporciona características de duplicación similares.

Tenga en cuenta que el proveedor de memoria caché dinámica eXtreme Scale ignora los valores de la política de duplicación DRS en una memoria caché o en una entrada de memoria caché. Los usuarios deben elegir la topología que sea apropiada para sus necesidades de duplicación.

- NOT_SHARED: actualmente ninguna de las topologías proporcionadas por el proveedor de memoria caché dinámica eXtreme Scale puede aproximarse a esta

política. Esto significa que todos los datos almacenados en la memoria caché deben tener claves y valores que implementen java.io.Serializable.

- SHARED_PUSH: la topología incorporada se aproxima a esta topología de duplicación. Cuando se crea una entrada de memoria caché, se duplica en todos los servidores. Los servidores sólo buscan las entradas de memoria caché localmente. Si no se encuentra una entrada de forma local, se da por supuesto que no existe y que no se consulta a otros servidores en relación con esta.
- SHARED_PULL y SHARED_PUSH_PULL: la topología incorporada particionada y la topología remota se aproximan a esta política de réplica. El estado distribuido de la memoria caché es completamente coherente entre todos los servidores.

Esta información se proporciona principalmente para que pueda garantizar que la topología cumple con sus necesidades de coherencia distribuida. Por ejemplo, si la topología incorporada es una mejor opción para sus necesidades de despliegue y rendimiento, pero necesita el nivel de coherencia de memoria caché proporcionado por SHARED_PUSH_PULL, considere utilizar la topología incorporada con particiones, aunque el rendimiento pueda disminuir ligeramente.

Seguridad

Puede proteger las instancias de memoria caché dinámica que se ejecutan en las topologías incorporada e incorporada con particiones con las funciones de seguridad incorporadas en WebSphere Application Server. Consulte la documentación en Protección de servidores de aplicaciones en el centro de información de WebSphere Application Server.

Cuando se ejecuta una memoria caché en la topología remota, es posible para un cliente autónomo de eXtreme Scale para conectarse a la memoria caché y afecta a los contenidos de la instancia de la memoria caché dinámica. El proveedor de la memoria caché dinámica eXtreme Scale tiene una característica de cifrado de sobrecarga baja que puede impedir que los clientes no WebSphere Application Server lean o modifiquen los datos de la memoria caché. Para habilitar esta característica, establezca el parámetro opcional **com.ibm.websphere.xs.dynacache.encryption_password** en el mismo valor en todas las instancias de WebSphere Application Server que accedan al proveedor de memoria caché dinámica. Así se cifrará el valor y los metadatos de usuario para la CacheEntry que utiliza el cifrado AES de 128 bits. Es muy importante que se establezca el mismo valor en todos los servidores. Los servidores no podrán leer los datos colocados en la memoria caché por los servidores con un valor diferente para este parámetro.

Si el proveedor eXtreme Scale detecta que se han establecido distintos valores para esta variable en la misma memoria caché, genera un aviso en el registro del proceso del contenedor eXtreme Scale.

Consulte la documentación de eXtreme Scale sobre “Visión general de seguridad” en la página 135 si se requiere autenticación SSL o de cliente.

Información adicional

- Redbook de memoria caché dinámica
- Documentación de la memoria caché dinámica
  - WebSphere Application Server 7.0
  - WebSphere Application Server 6.1

- Documentación de DRS
  - WebSphere Application Server 7.0
  - WebSphere Application Server 6.1

Integración de base de datos: almacenamiento en memoria caché de grabación diferida, en línea y complementaria

WebSphere eXtreme Scale se utiliza para atender una base de datos tradicional y eliminar la actividad de lectura que normalmente se envía a la base de datos. Puede utilizarse una memoria caché coherente con una aplicación mediante el uso directo o indirecto de un correlacionador de objetos relacionales. La memoria caché coherente puede después descargar de lecturas la base de datos o el programa de fondo. En un escenario ligeramente más complejo, como por ejemplo un acceso transaccional a un conjunto de datos donde sólo algunos de los datos necesitan garantías de persistencia tradicional, puede usarse el filtrado para descargar incluso transacciones de grabación.

Puede configurar WebSphere eXtreme Scale para que funcione como un espacio de proceso de base de datos en memoria muy flexible. No obstante, WebSphere eXtreme Scale no es un correlacionador de objetos relacionales (ORM). No sabe de dónde proceden los datos de la cuadrícula de datos. Una aplicación o un ORM puede colocar datos en un servidor eXtreme Scale. Es responsabilidad del origen de datos garantiza que son coherentes con la base de datos de la que proceden los datos. Esto significa que eXtreme Scale no puede invalidar los datos extraídos de una base de datos automáticamente. La aplicación o el correlacionador debe proporcionar esta función y gestionar los datos almacenados en eXtreme Scale.

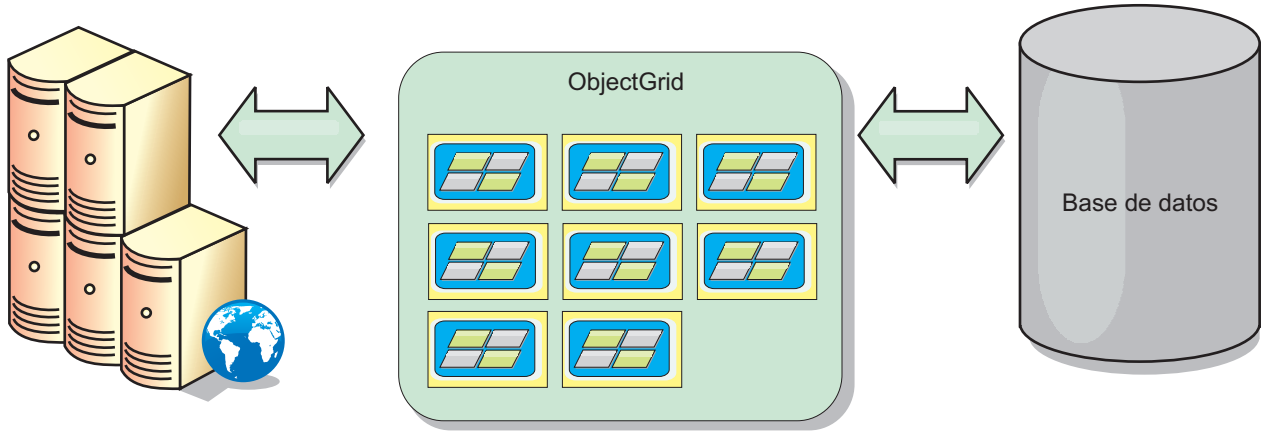


Figura 17. ObjectGrid como un almacenamiento intermedio de base de datos

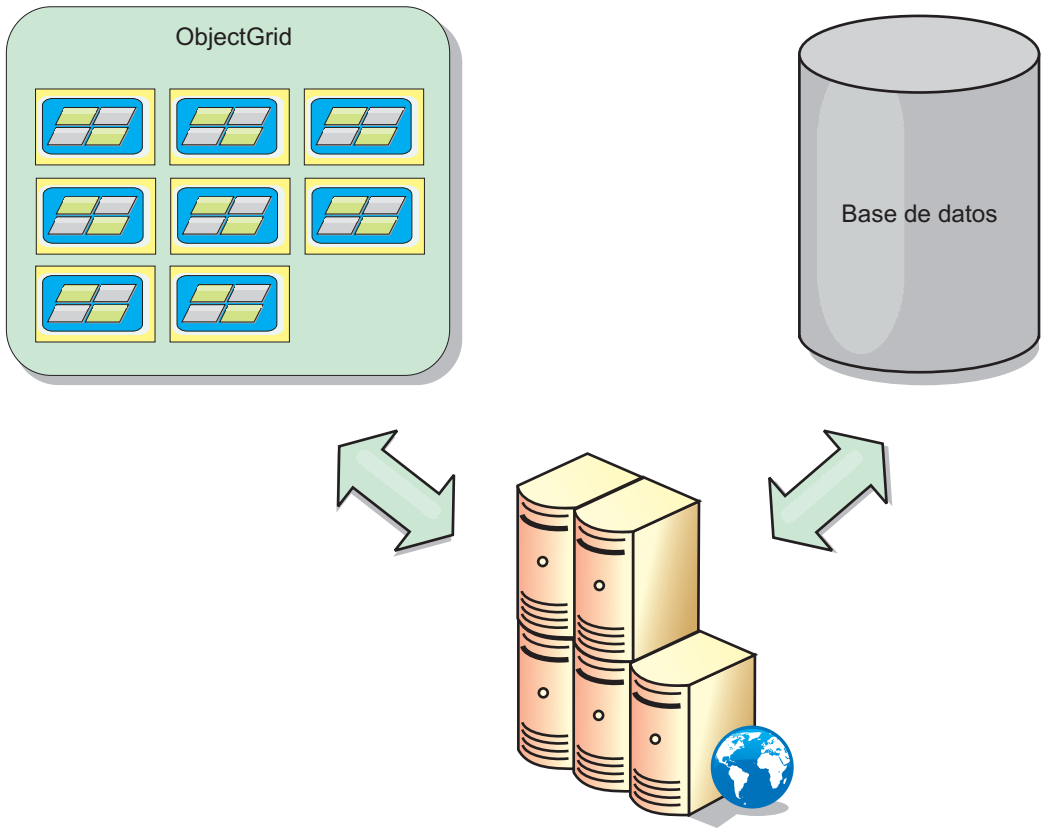


Figura 18. ObjectGrid como una memoria caché secundaria

Memoria caché escasa y completa

WebSphere eXtreme Scale puede utilizarse como una memoria caché escasa o una memoria caché completa. Una memoria caché escasa sólo mantiene un subconjunto de los datos totales, mientras que una memoria caché completa conserva todos los datos y se puede llenar de forma poca activa, conforme se requieran los datos. A las memorias caché escasas normalmente se accede utilizando claves (en lugar de índices o consultas) puesto que los datos sólo están parcialmente disponibles.

memoria caché escasa

Cuando una clave no está presente en una memoria caché escasa, o los datos no están disponibles y se produce una falta de coincidencia de memoria caché, se invoca el siguiente nivel. Los datos se captan, desde una base de datos, por ejemplo, y se insertan en el nivel de la memoria caché de cuadrícula de datos. Si utiliza una consulta o un índice, sólo se accede a los valores cargados actualmente y las solicitudes no se remiten a los demás niveles.

Memoria caché completa

Una memoria caché completa contiene todos los datos necesarios y se puede acceder a la misma utilizando atributos que no son de clave con índices o consultas. Una memoria caché completa se precarga con datos de la base de atos antes de que la aplicación intente acceder a los datos. Una memoria caché completa puede funcionar como una sustitución de base de datos después de que se carguen los datos. Puesto que están disponibles todos los datos, las consultas y

los índices se pueden utilizar para encontrar y agregar datos.

Memoria caché complementaria

Cuando se utiliza WebSphere eXtreme Scale como memoria caché complementaria, se utiliza el programa de fondo con la cuadrícula de datos.

Memoria caché complementaria

Puede configurar el producto como una memoria caché complementaria para la capa de acceso a datos de una aplicación. En este escenario, WebSphere eXtreme Scale se utiliza para almacenar temporalmente objetos que normalmente se recuperarían de una base de datos de programa de fondo. Las aplicaciones comprueban si la cuadrícula de datos contiene los datos. Si los datos están en la cuadrícula de datos, los datos se devuelven al emisor. Si los datos no existen, los datos se recuperan de la base de datos de fondo. A continuación, los datos se insertan en la cuadrícula de datos de forma que la siguiente solicitud pueda utilizar la copia almacenada en memoria caché. El diagrama siguiente muestra cómo se puede utilizar WebSphere eXtreme Scale como una memoria caché complementaria con una capa de acceso a datos arbitrarios como por ejemplo OpenJPA o Hibernate.

Plug-ins de memoria caché para Hibernate y OpenJPA

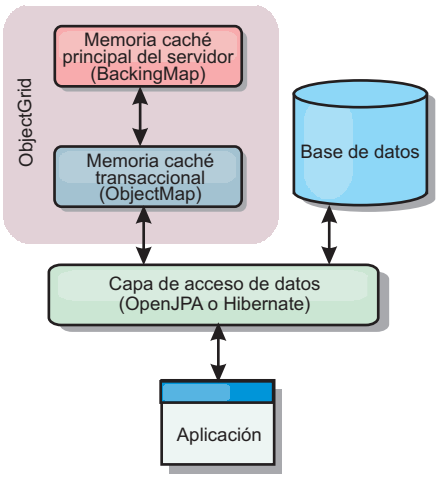


Figura 19. Memoria caché complementaria

Los plug-ins de memoria caché para OpenJPA e Hibernate se incluyen en WebSphere eXtreme Scale, de forma que puede utilizar el producto como una memoria caché complementaria automática. El uso de WebSphere eXtreme Scale como un proveedor de memoria caché aumenta el rendimiento cuando se leen y consultan datos y reduce la carga de la base de datos. WebSphere eXtreme Scale presenta algunas ventajas sobre las implementaciones de memoria caché incorporada ya que la memoria caché se replica automáticamente entre procesos. Cuando un cliente almacena en memoria caché un valor, todos los demás clientes pueden utilizar el valor almacenado en la memoria.

Memoria caché en línea

Puede configurar almacenamiento en memoria caché en línea para un programa de fondo de base de datos o como una memoria complementaria para una base de datos. El almacenamiento en memoria caché en línea utiliza eXtreme Scale como el

medio principal para interactuar con los datos. Cuando se utiliza eXtreme Scale como una memoria caché en línea, la aplicación interactúa con el programa de fondo mediante un plug-in Loader.

Memoria caché en línea

Cuando se utiliza como una memoria caché en línea, WebSphere eXtreme Scale interactúa con el programa de fondo utilizando un plug-in Loader. Este escenario puede simplificar el acceso a datos porque las aplicaciones pueden acceder a las API eXtreme Scale directamente. Se da soporte a distintos escenarios de almacenamiento en memoria caché en eXtreme Scale para garantizar que los datos de la memoria caché y los datos del programa de fondo estarán sincronizados. El diagrama siguiente ilustra cómo una memoria caché en línea interactúa con la aplicación y el programa de fondo.

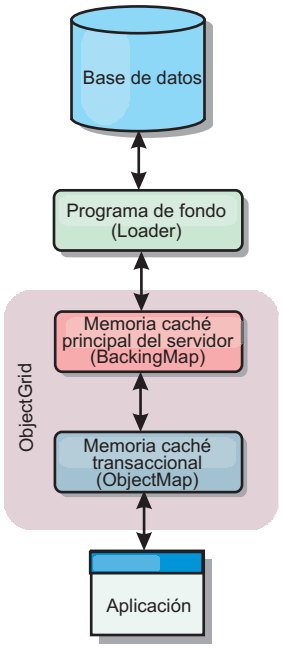


Figura 20. Memoria caché en línea

La opción de memoria caché en línea simplifica el acceso de datos, porque permite a las aplicaciones acceder a las API de eXtreme Scale directamente. WebSphere eXtreme Scale soporta varios escenarios de memoria caché en línea, del modo siguiente.

- Lectura directa
- Grabación directa
- Grabación diferida

Caso de ejemplo de almacenamiento en memoria caché de lectura directa

Una memoria caché de lectura directa es una memoria caché escasa que carga de forma poco activa entradas de datos por clave cuando se solicitan. Esto se lleva a cabo sin que el solicitante sepa cómo se llenan las entradas. Si los datos no se pueden encontrar en la memoria caché de eXtreme Scale, eXtreme Scale recuperará los datos que faltan del plug-in Loader, que carga los datos de la base de datos de programa de fondo y los inserta en la memoria caché. Las solicitudes subsiguientes

para la misma clave de datos se encontrarán en la memoria caché hasta que se elimina, anula o desaloja.

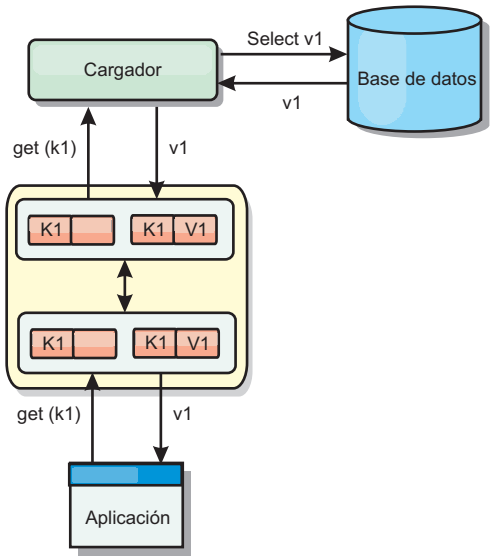


Figura 21. Almacenamiento en memoria caché de lectura directa

Caso de ejemplo de almacenamiento en memoria caché de grabación directa

En una memoria caché de grabación directa, cada grabación en la memoria caché graba de forma síncrona en la base de datos mediante el cargador. Este método proporciona coherencia con el programa de fondo, pero reduce el rendimiento de grabación porque la operación de la base de datos es síncrona. Como que la memoria caché y la base de datos están actualizadas, las lecturas subsiguientes para los mismos datos se encontrarán en la memoria caché, evitando la llamada a la base de datos. Una memoria caché de grabación directa suele utilizarse junto con una memoria caché de lectura directa.

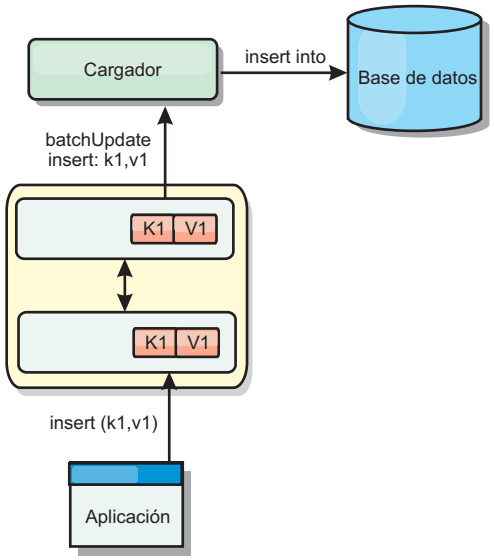


Figura 22. Almacenamiento en memoria caché de grabación directa

Caso de ejemplo de almacenamiento en memoria caché de grabación anticipada

La sincronización de base de datos se puede mejorar grabando los cambios de forma asíncrona. Esto se conoce como memoria caché de grabación diferida o de grabación aplazada. En su lugar, los cambios que normalmente se grabarían de forma síncrona en el cargador se colocarán en el almacenamiento intermedio de eXtreme Scale y se grabarán en la base de datos utilizando una hebra de subordinada. El rendimiento de grabación se mejora de forma significativa porque la operación de la base de datos se elimina de la transacción del cliente y se pueden comprimir las grabaciones de la base de datos.

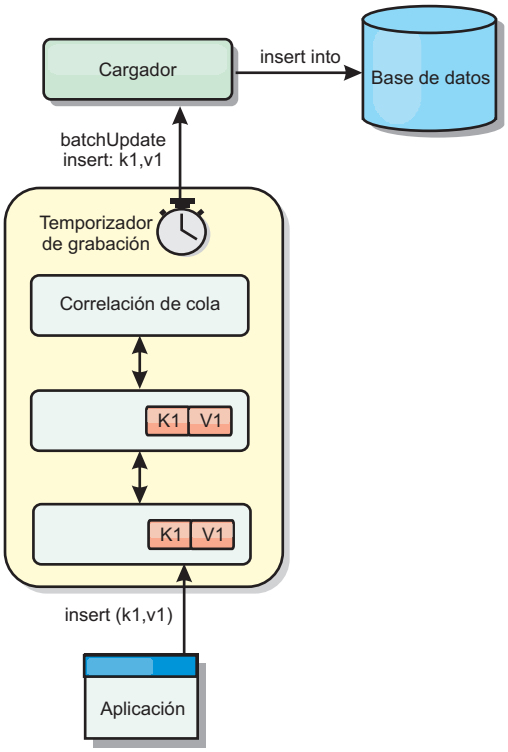


Figura 23. Almacenamiento en memoria caché de grabación diferida

Almacenamiento en memoria caché de grabación diferida

Puede utilizar el almacenamiento en la memoria caché de grabación diferida para reducir la sobrecarga que se produce al actualizar una base de datos utilizada como programa de fondo.

Visión general del almacenamiento en memoria caché con grabación diferida

El almacenamiento en memoria caché de grabación diferida pone en cola de forma asíncrona actualizaciones del plug-in de cargador (Loader). Puede mejorar el rendimiento mediante la desconexión de actualizaciones, inserciones y eliminaciones de una correlación, la sobrecarga de la actualización de la base de datos de programa de fondo. La actualización asíncrona se realiza después de un retardo basado en la hora (por ejemplo, cinco minutos) o un retardo basado en entradas (1000 entradas).

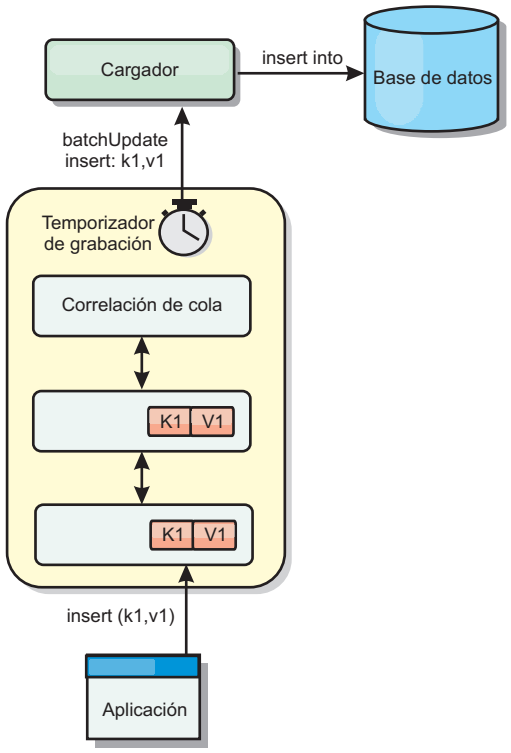


Figura 24. Almacenamiento en memoria caché de grabación diferida

La configuración de la grabación diferida en BackingMap crea una hebra entre el cargador y la correlación. El cargador delega las solicitudes de datos a través de la hebra de acuerdo con los valores de configuración del método BackingMap.setWriteBehind. Cuando una transacción de eXtreme Scale inserta, actualiza o elimina una entrada de una correlación, se crea un objeto LogElement para cada uno de estos registros. Estos elementos se envían al cargador de grabación diferida y se ponen en cola en un objeto ObjectMap especial llamado correlación de cola. Cada correlación de respaldo con el valor de grabación diferida habilitado tiene sus propias correlaciones de cola. Una hebra de grabación diferida elimina periódicamente los datos en cola de las correlaciones de cola y los envía al cargador de programa de fondo real.

El cargador de grabación diferida sólo envía los tipos de inserción, actualización y eliminación de objetos LogElement al cargador real. Todos los demás tipos de objetos LogElement, por ejemplo el tipo EVICT, se pasan por alto.

El soporte de grabación diferida *es* una ampliación del plug-in Loader, que puede utilizar para integrar eXtreme Scale con la base de datos. Por ejemplo, consulte la información del apartado Configuración de cargadores JPA sobre cómo configurar un cargador JPA.

Ventajas

La habilitación del soporte de grabación diferida tiene las ventajas siguientes:

- **Aislamiento de anomalía de programa de fondo:** el almacenamiento de grabación diferida proporciona una capa de aislamiento de las anomalías de programa de fondo. Cuando la base de datos de programa de fondo falla, las actualizaciones se ponen en cola en la correlación de cola. Las aplicaciones

pueden continuar con las transacciones a eXtreme Scale. Cuando se recupera el programa de fondo, los datos de la correlación de cola se envían al programa de fondo.

- **Carga reducida de programa de fondo** el cargador de grabación diferida fusiona las actualizaciones según una clave, de forma que sólo existe una actualización fusionada por clave en la correlación de cola. Este procedimiento reduce el número de actualizaciones en la base de datos de programa de fondo.
- **Rendimiento mejorado de transacciones:** los tiempos individuales de las transacciones de eXtreme Scale se reducen porque la transacción no necesita esperar a que los datos se sincronicen con el programa de fondo.

Consideraciones sobre el diseño de aplicaciones

Habilitar el soporte de grabación diferida es sencillo, pero diseñar una aplicación que funcione con el soporte de grabación diferida requiere un cuidado especial. Sin el soporte de grabación diferida, la transacción ObjectGrid encierra la transacción del programa de fondo. La transacción ObjectGrid se inicia antes de que se inicie la transacción de programa de fondo, pero termina después de que termine la transacción de programa de fondo.

Con el soporte de grabación diferida habilitado, la transacción ObjectGrid finaliza antes de que se inicie la transacción de programa de fondo. La transacción ObjectGrid y la transacción del programa de fondo se desacoplan.

Restricciones de la integridad referencial

Cada correlación de respaldo que se configura con soporte de grabación diferida tiene su propia hebra de grabación diferida que empuja los datos al programa de fondo. Por lo tanto, los datos que se actualizan en correlaciones diferentes de una transacción ObjectGrid se actualizan en el programa de fondo en diferentes transacciones de programa de fondo. Por ejemplo, la transacción T1 actualiza la clave key1 en la correlación Map1 y la clave key2 en la correlación Map2. La actualización de key1 en la correlación Map1 se actualiza en el programa de fondo en una transacción de programa de fondo, y la clave key2 actualizada en la correlación Map2 se actualiza en el programa de fondo en otra transacción de programa de fondo mediante distintas hebras de grabación diferida. Si los datos almacenados en Map1 y Map2 tienen relaciones, como restricciones de clave foránea en el programa de fondo, puede que se produzca un error en las actualizaciones.

Al diseñar las restricciones de la integridad referencial en la base de datos de programa de fondo, asegúrese de que se permiten las actualizaciones que no funcionan.

Comportamiento de bloqueo de correlaciones de cola

Otra diferencia principal en el comportamiento de las transacciones es el comportamiento de bloqueo. ObjectGrid admite tres estrategias de bloqueo distintas: pesimista (PESSIMISTIC), optimista (OPTIMISTIC) y ninguno (NONE). Las correlaciones de cola de grabación diferida utilizan la estrategia de bloqueo pesimista independientemente de la estrategia de bloqueo configurada en el mapa de respaldo. Existen dos tipos diferentes de operaciones que adquieren un bloqueo en la correlación de cola:

- Cuando se confirma una transacción ObjectGrid, o se produce un vaciado (vaciado de correlación o vaciado de sesión), la transacción lee la clave de la correlación de cola y coloca un bloqueo S en la clave.
- Cuando se confirma una transacción ObjectGrid, la transacción intenta actualizar el bloqueo S a un bloqueo X en la clave.

Debido a este comportamiento de correlación de colas adicional, puede ver algunas diferencias en el comportamiento del bloqueo.

- Si la correlación de usuarios está configurada como estrategia de bloqueo pesimista (PESSIMISTIC), no hay mucha diferencia de comportamiento en el bloqueo. Cada vez que se llama a una operación de desecho o confirmación, se coloca un bloqueo S en la misma clave de la misma correlación de colas. Durante la confirmación, no sólo se adquiere un bloqueo X para la clave en la correlación de usuarios, sino que además de adquiere para la clave en la correlación de colas.
- Si la correlación de usuarios está configurada como estrategia de bloqueo optimista (OPTIMISTIC) o ninguna (NONE), la transacción de usuario seguirá el patrón de estrategia de bloqueo pesimista (PESSIMISTIC). Cada vez que se llama a una operación de desecho o confirmación, se adquiere un bloqueo S en la misma clave de la misma correlación de colas. Durante la confirmación se adquiere un bloqueo X para la clave en la correlación de colas utilizando la misma transacción.

Reintentos de transacción de cargador

ObjectGrid no admite transacciones XA o en dos fases. La hebra de grabación diferida elimina los registros de la correlación de cola y actualiza los registros del programa de fondo. Si se produce una anomalía en el servidor durante la transacción, puede que se pierdan algunas actualizaciones del programa de fondo.

El cargador de grabación diferida reintentará automáticamente la grabación de las transacciones con anomalías y enviará un objeto LogSequence en duda al programa de fondo para evitar la pérdida de datos. Esta acción requiere que el cargador sea idempotente, que significa que cuando Loader.batchUpdate(TxId, LogSequence) se llama dos veces con el mismo valor, el resultado es como si se aplicara sólo una vez. Las implementaciones de cargador deben implementar la interfaz RetryableLoader para habilitar esta característica. Consulte la documentación de la API para obtener información detallada.

Anomalías del cargador

El plug-in de cargador puede fallar cuando no puede comunicarse con el programa de fondo de la base de datos. Esto puede suceder si el servidor de bases de datos o la conexión de red está inactivo. El cargador de grabación diferida pondrá en cola las actualizaciones e intentará empujar los cambios de los datos al cargador de forma periódica. El cargador debe notificar al tiempo de ejecución de ObjectGrid que hay un problema de conectividad de base de datos; para ello, emitirá una excepción LoaderNotAvailableException.

Por lo tanto, la implementación del cargador debe distinguir entre una anomalía de datos o un anomalía física del cargador. La anomalía de datos debe emitirse o volver a emitirse como excepción LoaderException o OptimisticCollisionException, pero una anomalía física del cargador debe emitirse o volver a emitirse como excepción LoaderNotAvailableException. ObjectGrid maneja estas dos excepciones de manera diferente:

- Si el cargador de grabación diferida obtiene una excepción LoaderException, el cargador de grabación diferida considerará la anomalía como un error de los datos, como por ejemplo un error de clave duplicada. El cargador de grabación diferida anulará el proceso por lotes de la actualización, e intentará actualizar un registro cada vez para aislar la anomalía de los datos. Si se vuelve a obtener una excepción {{LoaderException}} durante la actualización de un registro, se crea un registro de actualización con errores y se anota en la correlación de actualizaciones con errores.
- Si el cargador de grabación diferida obtiene una excepción LoaderNotAvailableException, el cargador de grabación diferida la considerará como un error porque no puede conectase a la base de datos, por ejemplo, el programa de fondo de la base de datos está inactivo, una conexión de base de datos no está disponible, o la red no está activa. El cargador de grabación diferida esperará 15 segundos y después volverá a intentar realizar la actualización por lotes en la base de datos.

El error habitual es emitir una excepción LoaderException cuando debería emitirse una excepción LoaderNotAvailableException. Todos los registros puestos en cola en el cargador de grabación diferida pasan a ser registros de actualizaciones con anomalías, que anula el propósito del aislamiento de anomalías de programa de fondo.

Consideraciones sobre el rendimiento

El soporte de almacenamiento en memoria caché de grabación diferida aumenta el tiempo de respuesta al eliminar la actualización del cargador de la transacción. También aumenta el rendimiento de base de datos ya que las actualizaciones de base de datos se combinan. Es importante comprender la sobrecarga que supone la hebra de grabación diferida, que extrae los datos de la correlación de cola y los envía al cargador.

El número máximo de actualizaciones o el tiempo máximo de actualización debe ajustarse en función del entorno y de los patrones de uso esperados. Si el valor del número máximo de actualizaciones o el tiempo máximo de actualización es demasiado pequeño, la sobrecarga de la hebra de grabación diferida puede sobrepasar las ventajas. Si se especifica un valor elevado para estos dos parámetros, podría aumentarse el uso de memoria al poner en cola los datos y aumentarse el tiempo obsoleto de los registros de la base de datos.

Para obtener un rendimiento óptimo, ajuste los parámetros de grabación diferida de acuerdo con los factores siguientes:

- Índice de transacciones de lectura y grabación.
- Misma frecuencia de actualización de registros.
- Latencia de actualización de la base de datos.

Cargadores

Con un plug-in Loader plug-in, una correlación de cuadrícula de datos puede actuar como una memoria caché de datos para los datos que se mantienen normalmente en un almacén persistente en el mismo sistema o en otro sistema. Generalmente, se utiliza una base de datos o un sistema de archivos como almacenamiento persistente. Una máquina virtual Java (JVM) remota también se puede utilizar como el origen de datos, lo que permite crear memorias caché basadas en hub utilizando eXtreme Scale. Un cargador tiene la lógica para leer y escribir datos en un almacén persistente.

Visión general

Los cargadores son plug-ins de correlaciones de respaldo que se invocan cuando se realizan cambios en la correlación de respaldo o ésta no puede satisfacer una solicitud de datos (una falta de memoria caché). Se invoca el cargador cuando la memoria caché no puede satisfacer la solicitud de una clave, proporcionando la capacidad de lectura a través y el relleno poco activo de la memoria caché. Un cargador también permite actualizar la base de datos cuando los valores de la memoria caché cambian. Todos los cambios de una transacción se agrupan para minimizar el número de interacciones de la base de datos. Se utiliza un plug-in TransactionCallback junto con el cargador para desencadenar la demarcación de la transacción de fondo. Utilizar este plug-in es importante cuando se incluyen varias correlaciones en una única transacción, o cuando se desechan los datos de una transacción en la memoria caché sin confirmar.

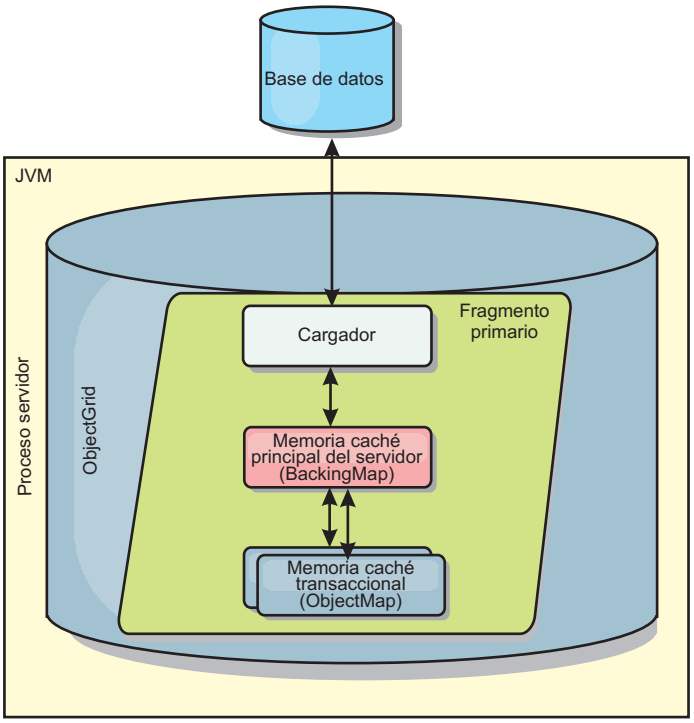


Figura 25. Cargador

El cargador también puede utilizar las actualizaciones sobrecualificadas para evitar mantener los bloqueos de base de datos. Al almacenar un atributo de versión en el valor de memoria caché, el cargador puede ver la imagen antes y después del valor tal como se actualiza en la memoria caché. Este valor se puede utilizar cuando se actualiza la base de datos o cuando se realiza un programa de fondo para verificar que los datos no se han actualizado. Un cargador también se puede configurar para precargar la cuadrícula de datos cuando se inicia. Cuando se realizan particiones, se asocia una instancia de cargador con cada partición. Si la correlación "Company" tiene diez particiones, hay diez instancias de cargador, una por partición primaria. Cuando se activa el fragmento primario de la correlación, se invoca el método preloadMap para el cargador de forma síncrona o asíncrona, que permite cargar automáticamente la partición de la correlación con los datos procedentes del programa de fondo. Cuando se invocan de forma síncrona, todas las transacciones de cliente se bloquean, lo que impide el acceso incoherente a la

cuadrícula de datos. De forma alternativa, se puede utilizar un precargador de cliente para cargar toda la cuadrícula de datos.

Dos cargadores incorporados pueden simplificar en gran medida la integración con los programas de fondo de la base de datos relacional. Los cargadores JPA utilizan las funciones de correlación de objetos relacionales (ORM) de ambas implementaciones, OpenJPA e Hibernate, de la especificación de JPA (Java Persistence API). Si desea más información, consulte "Cargadores JPA" en la página 66.

Si utiliza cargadores en una configuración de varios centros de datos, debe considerar cómo se mantiene la coherencia de los datos y la memoria caché entre las cuadrículas de datos. Para obtener más información, consulte "Consideraciones sobre el cargador en una topología multimaestro" en la página 172.

Configuración de cargador

Para añadir un cargador a la configuración de BackingMap, puede utilizar la configuración mediante programa o la configuración del archivo XML. Un cargador tiene la siguiente relación con una correlación de respaldo.

- Una correlación de respaldo sólo puede tener un cargador.
- Una correlación de respaldo de cliente (memoria caché cercana) no puede tener un cargador.
- Una definición de cargador se puede aplicar a varias correlaciones de respaldo, pero cada una de éstas tiene su propia instancia de cargador.

Precarga de datos y calentamiento

En muchos escenarios que incorporan el uso de un cargador, puede preparar la cuadrícula de datos precargándola con datos.

Cuando se utiliza como una memoria caché completa, la cuadrícula de datos debe alojar todos los datos y se debe cargar antes de que los clientes se puedan conectar a ella. Cuando se utiliza una memoria caché escasa, puede preparar la memoria caché con datos de forma que los clientes tengan acceso inmediato a los datos cuando estos se conecten.

Existen dos enfoques para la precarga de datos en la cuadrícula de datos: mediante un plug-in Loader o mediante un cargador de clientes, tal como se describe en las secciones siguientes.

Plug-in Loader

El plug-in Loader está asociado con cada correlación y es responsable de sincronizar un fragmento de partición primaria con la base de datos. El método preloadMap del plug-in Loader se invoca automáticamente cuando se activa un fragmento. Por ejemplo, si tiene 100 particiones, existen 100 instancias de cargador, y cada una carga los datos para su partición. Si se ejecuta de forma síncrona, todos los clientes se bloquean hasta que se complete la precarga.

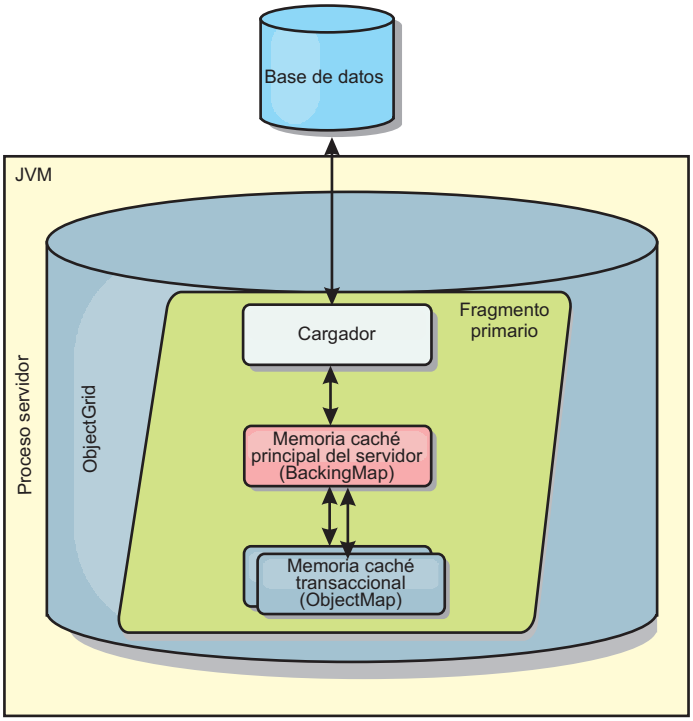


Figura 26. Plug-in Loader

Cargador de clientes

Un cargador de clientes es un patrón para utilizar uno o más clientes para carga la cuadrícula con datos. El uso de varios clientes para cargar los datos de cuadrícula puede ser eficaz cuando el esquema de partición no se almacena en la base de datos. Puede invocar los cargadores de clientes manual o automáticamente cuando se inicia la cuadrícula de datos. De forma opcional, los cargadores de clientes pueden utilizar StateManager para establecer el estado de la cuadrícula de datos en la modalidad de precarga, de forma que los clientes no pueden acceder a la cuadrícula mientras está precargando los datos. WebSphere eXtreme Scale incluye un cargador basado en JPA (Java Persistence API) que puede utilizar para cargar automáticamente la cuadrícula de datos con los proveedores OpenJPA o Hibernate JPA. Para obtener más información sobre los proveedores de memoria caché, consulte “Plug-in de memoria caché de nivel 2 (L2) JPA” en la página 26.

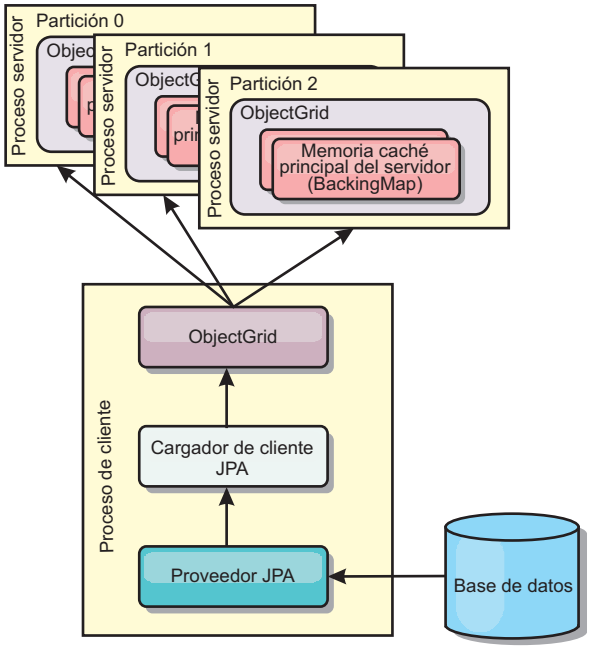


Figura 27. Cargador de clientes

Técnicas de sincronización de base de datos

Cuando se utiliza WebSphere eXtreme Scale como memoria caché, se deben escribir aplicaciones que admitan datos obsoletos si la base de datos puede actualizarse de forma independiente a una transacción de eXtreme Scale. Para servir como un espacio de proceso de base de datos en memoria sincronizado, eXtreme Scale proporciona distintos métodos para mantener la memoria caché actualizada.

Técnicas de sincronización de base de datos

Renovación periódica

La memoria caché se puede invalidar o actualizar de forma automática y periódica utilizando el actualizados de base de datos basado en el tiempo de JPA (Java Persistence API). El actualizador consulta periódicamente la base de datos utilizando un proveedor JPA para cualquier actualización o inserción que se haya producido desde la actualización anterior. Todos los cambios identificados se anulan o actualizan automáticamente cuando se utilizan con una memoria caché escasa. Si se utilizan con una memoria caché completa, las entradas se pueden descubrir e insertar en la memoria caché. Las entradas nunca se eliminan de la memoria caché.

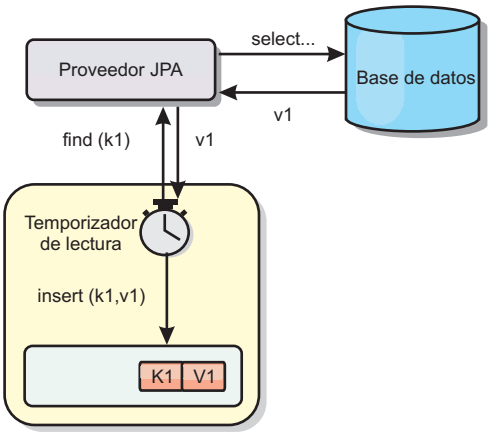


Figura 28. Renovación periódica

Desalojo

Las memorias caché escasas pueden utilizar políticas de desalojo para eliminar automáticamente datos de la memoria caché sin afectar a la base de datos. Existen tres políticas incorporadas incluidas en eXtreme Scale: tiempo de vida, menos usada recientemente y usada con menos frecuencia. Las tres políticas pueden, de forma opcional, desalojar datos de forma más agresiva a medida que la memoria pasa a estar limitada habilitando la opción de desalojo basado en memoria.

Anulación basada en sucesos

Las memorias caché escasas y completas se pueden invalidar o actualizar utilizando un generador de sucesos como, por ejemplo, JMS (Java Message Service). La anulación utilizando JMS puede unirse manualmente a cualquier proceso que actualiza el programa de fondo utilizando un desencadenante de base de datos. Se proporciona un plug-in JMS ObjectGridEventListener en eXtreme Scale que puede notificar a los clientes cuando la memoria caché del servidor tiene algún cambio. Esto puede disminuir la cantidad de tiempo que el cliente puede ver los datos obsoletos.

Anulación programática

Las API eXtreme Scale permiten la interacción manual de la memoria caché cercana y de servidor utilizando los métodos de API Session.beginNoWriteThrough(), ObjectMap.invalidate() y EntityManager.invalidate(). Si un proceso de cliente o servidor ya no necesita una parte de los datos, los métodos de anulación se pueden utilizar para eliminar datos de la memoria caché cercana o del servidor. El método beginNoWriteThrough se aplica cualquier operación ObjectMap o EntityManager a la memoria caché local sin llamar al cargador. Si se invoca desde un cliente, la operación sólo se aplica a la memoria caché cercana (el cargador remoto no se invoca). Si se invoca en el servidor, la operación sólo se aplica a la memoria caché principal del servidor sin invocar el cargador.

Invalidación de datos

Para eliminar los datos de memoria caché de escala, puede utilizar un mecanismo de invalidación basado en suceso o mediante programa.

Invalidación basada en sucesos

Las memorias caché escasas y completas se pueden invalidar o actualizar utilizando un generador de sucesos como, por ejemplo, JMS (Java Message Service). La anulación utilizando JMS puede unirse manualmente a cualquier proceso que actualiza el programa de fondo utilizando un desencadenante de base de datos. Se proporciona un plug-in JMS ObjectGridEventListener en eXtreme Scale que puede notificar a los clientes cuando la memoria caché de servidor cambia. Este tipo de notificación disminuye la cantidad de tiempo que el cliente puede ver los datos obsoletos.

La invalidación basada en sucesos consta normalmente de los tres componentes siguientes.

- **Cola de sucesos:** Una cola de sucesos almacena los sucesos de cambio de datos. Puede ser una cola JMS, una base de datos, una cola FIFO o cualquier clase de siempre que pueda gestionar los sucesos de cambio de datos.
- **Editor de sucesos:** Un editor de sucesos publica los sucesos de cambio de datos en la cola de sucesos. Un editor de sucesos es normalmente una aplicación que usted mismo crea o una implementación de plug-in de eXtreme Scale. El editor de sucesos sabe cuándo se cambian los datos o cambia los datos por sí mismo. Cuando se confirma una transacción, se generan los sucesos para los datos cambiados y el editor de sucesos publica estos sucesos en la cola de sucesos.
- **Consumidor de sucesos:** Un consumidor de sucesos consume sucesos de cambio de datos. El consumidor de sucesos es por lo general una aplicación para garantizar que los datos de la cuadrícula de destino se actualizan con el cambio más reciente de otras cuadrículas. Este consumidor de sucesos interactúa con la cola de sucesos para obtener los cambios de datos más recientes y aplica los cambios de datos en la cuadrícula de destino. Los consumidores de sucesos pueden utilizar las API de eXtreme Scale para invalidar datos obsoletos o actualizar la cuadrícula con los datos más recientes.

Por ejemplo, JMSObjectGridEventListener tiene una opción para un modelo cliente-servidor, en el cual la cola de sucesos es un destino de JMS designado. Todos los procesos del servidor son editores de sucesos. Cuando se confirma una transacción, el servidor obtiene los cambios de datos y los publica en la JMS de destino designada. Todos los procesos de cliente son consumidores de sucesos. Reciben los cambios de datos del destino de JMS designado y aplican los cambios en la memoria caché cercana del cliente.

Consulte el tema sobre la habilitación del mecanismo de invalidación del cliente en la *Guía de administración* si desea más información.

Anulación programática

Las API WebSphere eXtreme Scale permiten la interacción manual de la memoria caché cercana y de servidor utilizando los métodos de API Session.beginNoWriteThrough(), ObjectMap.invalidate() y EntityManager.invalidate(). Si un proceso de cliente o servidor ya no necesita una parte de los datos, los métodos de anulación se pueden utilizar para eliminar datos de la memoria caché cercana o del servidor. El método beginNoWriteThrough se aplica cualquier operación ObjectMap o EntityManager a la memoria caché local sin llamar al cargador. Si se invoca desde un cliente, la operación sólo se aplica a la memoria caché cercana (el cargador remoto no se invoca). Si se invoca en el servidor, la operación sólo se aplica a la memoria caché principal del servidor sin invocar el cargador.

Puede utilizar la anulación mediante programa con otras técnicas para determinar cuándo invalidar los datos. Por ejemplo, este método de invalidación utiliza mecanismos de invalidación basados en sucesos para recibir los sucesos de cambio de datos y luego utiliza interfaces de programación de aplicaciones para invalidar los datos obsoletos.

Índices

Utilice el plug-in MapIndexPlugin para crear un índice o varios índice en una BackingMap para dar soporte al acceso a datos no de clave.

Tipos de índices y configuración

La característica de indexación la representa el plug-in MapIndexPlugin, o Index de forma abreviada. Index es un plug-in BackingMap. Una BackingMap puede tener varios plug-ins Index configurados, siempre que cada uno de ellos siga las normas de configuración de Index.

Puede utilizar la característica de indexación para crear uno o más índices en una BackingMap. Un índice se crea a partir de un atributo o una lista de atributos de un objeto en la BackingMap. De esta manera, las aplicaciones pueden encontrar rápidamente determinados objetos. Con la característica de índices, las aplicaciones pueden encontrar objetos con un valor específico o dentro de un intervalo de valores de atributos indizados.

Existen dos tipos de índice: estático y dinámico. Con el índice estático, debe configurar el plug-in de índices en BackingMap antes de inicializar la instancia de ObjectGrid. Puede realizar esta configuración con una configuración de XML o mediante programa de la BackingMap. Los índices estáticos empiezan a construir un índice durante la inicialización de ObjectGrid. El índice siempre está sincronizado con la BackingMap y listo para ser utilizado. Después de que se inicie el proceso de indexación estática, el mantenimiento del índice forma parte del proceso de gestión de transacciones de eXtreme Scale. Cuando las transacciones confirman cambios, estos cambios también actualizan el índice estático y los cambios de índice se retrotraen si la transacción se retrotrae.

Con el índice dinámico, puede crear un índice en una correlación BackingMap antes o después de la inicialización de la instancia de ObjectGrid que contiene. Las aplicaciones tienen un control del ciclo de vida sobre el proceso de indexación dinámica, de forma que pueda eliminar un índice dinámico, cuando ya no sea necesario. Cuando una aplicación crea un índice dinámico, éste podría no estar listo para su uso inmediato debido al tiempo que tarda en completarse el proceso de creación del índice. Puesto que la cantidad de tiempo depende de la cantidad de datos indexados, se proporciona la interfaz DynamicIndexCallback para aplicaciones que desean recibir notificaciones cuando se produzcan determinados sucesos de indexación. Estos sucesos pueden incluir sucesos de error, destrucción y preparado. Las aplicaciones pueden implementar esta interfaz de devolución de llamada y registrarla con el proceso de índices dinámicos.

Si una BackingMap tiene un plug-in de índice configurado, podrá obtener el proxy de índice de aplicaciones de la ObjectMap correspondiente. Si se llama al método getIndex en la ObjectMap y se proporciona el nombre del plug-in de índice, se devolverá el objeto de proxy de índice. Debe difundir el objeto de proxy de índice en una interfaz apropiada de índice de aplicaciones como, por ejemplo, MapIndex, MapRangeIndex, o una interfaz personalizada de índices. Después de obtener el

objeto de proxy de índice, puede utilizar los métodos definidos en la interfaz de índices de aplicación para buscar objetos almacenados en memoria caché.

En la lista siguiente se resumen los pasos que debe seguir para utilizar los índices:

- Añada plug-ins de índices estáticos o dinámicos a BackingMap.
- Obtenga el objeto de proxy de índice de aplicación; para ello, emita el método getIndex de ObjectMap.
- Difunda el objeto de proxy de índice a una interfaz de índices de aplicación apropiada, como MapIndex, MapRangeIndex, o a una interfaz de índices personalizada.
- Utilice los métodos definidos en una interfaz de índices de aplicación para buscar los objetos almacenados en memoria caché.

La clase HashIndex es la implementación de plug-in de índice que puede soportar ambas interfaces de índice de aplicación incorporadas: MapIndex y MapRangeIndex. También puede crear sus propios índices. Puede añadir HashIndex como un índice estático o dinámico en BackingMap, obtener un objeto proxy de índice MapIndex o MapRangeIndex y utilizar el objeto proxy de índice para encontrar los objetos almacenados en memoria caché.

Índice predeterminado

Si desea iterar a través de las claves en una correlación local, puede utilizar el índice predeterminado. Este índice no requiere ninguna configuración, pero se debe utilizar en el fragmento, utilizando una instancia de ObjectGrid o agente recuperada del método ShardEvents.shardActivated(ObjectGrid shard).

Consideraciones sobre la calidad de los datos

Los resultados de los métodos de consulta de índice sólo representan una instantánea de los datos en un momento puntual. No se obtiene ningún bloqueo contra la entrada de datos después de que los resultados vuelvan a la aplicación. La aplicación tiene que ser consciente de que se pueden producir actualizaciones de datos en un conjunto de datos devuelto. Por ejemplo, la aplicación obtiene la clave de un objeto almacenado en memoria caché ejecutando el método findAll de MapIndex. Este objeto de clave devuelto se asocia a una entrada de datos de la memoria caché. La aplicación debe poder ejecutar el método get en ObjectMap para encontrar un objeto proporcionando el objeto de clave. Si otra transacción elimina el objeto de datos de la memoria caché, justo antes de que se llame al método get, el resultado devuelto será nulo.

Consideraciones sobre el rendimiento de los índices

Uno de los principales objetivos de la característica de índices es mejorar el rendimiento global de BackingMap. Si los índices no se utilizan correctamente, podría verse afectado el rendimiento de la aplicación. Tenga en cuenta los siguientes factores antes de utilizar esta característica.

- **El número de transacciones de escritura simultáneas:** el proceso de índices se puede producir cada vez que una transacción escribe datos en una BackingMap. El rendimiento disminuye si hay muchas transacciones grabando datos en una correlación al mismo tiempo que una aplicación realiza operaciones de consulta de índices.
- **El tamaño del conjunto de resultados devuelto por una operación de consulta:** a medida que el tamaño del conjunto de resultados aumenta, el rendimiento de

la consulta disminuye. El rendimiento tiene tendencia a disminuir si el tamaño del conjunto de resultados es un 15% o más de la BackingMap.

- **El número de índices creados sobre la misma BackingMap:** cada índice consume recursos del sistema. A medida que el número de índices creados sobre la BackingMap aumenta, disminuye el rendimiento.

La función de indexación puede mejorar el rendimiento de BackingMap de forma drástica. Los casos ideales se producen cuando la BackingMap tiene operaciones básicamente de lectura, el conjunto de resultados de la consulta es un pequeño porcentaje de las entradas de BackingMap, y sólo se crean unos pocos índices sobre la BackingMap.

Cargadores JPA

Java Persistence API (JPA) es una especificación que permite la correlación de objetos Java con bases de datos relacionales. JPA contiene una especificación de correlación de objetos relacionales (ORM) completa que utiliza las anotaciones de metadatos de lenguaje Java, los descriptores XML, o ambos, para definir la correlación entre los objetos Java y una base de datos relacional. Hay diversas implementaciones de código abierto y comerciales disponibles.

Puede utilizar una implementación de un plug-in de cargador de Java Persistence API (JPA) con eXtreme Scale para interactuar con cualquier base de datos soportada por su cargador elegido.Para utilizar JPA, debe tener un proveedor JPA soportado como, por ejemplo, OpenJPA o Hibernate, archivos JAR y un archivo `META-INF/persistence.xml` en la classpath.

Los plug-ins de JPALoader com.ibm.websphere.objectgrid.jpa.JPALoader y JPAEntityLoader com.ibm.websphere.objectgrid.jpa.JPAEntityLoader son dos plug-ins del cargador JPA incorporados que se utilizan para sincronizar las correlaciones de ObjectGrid con una base de datos. Debe tener una implementación JPA como, por ejemplo, Hibernate o OpenJPA, para utilizar esta característica. La base de datos puede ser cualquier programa de fondo soportado por el proveedor JPA elegido.

Puede utilizar el plug-in JPALoader al almacenar datos utilizando la API ObjectMap. Utilice el plug-in JPAEntityLoader al almacenar los datos utilizando la API EntityManager.

Arquitectura del cargador JPA

El cargador JPA se utiliza para las correlaciones de eXtreme Scale que almacenan los objetos POJO (Plain Old Java Object).

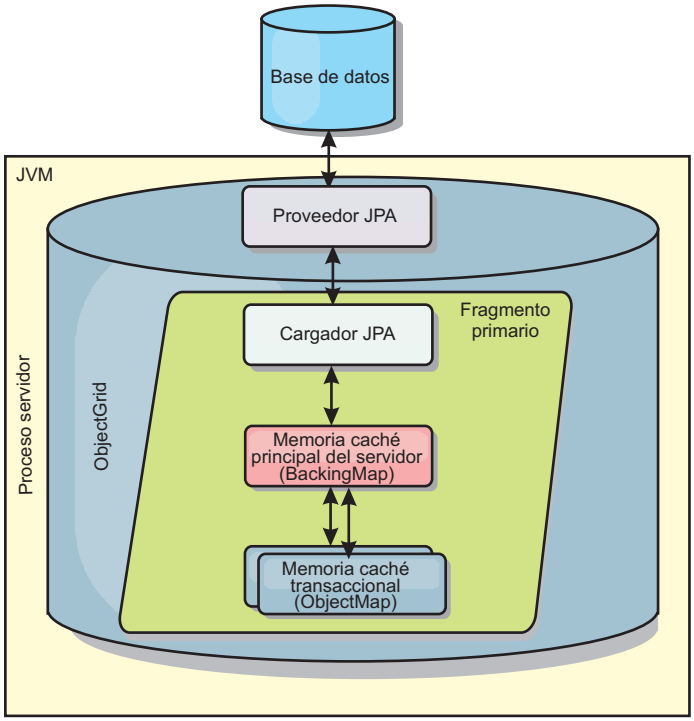


Figura 29. Arquitectura del cargador JPA

Cuando se llama un método ObjectMap.get(Object key), eXtreme Scale comprueba primero si la entrada se incluye en la capa ObjectMap. En caso negativo, el tiempo de ejecución delega la solicitud al cargador JPA. Después de solicitar la carga de la clave, JPALoader llama el método JPA EntityManager.find(Object key) para encontrar los datos de la capa JPA. Si los datos están contenidos en el gestor de entidades JPA, se devuelve; de lo contrario, el proveedor JPA interactúa con la base de datos para obtener el valor.

Cuando se produce una actualización en ObjectMap, por ejemplo, mediante el uso del método ObjectMap.update(Object key, Object value), el tiempo de ejecución de eXtreme Scale crea un LogElement para esta actualización y lo envía a JPALoader. JPALoader llama el método JPA EntityManager.merge(Object value) para actualizar el valor en la base de datos.

En JPAEntityLoader, también están implicadas las mismas cuatro capas. Sin embargo, dado que se utiliza el plug-in JPAEntityLoader para las correlaciones que almacenan las entidades de eXtreme Scale, las relaciones entre las entidades podrían complicar el escenario de uso. Se distingue una entidad eXtreme Scale de la entidad JPA. Para obtener más detalles, consulte la información sobre el plug-in JPAEntityLoader en la *Guía de programación*.

Métodos

Los cargadores proporcionan tres métodos principales:

1. get: devuelve una lista de valores que corresponden a la lista de claves que se pasan recuperando los datos que utilizan JPA. El método utiliza JPA para encontrar las entidades en la base de datos. Para el plug-in JPALoader, la lista devuelva contiene una lista de entidades JPA directamente de la operación de

búsqueda. Para el plug-in JPAEntityLoader, la lista devuelta contiene los tuples de valor de entidad eXtreme Scale que se han convertido a partir de las entidades JPA.

2. batchUpdate: graba los datos de las correlaciones ObjectGrid en la base de datos. En función de los distintos tipos de operación (insertar, actualizar o suprimir), el cargador utiliza las operaciones de persistir, fusionar y eliminar de JPA para actualizar los datos en la base de datos. En el caso de JPALoader, los objetos de la correlación se utilizan directamente como entidades JPA. En el caso de JPAEntityLoader, los tuples de entidad de la correlación se convierten en objetos que se utilizan como entidades JPA.
3. preloadMap: precarga la correlación utilizando el método de cargador de cliente ClientLoader.load. Para las correlaciones con particiones, sólo se llama al método preloadMap en una partición. La partición se especifica en la propiedad preloadPartition de la clase JPALoader o JPAEntityLoader. Si el valor preloadPartition se establece en un valor menor que cero, o mayor que value (*número_total_de_particiones* - 1), se inhabilita la precarga.

Ambos plug-ins, JPALoader y JPAEntityLoader, trabajan con la clase JPATxCallback para coordinar las transacciones eXtreme Scale y las transacciones JPA. Se debe configurar JPATxCallback en la instancia de ObjectGrid para utilizar estos dos cargadores.

Configuración y programación

Si está utilizando cargadores JPA en un entorno multimaestro, consulte "Consideraciones sobre el cargador en una topología multimaestro" en la página 172. Para obtener más información sobre cómo configurar cargadores JPA, consulte la información sobre los cargadores JPA en la *Guía de administración*. Si desea más información sobre cómo programar cargadores JPA, consulte *Guía de programación*.

Visión general de serialización

Los datos normalmente se expresan, pero no se almacenan necesariamente, como objetos Java en la cuadrícula de datos. WebSphere eXtreme Scale utiliza varios procesos Java para serializar los datos, convirtiendo las instancias de objeto Java en bytes y de nuevo en objetos, según se requiera, para mover datos entre procesos de cliente y servidor.

Los datos que se serializan se convierten en una secuencia de datos para transmitirse a través de una red, en las situaciones siguientes:

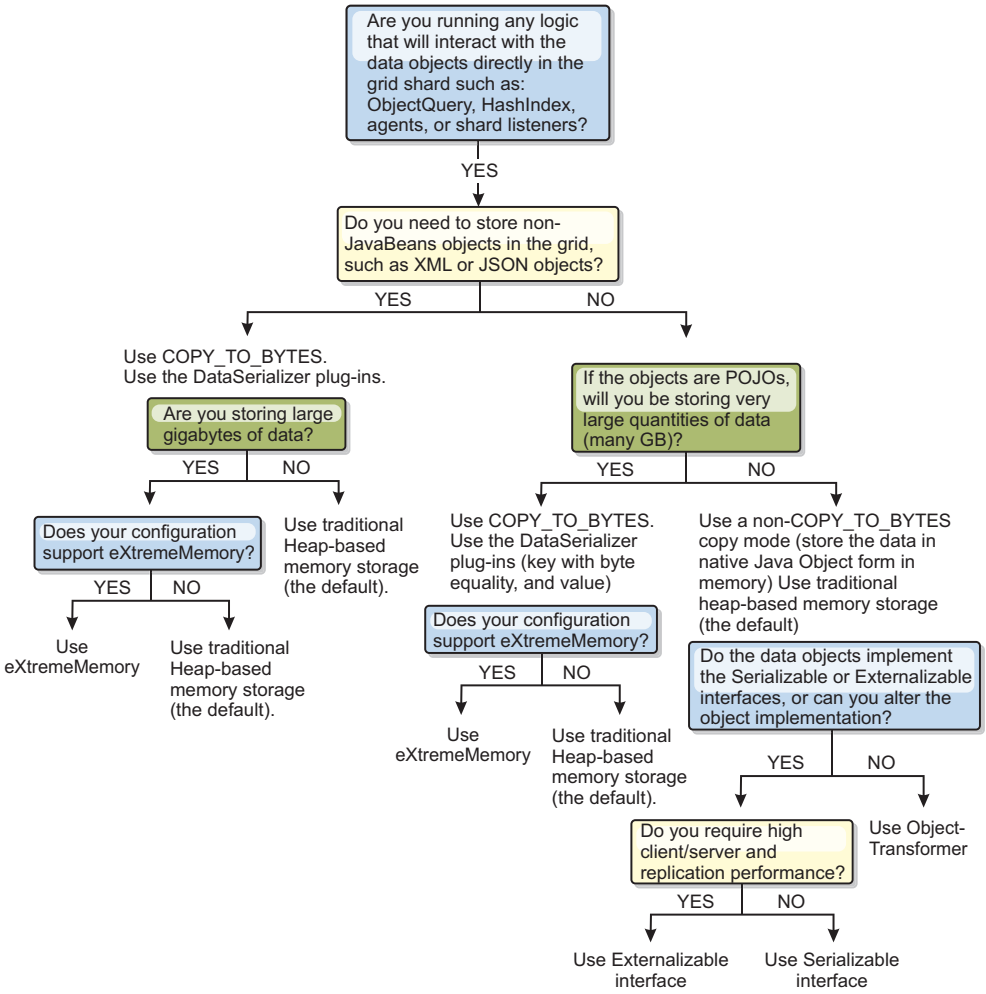
- Cuando los clientes se comunican con los servidores, y estos servidores envían información de nuevo al cliente
- Cuando los servidores replican datos de un servidor a otro

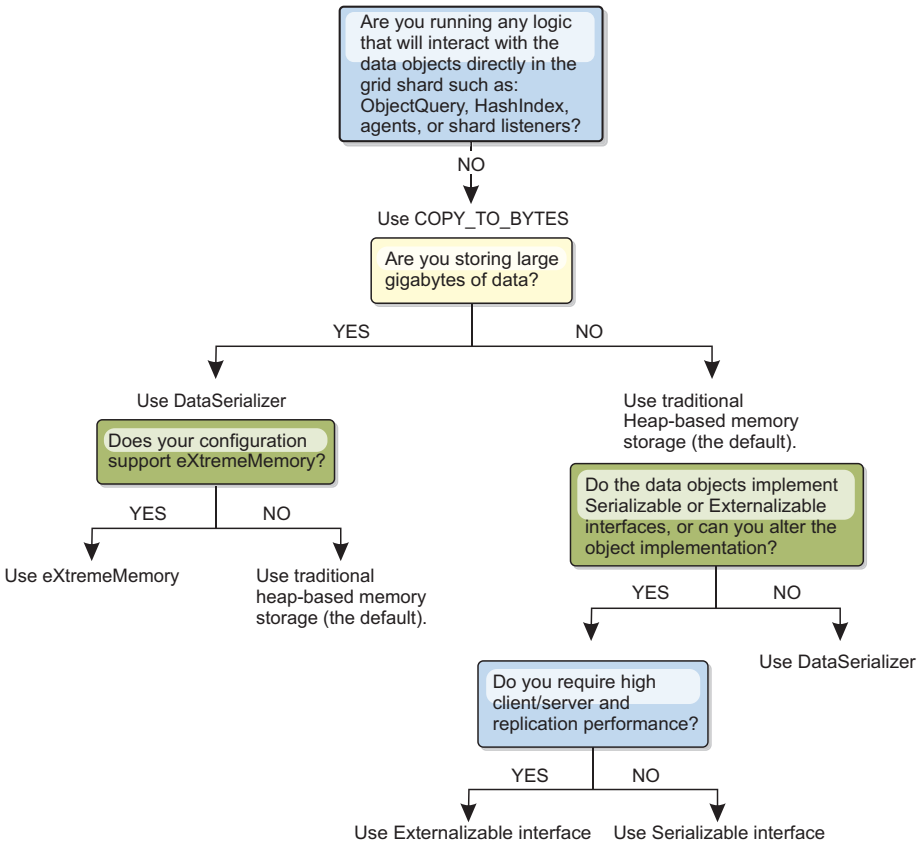
Opcionalmente, podría decidir omitir el proceso de serialización en WebSphere eXtreme Scale y almacenar datos en bruto como matrices de bytes. Las matrices de bytes son mucho menos costosas de almacenar en memoria ya que la JVM (Java Virtual Machine) tiene menos objetos que buscar durante la recogida de basura y se pueden aumentar sólo cuando es necesario. Utilice solo matrices de bytes si no necesita acceder a los objetos mediante consultas o índices. Puesto que los datos se almacenan como bytes, eXtreme Scale no tiene metadatos para describir los atributos a consultar.

Para serializar datos en eXtreme Scale, puede utilizar serialización Java, el plug-in ObjectTransformer o los plug-ins DataSerializer. Para optimizar la serialización con

cualquiera de estas opciones, puede utilizar la modalidad COPY_TO_BYTES para mejorar el rendimiento hasta un 70 por ciento porque los datos se serializan cuando se confirman las transacciones, lo que significa que la serialización solo tiene lugar una vez. Los datos serializados se envían sin cambios desde el cliente al servidor o desde el servidor al servidor replicado. Mediante la modalidad COPY_TO_BYTES, puede reducir la huella de memoria que puede consumir un gráfico grande de objetos.

Utilice las figuras siguientes como ayuda para determinar qué tipo de método de serialización es más apropiado para las necesidades de desarrollo. La primera figura describe los métodos de serialización que están disponibles cuando se ejecuta la lógica que interactúa con objetos de datos directamente en el fragmento de cuadrícula. La última cifra muestra las opciones disponibles cuando no se interactúa directamente el fragmento de cuadrícula.





Para obtener más información sobre las formas soportadas de serialización en el producto eXtreme Scale, consulte los temas siguientes:

Serialización mediante Java

La serialización de Java hace referencia a la serialización predeterminada, que utiliza la interfaz serializable, o serialización personalizada, que utiliza tanto la interfaz serializable como la interfaz externalizable.

Serialización predeterminada

Para utilizar la serialización predeterminada, implemente la interfaz java.io.Serializable, que incluye la API que convierte objetos en bytes, que posteriormente se deserializan. Utilice la clase java.io.ObjectOutputStream para persistir el objeto. A continuación, llame al método ObjectOutputStream.writeObject() para iniciar la serialización y "aplanar" el objeto Java.

Serialización personalizada

Hay algunos casos en los que deben modificarse los objetos para utilizar la serialización personalizada, como la implementación de la interfaz java.io.Externalizable o al implementar los métodos writeObject y readObject para las clases que implementan la interfaz java.io.Serializable. Las técnicas de serialización personalizada deben emplearse cuando se serializan los objetos mediante mecanismos que no sean los métodos de la API ObjectGrid o la API EntityManager.

Por ejemplo, cuando los objetos o las entidades se almacenan como datos de instancia en un agente de la API DataGrid o el agente devuelve objetos o entidades, dichos objetos no se transforman mediante ObjectTransformer. El agente utilizará automáticamente ObjectTransformer al utilizar la interfaz `EntityMixin`. Si desea obtener más información, consulte el tema Agentes DataGrid y correlaciones basadas en entidades

Plug-in ObjectTransformer

Con el plug-in ObjectTransformer puede serializar, deserializar y copiar objetos en la memoria caché para mejorar el rendimiento.

 La interfaz ObjectTransformer ha sido sustituida por los plug-ins DataSerializer, que puede utilizar para almacenar eficientemente datos arbitrarios en WebSphere eXtreme Scale de modo que las API existentes del producto puedan interactuar eficazmente con los datos.

Si observa problemas de rendimiento con el uso del procesador, añada un plug-in ObjectTransformer a cada correlación. Si no proporciona un plug-in ObjectTransformer, se emplea entre un 60 y un 70 por ciento del tiempo total de procesador en serializar y copiar entradas.

Finalidad

Con el plug-in ObjectTransformer, las aplicaciones pueden proporcionar métodos personalizados para las siguientes operaciones:

- Serializar o deserializar la clave de una entrada.
- Serializar o deserializar el valor de una entrada.
- Copiar una clave o valor de una entrada.

Si no se proporciona ningún plug-in ObjectTransformer, debe poder serializar las claves y los valores porque ObjectGrid utiliza una secuencia de serialización y deserialización para copiar los objetos. Éste es un método costoso, por lo que conviene utilizar un plug-in ObjectTransformer si el rendimiento es crítico. La operación de copia se produce cuando una aplicación busca un objeto en una transacción por primera vez. Puede evitar esta copia si establece la modalidad de copia de la correlación en NO_COPY o puede reducir la copia si establece la modalidad de copia en COPY_ON_READ. Optimice la operación de copia cuando sea necesario para la aplicación; para ello, proporcione un método de copia personalizada en este plug-in. Dicho plug-in puede reducir la sobrecarga de copia del 65 al 70 por ciento a 2/3 del porcentaje del tiempo total del procesador.

Las implementaciones predeterminadas del método copyKey y copyValue intentan, en primer lugar, utilizar el método clone, si se ha proporcionado el método. Si no se ha proporcionado ninguna implementación del método clone, la implementación toma el valor predeterminado de la serialización.

La serialización de objetos también se utiliza directamente cuando eXtreme Scale se ejecuta en la modalidad distribuida. El LogSequence utiliza el plug-in ObjectTransformer para ayudar a serializar claves y valores antes de transmitir los cambios a sus iguales en ObjectGrid. Debe actuar con detenimiento cuando proporcione un método de serialización personalizado, en lugar de utilizar la serialización del Java Developer Kit incorporada. La creación de versiones de

objetos es un asunto complejo y podría tener problemas con la compatibilidad de las versiones si no se asegura de que sus métodos personalizados se han diseñado para la creación de versiones.

La siguiente lista describe cómo eXtreme Scale intenta serializar tanto las claves, como los valores:

- Si se ha escrito y conectado un plug-in ObjectTransformer personalizado, eXtreme Scale llama a los métodos de la interfaz ObjectTransformer para serializar las claves y los valores y obtener copiadas de claves y valores de objeto.
- Si no se utiliza un plug-in ObjectTransformer personalizado, eXtreme Scale serializa y deserializa los valores de acuerdo con el valor predeterminado. Si se utiliza el plug-in predeterminado, cada objeto se implementa como externalizable o se implementa como serializable.
  - Si el objeto soporta la interfaz Externalizable, se llama al método writeExternal. Los objetos que se implementan como externalizables obtienen un mejor rendimiento.
  - Si el objeto no soporta la interfaz Externalizable e implementa la interfaz Serializable, el objeto se guarda mediante el método ObjectOutputStream.

Utilización de la interfaz ObjectTransformer

Un objeto ObjectTransformer debe implementar la interfaz ObjectTransformer y seguir las convenciones comunes de plug-in de ObjectGrid.

Se utilizan dos enfoques, configuración mediante programa y configuración de XML, para añadir un objeto ObjectTransformer a la configuración de BackingMap tal como se indica a continuación.

Conexión mediante programación de un objeto ObjectTransformer

El siguiente fragmento de código crea el objeto ObjectTransformer personalizado y lo añade a un BackingMap:

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

Conexión de ObjectTransformer mediante la configuración del XML

Imagine que el nombre de clase de la implementación ObjectTransformer es la clase com.company.org.MyObjectTransformer. Esta clase implementa la interfaz ObjectTransformer. Una implementación de ObjectTransformer se puede configurar utilizando el siguiente 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>
```

Escenarios de uso de ObjectTransformer

Puede utilizar un plug-in ObjectTransformer en las situaciones siguientes:

- Objeto no serializable
- Objeto serializable pero mejora el rendimiento de serialización
- Copia de clave o valor

En el ejemplo siguiente, se utiliza ObjectGrid para almacenar la clase Stock:

```
/**
* Objeto Stock para ObjectGrid
*
*
*/
public class Stock implements Cloneable {
    String ticket;
    double price;
    String company;
    String description;
    int serialNumber;
    long lastTransactionTime;
    /**
    * @return Devuelve la descripción.
    */
    public String getDescription() {
        return description;
    }
    /**
    * @param description La descripción que se debe establecer.
    */
    public void setDescription(String description) {
        this.description = description;
    }
    /**
    * @return Devuelve lastTransactionTime.
    */
    public long getLastTransactionTime() {
        return lastTransactionTime;
    }
    /**
    * @param lastTransactionTime El lastTransactionTime a establecer.
    */
    public void setLastTransactionTime(long lastTransactionTime) {
        this.lastTransactionTime = lastTransactionTime;
    }
    /**
    * @return Devuelve el precio.
    */
    public double getPrice() {
        return price;
    }
    /**
    * @param price El precio a establecer.
    */
    public void setPrice(double price) {
        this.price = price;
    }
    /**
    * @return Devuelve serialNumber.
    */
    public int getSerialNumber() {
        return serialNumber;
    }
    /**
    * @param serialNumber El serialNumber a establecer.
    */
    public void setSerialNumber(int serialNumber) {
        this.serialNumber = serialNumber;
    }
    /**
    * @return Devuelve el ticket.
```

```
    */
    public String getTicket() {
        return ticket;
    }
    /**
    * @param ticket El ticket a establecer.
    */
    public void setTicket(String ticket) {
        this.ticket = ticket;
    }
    /**
    * @return Devuelve la empresa.
    */
    public String getCompany() {
        return company;
    }
    /**
    * @param company La empresa a establecer.
    */
    public void setCompany(String company) {
        this.company = company;
    }
    //clone
    public Object clone() throws CloneNotSupportedException
    {
        return super.clone();
    }
}
```

Puede escribir una clase de ObjectTransformer para la clase Stock:

```
/**
* Implementación personalizada de ObjectGrid ObjectTransformer para el objeto stock
*
*/
public class MyStockObjectTransformer implements ObjectTransformer {
/* (no 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);
}

/* (no 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());
}

/* (no 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;
}

/* (no 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;
}
```

```
/* (no 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)
    {
        // mostrar mensaje de excepción }
}

/* (no 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;
}
}
```

A continuación, conecte esta clave MyStockObjectTransformer personalizada a BackingMap:

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

Serialización mediante plug-ins DataSerializer

Utilice los plug-ins DataSerializer para almacenar eficazmente datos arbitrarios en WebSphere eXtreme Scale de forma que las API existentes del producto puedan interactuar eficazmente con los datos.

Los métodos de serialización como por ejemplo la serialización de Java y el plug-in de ObjectTransformer permiten que los datos se clasifiquen a través de la red. Además, cuando se utilizan estas opciones de serialización con la modalidad de copia COPY_TO_BYTES, el traspaso de datos entre clientes y servidores se vuelve menos costoso y el rendimiento mejora. Sin embargo, estas opciones no solucionan los siguientes problemas que pueden existir:

- Las claves no se almacenan en bytes; todavía son objetos Java.
- El código del lado del servidor todavía debe aumentar el objeto; por ejemplo, la consulta y el índice aún utilizan reflejo y deben aumentar el objeto. Adicionalmente, los agentes, los escuchas y los plug-ins aún necesitan el formato de objeto.
- Sigue siendo necesario que las clases estén en la classpath del servidor.
- Los datos siguen estando en formato de serialización de Java (ObjectOutputStream).

El plug-in DataSerializer presenta una manera eficaz de solucionar estos problemas. Específicamente, el plug-in DataSerializer le proporciona una forma de describir el formato de serialización, o una matriz de bytes, en WebSphere eXtreme Scale ara que el producto pueda interrogar la matriz de bytes sin necesidad de un objeto de formato específico. Las clases y las interfaces del plug-in DataSerializer están en el paquete, com.ibm.websphere.objectgrid.plugins.io. Si desea más información, consulte la .

Importante: Los objetos Java de entidad no se almacenan directamente en BackingMaps cuando se utiliza la API EntityManager. La API EntityManager convierte el objeto de entidad en objetos Tuple. Las correlaciones de entidad se asocian automáticamente con un objeto ObjectTransformer altamente optimizado.

Siempre que se utiliza la API ObjectMap o EntityManager para interactuar con correlaciones de entidad, se invoca a la entidad ObjectTransformer. Por lo tanto, cuando se utilizan entidades, no se requiere ningún trabajo para la serialización porque el producto completa automáticamente este proceso.

Visión general de la escalabilidad

WebSphere eXtreme Scale se puede escalar a través del uso de datos particiones y puede escalar miles de contenedores si es necesario porque cada contenedor es independiente de otros contenedores.

WebSphere eXtreme Scale divide los conjuntos de datos en distintas particiones que se pueden mover entre procesos o incluso entre servidores físicos en tiempo de ejecución. Por ejemplo, puede empezar con un despliegue de cuatro servidores y, a continuación, ampliar a un despliegue con diez servidores a medida que crece la demanda en la memoria caché. Exactamente de la misma forma como puede añadir más servidores físicos y unidades de proceso para la escalabilidad vertical, puede ampliar la capacidad de escalabilidad elástica de forma horizontal con el particionamiento. El escalado horizontal es una ventaja importante de la utilización de WebSphere eXtreme Scale frente a una base de datos en memoria. Las bases de datos en memoria solo se pueden escalar verticalmente.

Con WebSphere eXtreme Scale, también puede utilizar un conjunto de API para obtener acceso transaccional a estos datos particionados y distribuidos. Desde la perspectiva del rendimiento, las selecciones que realice para interactuar con la memoria caché son tan importantes como las funciones para gestionar la memoria caché para la disponibilidad.

Nota: La escalabilidad no está disponible cuando los contenedores se comunican entre sí. El protocolo de la gestión de la disponibilidad, o de la agrupación principal, es un algoritmo de vista y pulsación $O(N^2)$, pero se mitiga manteniendo el número de miembros de principal de forma que no sea superior a 20. Sólo existe la réplica de igual a igual entre fragmentos.

Clientes distribuidos

El protocolo del cliente de WebSphere eXtreme Scale soporta una gran cantidad de clientes. La estrategia de particionamiento ofrece asistencia dando por supuesto que todos los clientes no siempre están interesado en todas las particiones porque las conexiones pueden esparcirse por varios contenedores. Los clientes se conectan directamente a las particiones, por lo que la latencia se limita a una conexión transferida.

Cuadrículas de datos, particiones y fragmentos

Una cuadrícula de datos se divide en particiones. Una partición contiene un subconjunto exclusivo de los datos. Una partición contiene uno o más fragmentos: un fragmento primario y fragmentos de réplica. Los fragmentos de réplica no están necesariamente en una partición, pero puede utilizarlos para proporcionar alta disponibilidad. Si su despliegue es una cuadrícula de datos en memoria independiente o un espacio de proceso de base de datos en memoria, el acceso a los datos en eXtreme Scale se basa en gran medida en fragmentos.

Los datos de una partición se almacenan en un conjunto de fragmentos en tiempo de ejecución. Este conjunto de fragmentos incluye un fragmento primario y

posiblemente uno más fragmentos réplica. Un fragmento es la unidad más pequeña que eXtreme Scale puede añadir a una Máquina virtual Java o eliminar de ésta.

Existen dos estrategias de colocación: colocación de partición fija (valor predeterminado) y colocación por contenedor. El tema siguiente se centra en el uso de la estrategia de colocación de partición fija.

Número total de fragmentos

Si el entorno incluye 10 particiones que contienen 1 millón de objetos sin réplicas, existirán 10 fragmentos, cada uno de los cuales almacenará 100.000 objetos. Si añadiera una réplica a este escenario, existiría un fragmento adicional en cada partición. En este caso, existen 20 fragmentos: 10 fragmentos primarios y 10 fragmentos de réplica. Cada uno de estos fragmentos almacena 100.000 objetos. Cada partición se compone de un fragmento primario y uno o más (N) fragmentos réplica. La determinación del número óptimo de fragmentos es clave. Si configura pocos fragmentos, los datos no se distribuyen equitativamente entre los fragmentos, lo que produce errores de memoria insuficiente y problemas de sobrecarga del procesador. Para escalar, debe tener como mínimo 10 fragmentos para cada JVM . Al desplegar inicialmente la cuadrícula de datos, utilizaría potencialmente muchas particiones.

Escenarios de número de fragmentos por JVM

Escenario: número pequeño de fragmentos para cada JVM

Los datos se añaden a una JVM y se eliminan de ésta mediante unidades de fragmentos. Los fragmentos nunca se dividen en partes. Si existen 10 GB de datos y existen 20 fragmentos para alojar estos datos, cada fragmento incluye de media 500 MB de datos. Si nueve Mäquinas virtuales Java alojan la cuadrícula de datos, de promedio cada JVM tiene dos fragmentos. Como 20 no es divisible entre 9, unas cuantas Mäquinas virtuales Java tienen tres fragmentos, con la distribución siguiente:

- Siete Mäquinas virtuales Java con dos fragmentos
- Dos Mäquinas virtuales Java con tres fragmentos

Puesto que cada fragmento incluye 500 MB de datos, la distribución de datos no es uniforme. Las 7 Mäquinas virtuales Java con 2 fragmentos contienen cada uno 1 GB de datos. Las 2 Mäquinas virtuales Java con 3 fragmentos tienen el 50% más de datos, o 1,5 GB, que es una carga de memoria mucho mayor. Dado que los dos Mäquinas virtuales Java alojan tres fragmentos, estos reciben 50% más solicitudes para sus datos. Como resultado, si se tienen pocos fragmentos para cada JVM se produce desequilibrio. Para aumentar el rendimiento, debe incrementar el número de fragmentos para cada JVM .

Escenario: número mayor de fragmentos por JVM

Para este escenario el número de fragmentos es mucho mayor. En este escenario, existen 101 fragmentos con nueve Mäquinas virtuales Java que alojan 10 GB de datos. En este caso, cada fragmento contiene 99 MB de datos. La distribución de fragmentos de las Mäquinas virtuales Java es la siguiente:

- Siete Mäquinas virtuales Java con 11 fragmentos
- Dos Mäquinas virtuales Java con 12 fragmentos

Las 2 Mäquinas virtuales Java con 12 fragmentos tienen sólo 99 MB más de datos que los otros fragmentos, que es una diferencia del 9%. Este escenario se distribuye mucho más equitativamente que la diferencia del 50% en este escenario con pocos fragmentos. Desde una perspectiva de uso del procesador, solo existe un 9% de trabajo adicional para las dos Mäquinas virtuales Java con los 12 fragmentos en comparación con las siete Mäquinas virtuales Java que tienen 11 fragmentos. Al incrementar el número de fragmentos en cada JVM , los datos y el uso del procesador se distribuyen de manera más equilibrada y uniforme.

Al crear el sistema, use 10 fragmentos para cada JVM como escenario de tamaño máximo, o cuando el sistema está ejecutando su número máximo de Mäquinas virtuales Java en el horizonte de planificación.

Factores adicionales de ubicación

El número de particiones, la estrategia de colocación, y el número y tipo de réplicas se establecen en la política de despliegue. El número de fragmentos que se colocan depende de la política de despliegue que define. Los atributos minSyncReplicas, developmentMode, maxSyncReplicas y maxAsyncReplicas afectan al lugar dónde se ponen las particiones y las réplicas.

Los factores siguientes afectan al momento en que se pueden poner los fragmentos:

- Los mandatos **xscmd -c suspendBalancing** y **xscmd -c resumeBalancing**.
- 7.1.1+ El archivo de propiedades de servidor, que tiene la propiedad **placementDeferralInterval** que define el número de milisegundos antes de que los fragmentos se pongan en los servidores de contenedor.
- El atributo **numInitialContainers** en la política de despliegue.

Si no se coloca el número máximo de réplicas durante el inicio inicial, se podrían colocar réplicas adicionales si se inician posteriormente servidores adicionales. Al planificar el número de fragmentos por JVM, el número máximo de fragmentos primarios y de réplica depende de tener suficientes JVM iniciadas para dar soporte al número máximo de réplicas. Una réplica nunca se coloca en el mismo proceso que su primario. Si se pierde un proceso, se perderá tanto el primario como la réplica. Cuando el atributo developmentMode se establece en `false`, el primario y las réplicas no se ponen en el mismo servidor físico.

Particionamiento

Utilice el particionamiento para escalar una aplicación. Puede definir el número de particiones en la política de despliegue.

Acerca del particionamiento

El particionamiento no es como la escritura en bandas de RAID (Redundant Array of Independent Disks), que corta cada instancia a lo largo de todas las bandas. Cada partición aloja los datos completos para las entradas individuales. El particionamiento es un medio muy eficaz para escalar, pero no puede aplicarse a todas las aplicaciones. Las aplicaciones que requieren garantías transacciones entre grandes conjuntos de datos no se escalan y no se pueden particionar eficazmente. WebSphere eXtreme Scale no da soporte actualmente a la confirmación de dos fases entre particiones.

Importante: Seleccione el número de particiones cuidadosamente. El número de particiones definidas en la política de despliegue afecta directamente al número de servidores de contenedor al que se puede escalar una aplicación. Cada partición

está compuesta de un fragmento primario y el número configurado de fragmentos de réplica. La fórmula (Número_Particiones*(1 + Número_Réplicas)) es el número de contenedores que se pueden utilizar para escalar de forma horizontal una única aplicación.

Uso de particiones

Una cuadrícula de datos puede tener hasta miles de particiones. Una cuadrícula de datos se puede escalar hasta el número resultante de la multiplicación del número de particiones por el número de fragmentos por partición. Por ejemplo, si tiene 16 particiones y cada partición tiene un primario y una réplica, o dos fragmentos, podrá escalarla de forma potencial a 32 Máquinas virtuales Java. En este caso, se define un fragmento para cada JVM. Debe elegir un número razonable de particiones basándose en el número esperado de Máquinas virtuales Java que probablemente vaya a utilizar. Cada fragmento aumenta el uso de procesador y memoria para el sistema. El sistema se ha diseñado para escalar de forma horizontal para manejar esta sobrecarga según el número disponible de Máquinas virtuales Java de servidor.

Las aplicaciones no deben utilizar miles de particiones si la aplicación se ejecuta en una cuadrícula de datos de cuatroMáquinas virtuales Java de servidor de contenedor. La aplicación se debe configurar para que tenga un número razonable de fragmentos para cada JVM de servidor de contenedor. Por ejemplo, una configuración no razonable sería establecer 2000 particiones con dos fragmentos que se ejecutan en cuatro Máquinas virtuales Java de contenedor. Esta configuración genera 4000 fragmentos que se colocan en cuatro Máquinas virtuales Java de contenedor o 1000 fragmentos por JVM de contenedor.

Por el contrario, una mejor configuración sería tener menos de 10 fragmentos para cada JVM de contenedor esperado. Esta configuración todavía proporciona la posibilidad de permitir una escalada elástica que es diez veces la configuración inicial, mientras que se mantiene un número razonable de fragmentos por JVM de contenedor.

Considere este ejemplo de escalado: actualmente tiene seis servidores físicos con dos Máquinas virtuales Java de contenedor por servidor físico. Espera que crezca hasta 20 servidores físicos a lo largo de los próximos tres años. Con 20 servidores físicos, tiene 40 Máquinas virtuales Java de servidor de contenedor y elige 60 para ser pesimista. Desea tener cuatro fragmentos por JVM de contenedor. Potencialmente tiene 60 contenedores o un total de 240 fragmentos. Si tiene un fragmento primario y una réplica por partición, tendrá 120 particiones. Este ejemplo le proporciona 240 dividido por 12 Máquinas virtuales Java de contenedor, o 20 fragmentos por JVM de contenedor para el despliegue inicial con el potencial de poder escalar de forma horizontal a 20 máquinas, más adelante.

ObjectMap y particionamiento

Con la estrategia de colocación FIXED_PARTITION predeterminada, las correlaciones se dividen entre las particiones y las claves realizan un método hash en particiones diferentes. No es necesario que el cliente sepa a qué partición pertenecen las claves. Si un mapSet tiene varias correlaciones, las correlaciones se deben confirmar en transacciones distintas.

Entidades y particionamiento

Las entidades del gestor de entidades tienen una optimización que ayuda a los clientes que trabajan con entidades en un servidor. El esquema de entidad en el servidor relativo al conjunto de correlaciones puede especificar una sola entidad raíz. El cliente debe acceder a todas las entidades a través de la entidad raíz. El gestor de entidades puede encontrar las entidades relacionadas en dicha raíz en la misma partición sin necesitar que las correlaciones relacionadas tengan una clave común. La entidad raíz establece afinidad con una única partición. Esta partición se utiliza en todas las búsquedas de entidad dentro de la transacción una vez establecida la afinidad. Esta afinidad puede ahorrar memoria porque las correlaciones relacionadas no necesitan una clave común. La entidad raíz debe especificarse con una anotación de entidad modificada, como se muestra en el ejemplo siguiente:

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

Utilice la entidad para encontrar la raíz del gráfico del objeto. El gráfico del objeto define la relación entre una o varias entidades. Cada entidad enlazada se debe resolver con la misma partición. Se presupone que todas las entidades hija están en la misma partición que la raíz. Sólo se puede acceder a las entidades hija de este gráfico del objeto desde un cliente de la entidad raíz. Las entidades raíz siempre son necesarias en entornos con particiones si se utiliza un cliente eXtreme Scale para comunicarse con el servidor. Sólo se puede definir un tipo de entidad raíz por cliente. Las entidades raíz no son necesarias cuando se utilizan ObjectGrdi de estilo XTP (Extreme Transaction Processing), ya que todas las comunicaciones a la partición se realizan mediante acceso local directo y no mediante el mecanismo de cliente y servidor.

Colocación y particiones

Tiene disponibles dos estrategias de colocación para WebSphere eXtreme Scale: partición fija y por contenedor. La elección de la estrategia de colocación afecta a cómo la configuración de despliegue coloca particiones en la cuadrícula de datos remota.

Colocación de partición fija

Puede establecer la estrategia de colocación en el archivo XML de la política de despliegue. La estrategia de colocación predeterminada es la ubicación de partición fija, habilitada con el valor `FIXED_PARTITION`. El número de fragmentos primarios que se colocan en los contenedores disponibles es igual al número de particiones que ha configurado con el atributo numberOfPartitions. Si ha configurado réplicas, el número mínimo total de fragmentos colocados se define a través de la siguiente fórmula: `((1 fragmento primario + fragmentos mínimos síncronos) * particiones definidas)`. El número máximo total de fragmentos colocados se define a través de la siguiente fórmula: `((1 fragmento primario + máximo de fragmentos síncronos + máximo de fragmentos asíncronos) * particiones)`. El despliegue de WebSphere eXtreme Scale se distribuye entre estos fragmentos sobre los contenedores disponibles. Las claves de cada correlación pasan por el código hash en particiones asignadas basándose en las particiones totales que ha definido. Las claves realizan el código hash en la misma partición, aunque la partición se mueva debido a una migración tras error o a cambios de servidor.

Por ejemplo, si el valor de numberPartitions es 6 y el valor de minSync es 1 para MapSet1, el número total de fragmentos para dicho conjunto de correlaciones es 12, porque cada una de las 6 particiones requiere una réplica síncrona. Si se inician

tres contenedores, WebSphere eXtreme Scale coloca cuatro fragmentos por contenedor paraMapSet1.

Colocación por contenedor

La estrategia de colocación alternativa es colocación por contenedor, que se habilita con el valor `PER_CONTAINER` para el atributo placementStrategy en el elemento de conjunto de correlaciones en el archivo XML de despliegue. Con esta estrategia, el número de fragmentos primarios colocados en cada contenedor nuevo es igual al número de particiones, *P*, que ha configurado. El entorno de despliegue de WebSphere eXtreme Scale coloca *P* réplicas de cada partición para cada contenedor restante. El valor numInitialContainers se ignora cuando se utiliza la colocación por contenedor. Las particiones cada vez son más grandes a medida que crecen los contenedores. Las claves para las correlaciones no son fijas para una determinada partición de esta estrategia. El cliente se direcciona a una partición y utiliza el primario aleatorio. Si un cliente desea volver a conectarse a la misma sesión que se ha utilizado para volver a encontrar la clave, debe utilizar un descriptor de contexto de sesión.

Para obtener más información, consulte el tema sobre la utilización de un SessionHandle para el direccionamiento en la *Guía de programación*.

Para los servidores detenidos o con migración tras error, el entorno de WebSphere eXtreme Scale traslada los fragmentos primarios en la estrategia de colocación por contenedor, si todavía contienen datos. Si los fragmentos están vacíos, se descartan. En la estrategia por contenedor, los fragmentos primarios antiguos no se conservan porque se colocan nuevos fragmentos primarios para cada contenedor.

WebSphere eXtreme Scale permite colocación por contenedor como alternativa a lo que se podría denominar la estrategia de colocación "típica", un enfoque de partición fija con la clave de una correlación en hash en una de estas particiones. En un caso por contenedor (que se establece con PER_CONTAINER), el despliegue coloca las particiones en el conjunto de los servidores de contenedor en línea y automáticamente las amplía o reduce a medida que se añaden o eliminan contenedores de la cuadrícula de datos del servidor. Una cuadrícula de datos con el enfoque de partición fija funciona bien para cuadrículas basadas en clave, donde la aplicación utiliza un objeto de clave para localizar los datos en la cuadrícula. A continuación se explica la alternativa.

Ejemplo de una cuadrícula de datos por contenedor

Las cuadrículas de datos PER_CONTAINER son distintas. Puede especificar que la cuadrícula de datos utiliza la estrategia de colocación PER_CONTAINER con el atributo placementStrategy en el archivo XML de despliegue. En lugar de configurar cuántas particiones en total desea en la cuadrícula de datos, puede especificar cuántas particiones desea por contenedor que inicie.

Por ejemplo, si establece cinco particiones por contenedor, se crearán cinco nuevos primarios de partición anónimos al iniciar el servidor de contenedor y se crearán las réplicas necesarias en los otros servidores de contenedor desplegados.

A continuación se muestra una secuencia potencial en un entorno por contenedor a medida que la cuadrícula de datos crece.

1. Iniciar el contenedor C0 que aloja 5 primarios (P0 - P4).
   - C0 aloja: P0, P1, P2, P3, P4.

2. Iniciar el contenedor C1 que aloja 5 primarios más (P5 - P9). Las réplicas se equilibran en los contenedores.
   - C0 aloja: P0, P1, P2, P3, P4, R5, R6, R7, R8, R9.
   - C1 aloja: P5, P6, P7, P8, P9, R0, R1, R2, R3, R4.
3. Iniciar el contenedor C2 que aloja 5 primarios más (P10 - P14). Las réplicas se siguen equilibrando.
   - C0 aloja: P0, P1, P2, P3, P4, R7, R8, R9, R10, R11, R12.
   - C1 aloja: P5, P6, P7, P8, P9, R2, R3, R4, R13, R14.
   - C2 aloja: P10, P11, P12, P13, P14, R5, R6, R0, R1.

El patrón continúa a medida que se inician más contenedores, creando cinco nuevas particiones primarias cada vez y reequilibrando las réplicas en los contenedores disponibles en la cuadrícula de datos.

Nota: WebSphere eXtreme Scale no traslada fragmentos primarios al utilizar la estrategia PER_CONTAINER, solo réplicas.

Recuerde que los números de partición son arbitrarios y no tienen nada que ver con las claves, por lo que no se puede utilizar el direccionamiento basado en claves. Si un contenedor se detiene, los ID de partición creados para dicho contenedor dejan de utilizarse, de modo que queda un vacío entre los ID de partición. En el ejemplo, ya no habría particiones P5 - P9 si el contenedor C2 fallase, quedando sólo P0 - P4 y P10 - P14, por lo que el hash basado en claves resulta imposible.

La utilización de números como cinco o incluso más probablemente 10 para el número de particiones por contenedor funciona mejor si tiene en cuenta las consecuencias de una anomalía de contenedor. Para distribuir la carga de alojar fragmentos equitativamente en la cuadrícula de datos, necesita más de una partición para cada contenedor. Si tiene una sola partición por contenedor, cuando un contenedor falle, un solo contenedor (el que aloja el fragmento de réplica correspondiente) deberá soportar la carga total del primario perdido. En este caso, la carga se dobla inmediatamente para el contenedor. Sin embargo, si tiene cinco particiones por contenedor, cinco contenedores asumen la carga del contenedor perdido, lo que disminuye el impacto en cada uno en un 80 por ciento. El uso de varias particiones por contenedor suele disminuir el impacto potencial sobre cada contenedor considerablemente. De forma más directa, considere un caso en el que un contenedor aumenta su carga de trabajo de improviso; la carga de réplica de dicho contenedor se distribuye en 5 contenedores en lugar de hacerlo en sólo uno.

Uso de la política por contenedor

Varios escenarios hacen que la estrategia por contenedor sea una configuración ideal, como por ejemplo la réplica de sesiones HTTP o el estado de sesión de aplicaciones. En tal caso, un direccionador HTTP asigna una sesión a un contenedor de servlet. El contenedor de servlet tiene que crear una sesión HTTP y elige uno de los 5 primarios de partición locales para la sesión. El "ID" de la partición elegida se almacena entonces en una cookie. El contenedor de servlet tiene ahora acceso local al estado de la sesión, lo que significa un acceso de latencia cero a los datos correspondientes a esta solicitud siempre que se mantenga la afinidad de sesiones. Y eXtreme Scale replica los cambios de la partición.

En la práctica, recuerde las repercusiones de un caso en el que tenga varias particiones por contenedor (por ejemplo, 5 otra vez). Naturalmente, con cada

contenedor nuevo que se inicie, tiene 5 primarios de partición más y 5 réplicas más. Con el tiempo, se deben crear más particiones y no se deben mover ni destruir. Pero los contenedores no se comportarían realmente de este modo. Cuando un contenedor se inicia, aloja 5 fragmentos primarios, que se pueden denominar ser primarios de "inicio", existentes en los contenedores respectivos que los crearon. Si el contenedor falla, las réplicas se convierten en primarios y eXtreme Scale crea 5 réplicas más para mantener la alta disponibilidad (a menos que se haya inhabilitado la reparación automática). Los nuevos nuevas primarios están en un contenedor diferente del que los creó y pueden denominarse primarios "externos". La aplicación nunca debe colocar un estado o sesiones nuevos en un primario externo. Al final, el primario externo no tiene ninguna entrada y eXtreme Scale lo suprime automáticamente y también suprime sus réplicas asociadas. El objetivo de los primarios externos es permitir que las sesiones existentes sigan estando disponibles (pero no las sesiones nuevas).

Un cliente puede seguir interactuando con una cuadrícula de datos que no se basa en claves. El cliente simplemente inicia una transacción y almacena los datos en la cuadrícula de datos independiente de cualquier clave. Solicita a la sesión un objeto SessionHandle, un descriptor de contexto serializable que permite al cliente interactuar con la misma partición cuando sea necesario. Si desea más información, consulte el tema sobre cómo utilizar un SessionHandle para el direccionamiento en la *Guía de programación*.WebSphere eXtreme Scale elige una partición para el cliente de la lista de primarios de partición iniciales. No devuelve una partición primaria externa. El SessionHandle se puede serializar en una cookie HTTP, por ejemplo, y más tarde se puede volver a convertir la cookie en un SessionHandle. A continuación, las API WebSphere eXtreme Scale pueden obtener una sesión vinculada de nuevo a la misma partición, utilizando SessionHandle.

Nota: No puede utilizar agentes para interactuar con una cuadrícula de datos PER_CONTAINER.

Ventajas

La descripción anterior es distinta a una cuadrícula de datos hash o FIXED_PARTITION normal porque el cliente por contenedor almacena los datos en un lugar de la cuadrícula, obtiene un manejador para ellos y lo utiliza para volver a acceder a ellos. No hay ninguna clave proporcionada por la aplicación, al contrario que en el caso de partición fija.

El despliegue no crea una partición nueva para cada sesión. Por lo tanto, en un despliegue por contenedor, las claves utilizadas para almacenar datos en la partición deben ser exclusivas en dicha partición. Por ejemplo, puede hacer que su cliente genere un SessionID exclusivo y luego lo utilice como clave para encontrar información en correlaciones en dicha partición. Luego, varias sesiones de cliente interaccionan con la misma partición, por lo que la aplicación tiene que utilizar claves exclusivas para almacenar datos de sesión en cada partición concreta.

Los ejemplos anteriores utilizaban 5 particiones, pero se puede utilizar el parámetro numberOfPartitions del archivo XML para especificar las particiones del modo necesario. En lugar de por cuadrícula de datos, el valor es por contenedor. (El número de réplicas se especifica del mismo modo que en la política de partición fija).

La política por contenedor también se puede utilizar con varias zonas. Si es posible, eXtreme Scale devuelve un SessionHandle a una partición cuyo primario está ubicado en la misma zona que el cliente en cuestión. El cliente puede

especificar la zona como un parámetro al contenedor o utilizando una API. El ID de zona del cliente se puede definir utilizando `serverproperties` o `clientproperties`.

La estrategia PER_CONTAINER para una cuadrícula de datos es adecuada para las aplicaciones que almacenan estado de tipo conversacional en lugar de datos orientados a base de datos. La clave para acceder a los datos sería un ID de conversación y no está relacionada con un registro de base de datos específico. Proporciona un rendimiento superior (porque los primarios de partición pueden compartir ubicación con los servlets por ejemplo) y facilita la configuración (no es necesario calcular particiones y contenedores).

Transacciones de partición única y transacciones entre cuadrículas de datos

La diferencia principal entre WebSphere eXtreme Scale y las soluciones de almacenamiento de datos tradicionales como las bases de datos relacionales o las bases de datos en memoria es el uso del particionamiento, que permite a la memoria caché realizar las escaladas de forma lineal. Los tipos importantes de transacciones a tener en cuenta son transacciones de partición única y transacciones de cada partición (entre cuadrículas de datos).

En general, las interacciones con la memoria caché se pueden categorizar como transacciones de una partición único o transacciones entre cuadrículas de datos, tal como se describe en la sección siguiente.

Transacciones de partición única

Las transacciones de partición única son el método preferible para interactuar con las memorias caché alojadas por WebSphere eXtreme Scale. Cuando una transacción está limitada a una única partición, de forma predeterminada, está limitada a una única Máquina virtual Java y, por lo tanto, un único sistema de servidor. Un servidor puede completar *M* número de estas transacciones por segundo y si tiene *N* sistemas, puede completar `M*N` transacciones por segundo. Si el negocio aumenta y debe doblar el rendimiento respecto a muchas de estas transacciones por segundo, puede doblar el valor *N* comprando más sistemas. Puede cumplir las demandas de capacidad sin modificar la aplicación, actualizar el hardware o, incluso, colocando la aplicación fuera de línea.

Además de permitir a la memoria caché realizar escaladas de forma significativa, las transacciones de partición única también maximizan la disponibilidad de la memoria caché. Cada transacción sólo depende de un sistema. Cualquiera de los otros `(N-1)` sistemas puede falla sin que esto afecte al éxito o al tiempo de respuesta de la transacción. Por lo tanto, si ejecuta 100 sistemas y uno de ellas falla, sólo el 1 por ciento de las transacciones en curso en el momento en que falla el servidor se retrotrae. Después de que el servidor falle, WebSphere eXtreme Scale reubica las particiones alojadas por el servidor anómalo en los otros 99 sistemas. Durante este breve periodo, antes de que se complete la operación, los otros 99 sistemas pueden seguir completando transacciones. Sólo las transacciones que podrían implicar que las particiones que se están reubicando se bloqueen. Después de que se complete el proceso de migración tras error, la memoria caché puede seguir ejecutándose, plenamente operativa a un 99 por ciento de su capacidad de rendimiento original. Después de que se sustituya un servidor anómalo y se devuelva a la cuadrícula de datos, la memoria caché vuelva al 100 por cien de la capacidad de rendimiento.

Transacciones entre cuadrículas de datos

En términos de rendimiento, disponibilidad y escalabilidad, las transacciones entre cuadrículas de datos son lo contrario a la transacciones de una partición única. Las transacciones entre cuadrículas de datos acceden a cada partición y por lo tanto cada sistema de la configuración. Se solicita a cada sistema de la cuadrícula de datos que busque algunos datos y que a continuación devuelva el resultado. La transacción no se puede completar hasta que han respondido todos los sistemas y, por lo tanto, el rendimiento de toda la cuadrícula de datos está limitado por el sistema más lento. Añadir sistemas no hace que el sistema más lento sea más rápido y, por lo tanto, no mejora el rendimiento de la memoria caché.

Las transacciones entre cuadrículas de datos tiene un efecto similar en la disponibilidad. Ampliando el ejemplo anterior, si ejecuta 100 servidores y uno falla, el 100 por ciento de las transacciones que están en curso en el momento en el que falló el servidor se retrotraen. Después de que falle el servidor, WebSphere eXtreme Scale empieza a reubicar las particiones alojadas por dicho servidor a los otros 99 sistemas. Durante este tiempo, antes de que se complete el proceso de migración tras error, la cuadrícula de datos no puede procesar ninguna de estas transacciones. Después de que se complete el proceso de migración tras error, la memoria caché puede seguir ejecutándose, pero a una capacidad reducida. Si cada sistema de la cuadrícula de datos presta servicio a 10 particiones, 10 de los 99 sistemas restantes recibirán como mínimo una partición adicional como parte del proceso de migración tas error. Añadir una partición adicional aumentar la carga de trabajo de dicho sistema en un 10 por ciento, como mínimo. Debido a que el rendimiento de la cuadrícula de datos está limitado al rendimiento del sistema más lento en una transacción entre cuadrículas de datos, de promedio el rendimiento se reduce en un 10 por ciento.

Las transacciones de partición única son preferibles a las transacciones entre cuadrículas de datos para el escalado con una memoria caché de objetos distribuida con alta disponibilidad, como WebSphere eXtreme Scale. La maximización del rendimiento de estas clases de sistemas requiere el uso de técnicas distintas a las metodologías relacionales tradicionales, pero puede convertir las transacciones entre cuadrículas de datos en transacciones escalables de una partición única.

Procedimientos recomendados para crear modelos de datos escalables

Los procedimientos recomendados para crear aplicaciones escalables con productos como WebSphere eXtreme Scale incluyen dos categorías: los principios fundacionales y las sugerencias de implementación. Los principios fundacionales son ideas principales que se deben capturar en el diseño de los propios datos. Una aplicación que no observa estos principios probablemente no realizará bien las escaladas, incluso para sus transacciones principales. Se aplican las sugerencias de implementación para las transacciones problemáticas en una aplicación bien diseñada de otra forma que observa los principios generales para los modelos de datos escalables.

Principios fundacionales

Algunos de los métodos importantes para optimizar la escalabilidad son conceptos o principios básicos que se deben tener en cuenta.

Duplicar en lugar de normalizar

El concepto clave para recordar sobre los productos como WebSphere eXtreme Scale es que se han diseñado para distribuir los datos entre un gran número de sistemas. Si el objetivo es completar la mayoría o todas las transacciones en una única partición, el diseño del modelo de datos debe garantizar que todos los datos que podría necesitar la transacción se encuentran en la partición. La mayoría del tiempo, la única forma de conseguir esto es duplicando los datos.

Por ejemplo, considere una aplicación como un tablón de mensajes. Dos transacciones muy importantes para un tablón de mensajes son mostrar todas las publicaciones de un usuario proporcionado y todas las publicaciones sobre un tema determinado. En primer lugar, considere cómo estas transacciones funcionarían con un modelo de datos normalizado que contiene un registro de usuarios, un registro de temas y un registro de publicaciones que contiene el texto real. Si las publicaciones se particionan con registros de usuarios, la visualización del tema pasa a ser una transacción entre cuadrícula y viceversa. Los temas y los registros no se pueden particionar juntos porque tienen una relación de muchos a muchos.

El mejor método para realizar esta escalada del tablón de mensajes es duplicar las publicaciones, almacenando una copia con el registro de temas y una copia con el registro de usuarios. A continuación, la visualización de las publicaciones de un usuario es una transacción de partición única, la visualización de las publicaciones sobre un tema es una transacción de partición única y la actualización o la supresión de una publicación es una transacción de dos particiones. Estas tres transacciones se escalarán de forma lineal, ya que el número de sistemas de la cuadrícula de datos aumenta.

Escalabilidad en lugar de recursos

El mayor obstáculo para superar cuando se considera eliminar la normalización de los modelos de datos es el impacto que estos modelos tendrían en los recursos. Podría parecer que conservar dos, tres o más copias de algunos datos utiliza demasiados recursos para que sea práctico. Cuando lo confronta con este escenario, recuerde los siguientes hechos: los recursos de hardware son más baratos cada año. En segundo lugar, y más importante, WebSphere eXtreme Scale elimina los costes más ocultos asociados al despliegue de más recursos.

Medir los recursos en términos de coste, en lugar de en términos de sistema como, por ejemplo, megabytes y procesadores. Generalmente, los almacenes de datos que funcionan con datos relacionales normalizados deben estar situados en el mismo sistema. Esta ubicación necesaria significa que se debe adquirir un único gran sistema empresarial, en lugar de varios sistemas pequeños. Con el hardware de empresa, no es raro que un sistema capaz de completar un millón de transacciones por segundo cueste muchos más que el coste combinado de 10 sistemas capaces de realizar 100.000 transacciones por segundos cada uno.

También existe un coste empresarial en la adición de recursos. Una negocio creciente acaba por agotar la capacidad. Cuando se agota la capacidad, debe concluir mientras se traslada a un sistema mayor y más rápido, o bien crear un segundo entorno de producción al que se puede pasar. De cualquier modo, los costes adicionales vendrán en forma de pérdidas de negocio o en el mantenimiento de casi el doble de la capacidad necesaria durante el periodo de transacción.

Con WebSphere eXtreme Scale, no es necesario concluir la aplicación para añadir capacidad. Si la empresa proyecta que se requiere un 10 por ciento más de capacidad para el próxima año, aumente el número de sistemas de la cuadrícula de datos en un 10 por ciento. Puede aumentar este porcentaje sin tiempo de inactividad de la aplicación y sin adquirir excesiva capacidad.

Evitar transformaciones de datos

Cuando se utiliza WebSphere eXtreme Scale, los datos se deben almacenar en un formato que pueda consumir directamente la lógica empresarial. Desglosar los datos en un formato más primitivo es costoso. La transformación se debe realizar cuando los datos se escriben y cuando los datos se leen. Con las bases de datos relacionales, esta transformación se realiza por necesidad, porque los datos se persisten de forma última en el disco con bastante frecuencia, pero con WebSphere eXtreme Scale, no es necesario que realice estas transformaciones. Para la mayoría de las partes, los datos se almacenan en la memoria y, por lo tanto, se almacenan en el formato exacto que necesita la aplicación.

Observar esta regla simple le ayuda a eliminar la normalización de los datos de acuerdo con el primer principio. El tipo más común de transformación para los datos empresariales es las operaciones JOIN que son necesarias para convertir los datos normalizados en un conjunto de resultados que se ajuste a las necesidades de la aplicación. Almacenar los datos en el formato correcto impide de forma implícita realizar estas operaciones JOIN y genera un modelo de datos no normalizados.

Eliminar consultas no enlazadas

Independientemente de cómo se estructuren los datos, las consultas no enlazadas no se escalan bien. Por ejemplo, no se tiene una transacción que solicite una lista de todos los elementos ordenados por un valor. Esta transacción podría funcionar a la primera, si el número total de elementos es 1000, pero si el número total de elementos llega a 10 millones, la transacción devuelve todos los 10 millones de elementos. Si ejecuta esta transacción, los dos resultados más probables son que la transacción agote el tiempo o que el cliente encuentre un error de memoria agotada.

La mejor opción es alterar la lógica empresarial de forma que sólo se puedan devolver los 10 o 20 primeros elementos. La alteración de la lógica mantiene el tamaño de la transacción gestionable, independientemente de cuántos elementos contenga la memoria caché.

Definir esquema

La principal ventaja de normalizar los datos es que el sistema de la base de datos puede ocuparse de la coherencia de los datos de forma interna. Cuando se elimina la normalización de los datos para la escalabilidad, deja de existir esta gestión automática de la coherencia de los datos. Debe implementar un modelo de datos que puede funcionar en la capa de la aplicación o como plug-in en la cuadrícula de datos distribuida para garantizar la coherencia de los datos.

Considere el ejemplo del tablón de mensajes. Si una transacción elimina una publicación de un tema, se debe eliminar la publicación duplicada del registro de usuarios. Sin un modelo de datos, es posible que un desarrollador escriba el código de la aplicación para eliminar la publicación del tema y olvide eliminar la publicación del registro de usuarios. Sin embargo, si el desarrollador estuviera utilizando un modelo de datos, en

lugar de interactuando directamente con la memoria caché, el método removePost en el modelo de datos podría extraer el ID de usuario de la publicación, buscar el registro de usuarios y eliminar la publicación duplicada de forma interna.

De forma alternativa, puede implementar un receptor que se ejecuta en la partición real que detecta el cambio en el tema y ajusta automáticamente el registro de usuarios. Un receptor podría ser beneficioso porque el ajuste del registro de usuarios se podría realizar de forma local si la partición parece tener el registro de usuarios, o incluso si el registro de usuarios está en una partición distinta, la transacción se produce entre los servidores, en lugar de entre el cliente y el servidor. Probablemente, la conexión de red entre los servidores es más rápida que la conexión de red entre el cliente y el servidor.

Impedir la competencia

Se impiden escenarios como tener un contador global. La cuadrícula de datos no se escalará si un único registro se está utilizando un número desproporcionado de veces en comparación con los demás registros. El rendimiento de la cuadrícula de datos se limitará al rendimiento del sistema que aloja un registro determinado.

En estas situaciones, intente dividir el registro para que sea gestionado por partición. Por ejemplo, considere una transacción que devuelve el número total de entradas en la memoria caché distribuida. En lugar de tener cada acceso de la operación insert y remove en un único registro que aumenta, tener un receptor en cada partición rastrea las operaciones insert y remove. Con este rastreo del receptor, las operaciones insert y remove se pueden convertir en operaciones de partición única.

La lectura de un contador pasará a ser una operación entre cuadrículas de datos, pero para la mayoría, ya era ineficaz como operación entre cuadrículas de datos porque su rendimiento estaba ligado al rendimiento del sistema que alojaba el registro.

Sugerencias de implementación

También puede considerar las siguientes sugerencias para conseguir la mejor escalabilidad.

Utilizar los índices de búsqueda inversa

Considere un modelo de datos que ha eliminado la normalización correctamente donde los registros de clientes se particionan basándose en el número del ID de cliente. Este método de particionamiento es la opción lógica porque casi todas las operaciones empresariales realizadas con el registro de clientes utilizan el número del ID del cliente. Sin embargo, una transacción importante que no utiliza el número del ID del cliente es la transacción de inicio de sesión. Es más común tener nombres de usuario o direcciones de correo electrónico para el inicio de sesión, en lugar de números de ID de cliente.

El enfoque sencillo al escenario de inicio de sesión es utilizar una transacción entre cuadrículas de datos para buscar el registro de cliente. Tal como se ha explicado previamente, este enfoque no realiza escaladas.

La siguiente opción podría ser la partición en el nombre del usuario o el correo electrónico. Esta opción no es práctica porque todas las operaciones basadas en ID de cliente pasan a ser transacciones entre cuadrículas de

datos. Asimismo, los clientes del sitio podrían desear cambiar su nombre de usuario o dirección de correo electrónico. Los productos como WebSphere eXtreme Scale necesitan el valor que se utiliza para la partición de datos en constantes de permanencia.

La solución correcta es utilizar un índice de búsqueda inversa. Con WebSphere eXtreme Scale, se puede crear una memoria caché en la misma cuadrícula distribuida que la memoria caché que aloja todos los registros de usuarios. Esta memoria caché es altamente disponible, está particionada y es escalable. Se puede utilizar esta memoria caché para correlacionar un nombre de usuario o dirección de correo electrónico con un ID de cliente. Esta memoria caché convierte el inicio de sesión en una operación de dos particiones, en lugar de una operación entre cuadrícula. Este escenario no es tan bueno como una transacción de partición única, pero el rendimiento se sigue escalando de forma lineal a medida que el número de sistemas aumenta.

Calcular en el momento de la escritura

Los valores calculados comúnmente como promedios o totales pueden resultar caros para generarse, porque normalmente estas operaciones requieren leer un gran número de entradas. Puesto que leer es más comunes que escribir en la mayoría de las aplicaciones, es eficaz calcular estos valores en el momento de escribir y, a continuación, almacenar el resultado en la memoria caché. Esta práctica hace que las operaciones de lectura sean más rápidas y más escalables.

Campos opcionales

Considere un registro de usuarios que incluya una empresa, un lugar y un número de teléfono. Un usuario puede tener todos estos números definidos, o ninguno o alguna combinación de éstos. Si los datos se normalizaron, podría existir una tabla de usuarios y una tabla de números de teléfono. Los números de teléfono para un usuario determinado se podrían encontrar utilizando una operación JOIN entre las dos tablas.

Eliminar la normalización de este registro no requiera la duplicación de datos, porque la mayoría de los usuarios no comparten los números de teléfono. En lugar de esto, debe estar permitido vaciar las ranuras del registro de usuarios. En lugar de tener una tabla de números de teléfono, añada tres atributos a cada registro de usuarios, uno para cada tipo de número de teléfono. Esta adición de atributos elimina la operación JOIN y realiza una búsqueda de número de teléfono para un usuario y una operación de partición única.

Colocación de relaciones de muchos a muchos

Considere una aplicación que rastrea los productos y las tiendas en las que se venden los productos. Un único producto se vende en muchas tiendas y una sola tienda vende muchos productos. Suponga que esta aplicación rastrea 50 tiendas grandes. Cada producto se vende en un máximo de 50 tiendas, con cada tienda que vende miles de productos.

Conservar una lista de tiendas dentro de la entidad de producto (disposición A), en lugar de conservar una lista de productos dentro de cada entidad de tienda (disposición B). Consultando algunas de las transacciones, esta aplicación podría realizar ilustraciones que expliquen por qué la disposición A es más escalable.

En primer lugar, consulte las actualizaciones. Con la disposición A, eliminar un producto del inventario de una tienda bloquea la entidad del

producto. Si la cuadrícula de datos aloja 10000 productos, solo será necesario bloquear el 1/10000 de la cuadrícula para realizar la actualización. Con la disposición B, la cuadrícula de datos solo contiene 50 tiendas, por lo que se debe bloquear el 1/50 de la cuadrícula para comletar la actualización. Así pues, aunque ambas disposiciones se podrían considerar operaciones de partición única, la disposición A se escala de forma horizontal de forma más eficaz.

Ahora, si se consideran las lecturas con la disposición A, buscar las tiendas en las que se vende un producto es una transacción de partición única que se escala y es rápida porque la transacción sólo transmite una pequeña cantidad de datos. Con la disposición B, esta transacción pasa a ser una transacción entre cuadrículas de datos porque se debe acceder a cada entidad de tienda para ver si el producto se vende en esa tienda, lo que implica una gran ventaja de rendimiento respecto a la disposición A.

Escalar con datos normalizados

Un uso legítimo de las transacción entre cuadrícula de datos es escalar el proceso de datos. Si una cuadrícula d e datos tiene 5 sistemas y se envía una transacción entre cuadrículas de datos que clasificar unos 100.000 registros e n cada sistema, esa transacción ordenará unos 500.000 registros. Si el sistema más lento de la cuadrícula de datos pueden realizar 10 de estas transacciones por segundo, la cuadrícula de datos puede ordenar unos 5.000.000 registros por segundo. Si los datos de la cuadrícula se doblan, cada sistema realiza una clasificación entre 200.000 registros y cada transacción realiza una clasificación entre 1.000.000 de registros. Este aumento de datos disminuye el rendimiento del sistema más lento a 5 transacciones por segundo, reducendo de esta forma el rendimiento de la cuadrícula de datos a 5 transacciones por segundo. Sin embargo, la cuadrícula de datos ordena unos 5.000.000 registros por segundo.

En este escenario, doblar el número de sistemas permite a cada sistema volver a su carga previa de clasificación entre 100.000 registros, lo que permite al sistema más lento procesar 10 de estas transacciones por segundo. El rendimiento de la cuadrícula de datos continúa siendo el mismo en 10 solicitudes por segundo, pero ahora cada transacción procesa 1.000.000 registros, así que la cuadrícula ha doblado su capacidad de proceso de registros a 10.000.000 por segundo.

Las aplicaciones como por ejemplo un motor de búsqueda que es necesario escalar en términos de proceso de datos para alojar el tamaño creciente de Internet y el rendimiento para acomodar el crecimiento en el número de usuarios, debe crear varias cuadrículas de datos, con un turno circular de las solicitudes entre cuadrículas. Si debe escalar el rendimiento, añada sistemas y añada otra cuadrícula de datos a las solicitudes de servicio. Si es necesario escalar el proceso de datos, añada más sistemas y mantenga constante el número de cuadrículas de datos.

Escalado en unidades o contenedores

Aunque puede desplegar una cuadrícula de datos sobre miles de máquinas virtuales Java, es posible que desee considerar dividir la cuadrícula de datos en unidades o contenedores para que sea más fiable y fácil probar la configuración. Un contenedor es un grupo de servidores que ejecuta el mismo conjunto de aplicaciones.

Despliegue de una sola cuadrícula de datos grande

Las pruebas han verificado que eXtreme Scale puede escalar hasta más de 1.000 JVM. Estas pruebas promueven la creación de aplicaciones para desplegar cuadrículas de datos individuales en gran cantidad de máquinas. Aunque es posible hacerlo, no se recomienda, por diversas razones:

1. **Cuestiones presupuestarias:** el entorno no puede probar de forma realista una cuadrícula de datos de 1000 servidores. Sin embargo puede probar una cuadrícula de datos mucho menor considerando las cuestiones presupuestarias, de forma que no necesite comprar dos veces el hardware, especialmente para un número tan grande de servidores.
2. **Diferentes versiones de aplicaciones:** Necesitar un número elevado de máquinas para cada hebra de pruebas no es práctico. El riesgo consiste en que no se prueban los mismos factores que se probarían en un entorno de producción.
3. **Pérdida de datos:** La ejecución de una base de datos en un solo disco duro no resulta fiable. Cualquier problema con el disco duro provocará una pérdida de datos. La ejecución de una aplicación creciente en una única cuadrícula de datos es similar. Probablemente experimentará errores en el entorno y en las aplicaciones. Por lo tanto, la colocación de todos los datos en un solo sistema de gran tamaño a menudo provocará una pérdida de grandes cantidades de datos.

División de la cuadrícula de datos

La división de la cuadrícula de datos de la aplicación en contenedores (unidades) es una opción más fiable. Un contenedor es un grupo de servidores que se ejecutan en una pila de aplicaciones homogénea. Los contenedores pueden tener cualquier tamaño, pero lo ideal es que consten de unos 20 servidores físicos. En lugar de tener 500 servidores físicos en una única cuadrícula de datos, puede tener 25 contenedores de 20 servidores físicos. Una sola versión de una pila de aplicaciones se debe ejecutar en un determinado contenedor, pero distintos contenedores pueden tener sus propias versiones de una pila de aplicaciones.

Generalmente, una pila de aplicaciones tiene en cuenta niveles de los componentes siguientes.

- Sistema operativo
- Hardware
- JVM
- Versión de WebSphere eXtreme Scale
- Aplicación
- Otros componentes necesarios

Un contenedor es una unidad de despliegue de tamaño apropiado para las pruebas. En lugar de tener cientos de servidores para las pruebas, es más práctico tener 20 servidores. En este caso, se sigue probando la misma configuración que tendría en el entorno de producción. En el entorno de producción se utilizan cuadrículas con un tamaño máximo de 20 servidores, que constituyen un contenedor. Puede someter a pruebas de tensión un solo contenedor y determinar su capacidad, número de usuarios, cantidad de datos y rendimiento de las transacciones. Esto facilita la planificación y se ajusta al estándar de disponer de un escalamiento previsible por un coste previsible.

Configuración de un entorno basado en contenedor

En diferentes casos, el contenedor no tiene que tener necesariamente 20 servidores. El objetivo del tamaño del contenedor es ajustarse a las pruebas prácticas. El tamaño de un contenedor debe ser suficientemente pequeño, de modo que si un contenedor encuentra problemas en producción la fracción de transacciones afectadas resulte tolerable.

Idealmente, cualquier error afecta a un único contenedor. Un error sólo afectaría al 4% de las transacciones de aplicación en lugar de afectar al 100%. Además, las actualizaciones resultan más fáciles porque se pueden desplegar en un contenedor a la vez. Como resultado, si una actualización de un contenedor causa problemas, el usuario puede conmutar de contenedor de nuevo al nivel anterior. Entre las actualizaciones figuran los cambios a la aplicación, la pila de aplicaciones o las actualizaciones del sistema. En la mayor medida posible, las actualizaciones sólo deben cambiar un único elemento de la pila a la vez para conseguir que el diagnóstico de problemas resulte más preciso.

Para implementar un entorno con contenedores, necesita una capa de direccionamiento sobre los contenedores que sea compatible con versiones anteriores y posteriores si se aplican actualizaciones de software a los contenedores. Además, debe crear un directorio que incluya la información acerca de qué contenedor incluye qué datos. Puede utilizar otra cuadrícula de datos de eXtreme Scale con este fin con una base de datos tras ella, preferiblemente utilizando el escenario de grabación diferida). Esto ofrece una solución de dos niveles. El nivel 1 es el directorio y se utiliza para ubicar qué contenedor maneja una determinada transacción. El nivel 2 consta de los propios contenedores. Cuando el nivel 1 identifica un contenedor, la configuración direcciona cada transacción al servidor correcto del contenedor, que es por lo general el servidor que contiene la partición correspondiente a los datos utilizados por la transacción. Opcionalmente, también puede utilizar una memoria caché cercana en el nivel 1 para reducir el impacto asociado con la búsqueda del contenedor correcto.

La utilización de contenedores es un poco más compleja que disponer de una sola cuadrícula de datos, pero las mejoras en el funcionamiento, las pruebas y la fiabilidad hacen que sea una parte crucial de las pruebas de escalabilidad.

Visión general de disponibilidad

Alta disponibilidad

Con una alta disponibilidad, WebSphere eXtreme Scale proporciona datos fiables, redundancia y detección de anomalías.

WebSphere eXtreme Scale autoorganiza las cuadrículas de datos de Mäquinas virtuales Java en un árbol federado libremente. El servicio de catálogo en los grupos raíz y principal que aloja los contenedores se encuentra en las hojas del árbol. Si desea más información, consulte “Arquitectura de memoria caché: correlaciones, contenedores, clientes y catálogos” en la página 11.

El servicio de catálogo crea cada grupo principal automáticamente en grupos de unos 20 servidores. Los miembros del grupo principal proporcionan la supervisión de estado de los otros miembros del grupo. Además, cada grupo principal elige un miembro para que sea el líder para comunicar la información del grupo al servicio de catálogo. Limitar el tamaño del grupo principal permite una supervisión de buena salud y un entorno con una gran capacidad de ampliación.

Nota: en un entorno de WebSphere Application Server, en el que se puede modificar el tamaño del grupo principal, eXtreme Scale no admite más de 50 miembros por grupo principal.

Pulsaciones

1. Los sockets se mantienen abiertos entre Mäquinas virtuales Java, y si un socket se cierra inesperadamente, este cierre imprevisto se detecta como una anomalía de la Máquina virtual Java de igual. Esta detección capta los casos de anomalías como, por ejemplo, la Máquina virtual Java que termina muy rápidamente. Dicha detección también permite la recuperación de estos tipos de anomalías, normalmente, en menos de un segundo.
2. Otros tipos de anomalías incluyen: pánico del sistema operativo, anomalía del servidor físico o anomalía de red. Estas anomalías se descubren mediante pulsaciones.

Las pulsaciones se envían de forma periódica entre pares de procesos: cuando se pierde un número fijo de pulsaciones, se supone que hay una anomalía. Este enfoque detecta anomalías en N*M segundos. N es el número de pulsaciones que se han perdido y M es el intervalo de pulsación. No se permite especificar directamente M ni N. Se utiliza un mecanismo de graduador para permitir que se utilice un rango de combinaciones M y N probadas.

Anomalías

Puede producirse una anomalía en un proceso por diversas causas. La anomalía puede ser debida a que se ha alcanzado un límite de recursos, como el tamaño máximo de almacenamiento dinámico, o que alguna lógica de control de proceso terminara un proceso. El sistema operativo podría fallar, lo que implicaría que se perdieran todos los procesos que se estuvieran ejecutando en el sistema. El hardware puede fallar, aunque es menos frecuente, como por ejemplo la tarjeta de interfaz de red (NIC), lo que provocaría que el sistema operativo se desconectase de la red. Pueden producirse más puntos de anomalías, que dejaría el proceso como no disponible. En este contexto, todas estas anomalías pueden clasificarse en uno de estos dos tipos: anomalías de proceso y pérdida de conectividad.

Anomalías de proceso

WebSphere eXtreme Scale reacciona rápidamente a anomalías de proceso. Cuando se produce una anomalía en un proceso, el sistema operativo es el responsable de limpiar los recursos sobrantes que utilizada el proceso. Esta limpieza incluye la asignación de puertos y conectividad. Cuando un proceso falla, se envía una señal a las conexiones que el proceso utilizaba para cerrar cada conexión. Gracias a estas señales, otro proceso conectado con el proceso que ha fallado puede detectar inmediatamente una anomalía en el proceso.

Pérdida de conectividad

Se produce pérdida de conectividad cuando se desconecta el sistema operativo. Como resultado, el sistema operativo no puede enviar señales a otros procesos. Las razones de la pérdida de conectividad son diversas, pero se pueden dividir en dos categorías: anomalía de host y aislamiento.

Anomalía de host

Si la máquina se desconecta de la corriente, se apaga inmediatamente.

Aislamiento

Este escenario presenta la condición de anomalía más complicada para que el software pueda gestionarlo correctamente porque el proceso aparenta no estar disponible, aunque lo esté. Básicamente, el sistema cree que un servidor u otro proceso ha fallado, mientras que realmente se está ejecutando correctamente.

Anomalías de contenedor

Las anomalías de contenedor generalmente las descubren los contenedores de igual a través del mecanismo de grupo principal. Cuando se produce una anomalía en un contenedor o grupo de contenedores, el servicio de catálogo migra los fragmentos alojados en dichos contenedores. El servicio de catálogo busca primero una réplica síncrona antes de migrar a una réplica asíncrona. Después de que los fragmentos primarios se migren a contenedores de host nuevos, el servicio de catálogo busca los contenedores host nuevos de las réplicas que faltan.

Nota: Aislamiento de contenedor: este servicio de catálogo migra fragmentos fuera de los contenedores cuando se descubre que el contenedor no está disponible. Si dichos contenedores pasan a estar disponibles, el servicio de catálogo considera los contenedores adecuados para colocación como si fuera un flujo de arranque normal.

Latencia de detección de anomalía de contenedor

Las anomalías se pueden dividir en anomalías de poca importancia y anomalías graves. Las anomalías de poca importancia se suelen producir por un fallo en el proceso. Este tipo de anomalías lo detecta el sistema operativo, que puede recuperar rápidamente recursos utilizados, como por ejemplo sockets de red. La detección de este tipo de anomalía se realiza en menos de un segundo. Es posible que la detección de anomalías graves con el ajuste de pulsación predeterminado requiera hasta 200 segundos. Este tipo de anomalías incluye: bloqueos de la máquina física, desconexiones del cable de red o anomalías del sistema operativo. El tiempo de ejecución se basa en la pulsación para detectar anomalías graves que se puedan configurar.

Anomalía del servicio de catálogo

Puesto que la cuadrícula del servicio de catálogo es una cuadrícula de eXtreme Scale, también utiliza el mecanismo de agrupación principal del mismo modo que el proceso de anomalía del contenedor. La diferencia principal es que el dominio de servicio de catálogo utiliza un proceso de elección de igual para definir el fragmento primario, en lugar del algoritmo del servicio de catálogo que se utiliza para los contenedores.

El servicio de colocación y el servicio de agrupación principal son uno de N servicios. Uno de N servicios se ejecuta en un miembro del grupo de alta disponibilidad. El servicio de ubicación y la administración se ejecutan en todos los miembros del grupo de alta disponibilidad. El servicio de colocación y el servicio de agrupamiento principal son objetos singleton porque son responsables de la presentación del sistema. El servicio de ubicación y administración son servicios de solo lectura y existen en cualquier punto para proporcionar escalabilidad.

El servicio de catálogo utiliza la réplica para convertirse en tolerante a errores. Si el proceso de servicio de catálogo falla, el servicio se reinicia para restaurar el sistema en el nivel de disponibilidad que se desea. Si todos los procesos que alojan el

servicio de catálogo fallan, la cuadrícula de datos tiene una pérdida de datos crítica. Esta anomalía genera un reinicio necesario de todos los servidores de contenedor. Como el servicio de catálogo puede ejecutarse en numerosos procesos, esta anomalía es poco probable. No obstante, si ejecuta todos los procesos en una única máquina, dentro de un armazón blade sencillo, o en un conmutador de red, es más probable que se produzca una anomalía. Elimine las modalidades de anomalías comunes de las máquinas que alojan el servicio de catálogo para reducir la posibilidad de anomalía.

Anomalías de varios contenedores

Una réplica nunca se coloca en el mismo proceso que su fragmento primario porque si el proceso se pierde, se perderían tanto la réplica como el fragmento primario. En un entorno de despliegue en una única máquina, puede tener dos contenedores y realizar réplicas entre ellos. Puede definir el atributo de modalidad de desarrollo en la política de despliegue para configurar una réplica para colocarla en la misma máquina que un primario. Sin embargo, en producción, la utilización de una sola máquina no es suficiente porque la pérdida de datos de ese host produce la pérdida de ambos servidores de contenedor. Para cambiar de modalidad de desarrollo en una única máquina a una modalidad de producción con varias máquinas y viceversa, inhabilite la modalidad de desarrollo en el archivo de configuración de la política de despliegue.

Tabla 4. Resumen del descubrimiento de anomalías y la recuperación

Tipo de pérdida	Mecanismo de descubrimiento (detección)	Método de recuperación
Pérdida de proceso	E/S	Reiniciar
Pérdida del servidor	Pulsación	Reiniciar
Parada de la red	Pulsación	Restablecer la red y la conexión
Bloqueo del lado del servidor	Pulsación	Detener y reiniciar el servidor
Servidor ocupado	Pulsación	Esperar hasta que el servidor esté disponible

Réplica para la disponibilidad

La duplicación proporciona tolerancia a anomalías y aumenta el rendimiento para una topología de eXtreme Scale distribuida. La réplica se habilita asociando correlaciones de respaldo con un conjunto de correlaciones.

Acerca de los conjuntos de correlaciones

Un conjunto de correlaciones es una colección de correlaciones que se categorizan por clave de partición. Esta clave de partición se obtiene de la clave de la correlación individual tomando el módulo hash para el número de particiones. Si un grupo de correlaciones del conjunto de correlaciones tiene la clave de partición X, esas correlaciones se almacenan en una partición X correspondiente en la cuadrícula de datos. Si otro grupo tiene la clave de partición Y, todas las correlaciones se almacenan en una partición Y y así sucesivamente. Los datos de las correlaciones se replican en función de la política definida en el conjunto de correlaciones. La réplica se produce en topologías distribuidas.

A los conjuntos de correlaciones se les asigna el número de particiones y una política de réplica. La configuración de réplica de conjunto de correlaciones identifica el número de fragmentos de réplica síncronos y asíncronos que el conjunto de correlaciones debe tener además del fragmento primario. Por ejemplo, si existe una réplica síncrona y una réplica asíncrona, en todas las BackingMaps asignadas al conjunto de correlaciones se distribuye automáticamente un fragmento de réplica en el conjunto de servidores de contenedor disponibles para la cuadrícula de datos. La configuración de réplica también puede permitir que los clientes lean datos de servidores duplicados de forma síncrona. Esto puede esparcir la carga de las solicitudes de lectura entre servidores adicionales en eXtreme Scale. La réplica sólo tiene un impacto de modelo de programación cuando se realiza la precarga de las correlaciones de respaldo.

Precarga de correlaciones

Las correlaciones se pueden asociar a cargadores. Un cargador se utiliza para captar objetos cuando no se pueden encontrar en la correlación (una falta de coincidencia) así como para grabar los cambios en un programa de fondo cuando se confirma una transacción. Los cargadores también se pueden utilizar para cargar previamente datos en una correlación. Se llama al método preloadMap de la interfaz Loader en cada correlación cuando la partición correspondiente del conjunto de correlaciones se convierte en primario. El método preloadMap no se llama en las réplicas. Intenta cargar todos los datos referenciados previstos del programa de fondo en la correlación utilizando la sesión proporcionada. La correlación pertinente se identifica mediante el argumento BackingMap que se pasa al método preloadMap.

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

Precarga en conjunto de correlaciones particionado

Las correlaciones puede particionarse en N particiones. Por lo tanto, las correlaciones pueden extenderse por varios servidores, con cada entrada identificada por una clave que sólo se almacena en uno de esos servidores. Las correlaciones muy grandes pueden mantenerse en un eXtreme Scale porque la aplicación ya no está limitada por el tamaño del almacenamiento dinámico de una sola JVM para mantener todas las entradas de una correlación. Las aplicaciones que desea cargar previamente con el método preloadMap de la interfaz del cargador deben identificar el subconjunto de datos que carga previamente. Siempre existe un número fijo de particiones. Puede determinar este número utilizando el siguiente ejemplo de código:

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

Este ejemplo de código muestra que una aplicación puede identificar el subconjunto de los datos que se deben precargar de la base de datos. Las aplicaciones siempre deben utilizar estos métodos incluso cuando la correlación no está particionada inicialmente. Estos métodos permiten una cierta flexibilidad: si posteriormente los administradores particionan la correlación, el cargador sigue funcionando correctamente.

La aplicación debe emitir consultas para recuperar el subconjunto *myPartition* del programa de fondo. Si se utiliza una base de datos, puede ser más fácil tener una columna con un identificador de partición para un registro dado salvo que haya alguna consulta natural que permita a los datos de la tabla particionarse fácilmente.

Rendimiento

La implementación de la precarga copia datos del programa de fondo en la correlación almacenando varios objetos en la correlación de una única transacción. El número óptimo de registros para almacenar por transacción depende de varios factores, incluidos la complejidad y el tamaño. Por ejemplo, después de que la transacción incluya bloques de más de 100 entradas, se reduce la ventaja del rendimiento a medida que aumenta el número de entradas. Para determinar el número óptimo, empiece con 100 entradas y, a continuación, aumente el número hasta que no se detecte más aumento en el rendimiento. Las transacciones de mayor tamaño dan como resultado un mayor rendimiento de duplicación. Recuerde que sólo el fragmento primario ejecuta el código de precarga. Los datos cargados previamente se duplican desde el fragmento primario hasta todas las réplicas que están en línea.

Precarga de conjuntos de correlaciones

Si la aplicación utiliza un conjunto de correlaciones con varias correlaciones, cada correlación tiene su propio cargador. Cada cargador tiene un método de carga previa. eXtreme Scale carga cada correlación en serie. Será más eficaz precargar todas las correlaciones designando una única correlación como la correlación de precarga. Este proceso es un convenio de aplicación. Por ejemplo, dos correlaciones, departamento y empleado, podrían utilizar el cargador de departamento para cargar previamente las correlaciones de departamento y de empleado. Este procedimiento asegura que, transaccionalmente, si una aplicación desea un departamento los empleados de dicho departamento están en la memoria caché. Cuando el cargador de departamento precarga un departamento desde el programa de fondo, también capta los empleados de dicho departamento. El objeto de departamento y sus objetos de empleados asociados se añadirán a la correlación utilizando una sola transacción.

Precarga recuperable

Algunos clientes tienen conjuntos de datos de gran tamaño que necesitan almacenarse en la memoria caché. La precarga de estos datos puede requerir mucho tiempo. A veces, la precarga debe finalizar para que la aplicación pueda ir en línea. Puede sacar provecho de que la precarga sea recuperable. Suponga que hay un millón de registros que se deben precargar. El fragmento primario los precarga y falla al llegar al registro número 800.000. Normalmente, la réplica elegida como el nuevo fragmento primario borra los estados duplicados y empieza desde el principio. eXtreme Scale puede emplear una interfaz ReplicaPreloadController. El cargador de la aplicación también necesitará implementar la interfaz ReplicaPreloadController. Este ejemplo añade un solo método al cargador: `Status checkPreloadStatus(Session session, BackingMap bmap);`. Este método lo invoca el tiempo de ejecución de eXtreme Scale antes de que se llame al método de carga previa de la interfaz del cargador. eXtreme Scale comprueba el resultado de este método (estado) para determinar su comportamiento siempre que una réplica pasa a ser un fragmento primario.

Tabla 5. Valor de estado y respuesta

Valor de estado devuelto	Respuesta de eXtreme Scale
Status.PRELOADED_ALREADY	eXtreme Scale no llama al método de precarga porque su valor de estado indica que la correlación se ha precargado completamente.
Status.FULL_PRELOAD_NEEDED	eXtreme Scale borra la correlación y llama de forma normal al método de precarga.

Tabla 5. Valor de estado y respuesta (continuación)

Valor de estado devuelto	Respuesta de eXtreme Scale
Status.PARTIAL_PRELOAD_NEEDED	eXtreme Scale deja la correlación tal cual y llama a la precarga. Esta estrategia permite al cargador de aplicación seguir realizando la precarga a partir de ese momento.

Evidentemente, mientras un fragmento primario está precargando la correlación, debe dejar algún estado en una correlación del conjunto de correlaciones que se está replicando para que la réplica determine qué estado debe devolver. Puede utilizar una correlación adicional llamada, por ejemplo, RecoveryMap. Esta RecoveryMap debe formar parte del mismo conjunto de correlaciones que se está precargando para asegurar que la correlación se replica coherentemente con los datos que se están precargando. A continuación se muestra una implementación sugerida.

Cuando la precarga confirma cada bloque de registros, el proceso también actualiza un contador o valor en la RecoveryMap como parte de esa transacción. Los datos precargados y los datos de RecoveryMap se duplican de forma atómica en las réplicas. Cuando la réplica se promociona a fragmento primario, puede comprobar la RecoveryMap para ver qué ha pasado.

RecoveryMap puede mantener una sola entrada con la clave de estado. Si no existe ningún objeto para esta clave, necesita una precarga completa (checkPreloadStatus devuelve FULL_PRELOAD_NEEDED). Si existe un objeto para esta clave de estado y el valor es COMPLETE, la precarga se completa y el método checkPreloadStatus devuelve PRELOADED_ALREADY. De lo contrario, el objeto de valor indica dónde se inicia la precarga y el método checkPreloadStatus devuelve: `PARTIAL_PRELOAD_NEEDED`. El cargador puede almacenar el punto de recuperación en una variable de instancia para el cargador de forma que, cuando se invoque la precarga, el cargador sepa el punto de partida. RecoveryMap también puede mantener una entrada por correlación si cada correlación se precarga independientemente.

Manejo de recuperación en modalidad de réplica síncrona con un cargador

El tiempo de ejecución de eXtreme Scale se ha diseñado para que no pierda datos confirmados cuando el fragmento primario falla. En la siguiente sección se muestran los algoritmos utilizados. Estos algoritmos sólo se aplican cuando un grupo de réplicas utiliza la réplica síncrona. Un cargador es opcional.

El tiempo de ejecución de eXtreme Scale puede configurarse de modo que duplique de forma síncrona todos los cambios de un fragmento primario en las réplicas. Cuando se coloca una réplica síncrona, recibe una copia de los datos existentes en el fragmento primario. Durante este tiempo, el primario continúa recibiendo transacciones y las copia en la réplica de forma asíncrona. La réplica no se considera en línea en este momento.

Después de que la réplica capte el primario, la réplica entra en la modalidad de igual y se inicia la réplica síncrona. Cada transacción confirmada en el primario se envía a las réplicas síncronas y el primario espera una respuesta de cada réplica. Una secuencia de confirmación síncrona con un cargador en el primario se parece al siguiente conjunto de pasos:

Tabla 6. Secuencia de confirmación del fragmento primario

Paso con cargador	Paso sin cargador
Obtener bloqueos para entradas	igual
Desechar cambios para el cargador	no operativo
Guardar cambios en la memoria caché	igual
Enviar cambios a réplicas y esperar el reconocimiento	igual
Confirmar en el cargador a través del plug-in TransactionCallback	Se llama a la confirmación de plug-in, pero no sucede nada
Liberar bloqueos para entradas	igual

Tenga en cuenta que los cambios se envían a la réplica antes de que se confirmen en el cargador. Para determinar cuando se confirman los cambios en la réplica, revise esta sentencia: en el momento de la inicialización, inicializar las listas tx en el fragmento primario tal como se indica a continuación.

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

Durante el proceso de confirmaciónsíncrono, utilice la siguiente secuencia:

Tabla 7. Proceso de confirmación síncrona

Paso con cargador	Paso sin cargador
Obtener bloqueos para entradas	igual
Desechar cambios para el cargador	no operativo
Guardar cambios en la memoria caché	igual
Enviar cambios con una transacción confirmada, retrotraer la transacción a la réplica y esperar el reconocimiento	igual
Borrar lista de transacciones confirmadas y transacciones retrotraídas	igual
Confirmar en el cargador a través del plug-in TransactionCallback	Se sigue llamando a la confirmación del plug-in TransactionCallBack, pero normalmente no sucede nada
Si la confirmación es satisfactoria, añada la transacción a las transacciones confirmadas, de lo contrario, añádala a las transacciones retrotraídas	no operativo
Liberar bloqueos para entradas	igual

Para el proceso de réplicas, utilice la siguiente secuencia:

1. Recibir cambios
2. Confirmar todas las transacciones recibidas en la lista de transacciones confirmadas
3. Retrotraer todas las transacciones recibidas en la lista de transacciones retrotraídas
4. Iniciar una transacción o sesión
5. Aplicar cambios en la transacción o sesión
6. Guardar la transacción o sesión en la lista de pendientes
7. Devolver respuesta

Tenga en cuenta que en la réplica, no se produce ninguna interacción de cargador mientras la réplica está en modalidad de réplica. El fragmento primario debe pasar todos los cambios a través del cargador. La réplica no pasa ningún cambio. Un efecto secundario de este algoritmo es que la réplica siempre tiene las transacciones, pero éstas no se confirman hasta que la siguiente transacción primaria envía el estado de confirmado de estas transacciones. A continuación, las transacciones se confirman o retrotraen en la réplica. Hasta entonces, las transacciones no están confirmadas. Puede añadir un temporizador en el primario que envía el resultado de la transacción después de un breve periodo (unos pocos minutos). Este temporizador limita, pero no elimina, cualquier obsolescencia a este periodo de tiempo. Esta obsolescencia sólo es un problema si se utiliza la modalidad de lectura de réplica. Si no, la obsolescencia no tiene ningún impacto en la aplicación.

Cuando el fragmento primario falla, es probable que haya unas pocas transacciones confirmadas o retrotraídas en el fragmento primario, pero el mensaje nunca llega a la réplica con estos resultados. Cuando una réplica se promociona y pasa a ser el nuevo fragmento primario, una de las primeras acciones es manejar esta condición. Cada transacción pendiente se vuelve a procesar respecto al conjunto de correlaciones del nuevo fragmento primario. Si hay un cargador, cada transacción se ofrece al cargador. Estas transacciones se aplican estrictamente en el orden primero en entrar, primero en salir (FIFO). Si una transacción falla, se ignora. Si hay tres transacciones pendientes, A, B y C, es posible que A se confirme, B se retrotraiga y C también se confirme. Ninguna transacción tiene ningún impacto en las demás. Suponga que son independientes.

Un cargador puede que desee utilizar una lógica un poco distinta cuando está en modalidad de recuperación de migración tras error comparada con la modalidad normal. El cargador puede saber fácilmente cuando está en modalidad de recuperación de migración tras error implementando la interfaz ReplicaPreloadController. El método checkPreloadStatus sólo se invoca cuando se completa la recuperación de la migración tras error. Por lo tanto, si el método de aplicación de la interfaz del cargador se invoca antes del método checkPreloadStatus, se trata de una transacción de recuperación. Después de llamar al método checkPreloadStatus, la recuperación de migración tras error está completa.

Equilibrio de carga entre réplicas

eXtreme Scale, salvo que se configure de otra manera, envía todas las solicitudes de lectura y grabación al servidor primario para un grupo de réplicas determinado. El fragmento primario debe atender todas las solicitudes de los clientes. Es posible que desee permitir que las solicitudes de lectura se envíen a las réplicas del fragmento primario. Si envía solicitudes de lectura a las réplicas la carga de las solicitudes de envío se podrá compartir entre varias JVM (Java Virtual Machines). No obstante, el uso de réplicas para las solicitudes de lectura puede generar repuestas incoherentes.

El equilibrio de carga entre réplicas normalmente se utiliza sólo cuando los clientes almacenan en la memoria caché datos que almacenan siempre o cuando los clientes utilizan el bloqueo pesimista.

Si lo datos se almacenan en la memoria caché constantemente y luego se invalidan en la memoria caché cercana del cliente, como resultado el fragmento primario debe detectar un índice de solicitudes get relativamente alto de los clientes.

Asimismo, en modalidad de bloqueo pesimista, no existe memoria caché local, y por ello todas las solicitudes se envían al fragmento primario.

Si los datos son relativamente estáticos o si no se utiliza la modalidad pesimista, el envío de solicitudes de lectura a la réplica no tiene un gran impacto en el rendimiento. La frecuencia de las solicitudes get de los clientes con memoria caché que están llenas de datos no es alta.

Cuando un cliente se inicia, su memoria caché cercana está vacía. Las solicitudes de memoria caché para la memoria caché vacía se remiten al fragmento primario. La memoria caché del cliente obtendrá datos con el tiempo, lo que provocará que la carga de solicitudes baje. Si muchos clientes se inician simultáneamente, es posible que la carga sea significativa y la lectura de réplicas puede ser una opción de rendimiento apropiada.

Réplica del lado del cliente

Con eXtreme Scale, puede duplicar una correlación de servidor con uno o más clientes utilizando la réplica asíncrona. Un cliente puede solicitar una copia de sólo lectura local de una correlación en el servidor utilizando el método ClientReplicableMap.enableClientReplication.

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

El primer parámetro es la modalidad de réplica. Esta modalidad puede ser una réplica continua o una réplica de instantánea. El segundo parámetro es una matriz de ID de particiones que representan las particiones desde las que duplicar los datos. Si el valor es nulo o una matriz vacía, los datos se duplican desde todas las particiones. El último parámetro es un escucha para recibir los sucesos de réplica de cliente. Para obtener detalles, consulte ClientReplicableMap y ReplicationMapListener en la documentación de la API.

Después de habilitar la réplica, el servidor empieza a duplicar la correlación con el cliente. Con el tiempo, el cliente sólo estará a unas pocas transacciones por detrás del servidor en cualquier momento dado.

Servicio de catálogo de alta disponibilidad

Un dominio de servicio de catálogo es la cuadrícula de datos de los servidores de catálogo que está utilizado, que mantienen información de topología de todos los contenedores del entorno de eXtreme Scale. El servicio de catálogo controla el equilibrio de carga y el direccionamiento para todos los clientes. Para desplegar eXtreme Scale como un espacio de proceso de base de datos en memoria, debe agrupar en clúster el servicio de catálogo en un dominio de servicio de catálogo para alta disponibilidad.

Componentes del dominio de servicio de catálogo

Cuando se inician varios servidores, uno de ellos se elige como el servidor de catálogo maestro que acepta las pulsaciones IIOP (Internet Inter-ORB Protocol) y maneja los cambios de datos del sistema en respuesta a cualquier cambio de servicio de catálogo o contenedor.

Cuando los clientes se ponen en contacto con uno de los servidores de catálogo, la tabla de direccionamiento del dominio de servicio de catálogo se propaga en los clientes a través del contexto del servicio CORBA (Common Object Request Broker Architecture).

Configure, como mínimo, tres servidores de catálogo. Los servidores de catálogo se deben instalar en nodos distintos o en imágenes de instalación distintas a los servidores de contenedor, a fin de garantizar que se podrán actualizar sin problemas los servidores en cualquier momento en el futuro. Si la configuración tiene zonas, puede configurar un servidor de catálogo por zona.

Cuando un servidor o contenedor eXtreme Scale se pone en contacto con uno de los servidores de catálogo, la tabla de direccionamiento del dominio de servicio de catálogo también se propaga en el servidor y contenedor eXtreme Scale a través del contexto de servicio CORBA. Además, si el servidor de catálogo contactado no es actualmente el servidor de catálogo maestro, la solicitud se redirecciona automáticamente al servidor de catálogo maestro actual y la tabla de direccionamiento del servidor de catálogo se actualiza.

Nota: Un dominio de servicio de catálogo y una cuadrícula de datos de servidor de contenedor son muy distintos. El dominio de servicio de catálogo es para la alta disponibilidad de los datos del sistema. La cuadrícula de datos de servidor de catálogo es para la alta disponibilidad de datos, la escalabilidad y la gestión de la carga de trabajo. Por lo tanto, existen dos tablas de direccionamiento distintas: la tabla de direccionamiento para el dominio de servicio de catálogo y la tabla de direccionamiento para los fragmentos de cuadrícula de datos de servidor de contenedor.

Las responsabilidades del dominio de servicio de catálogo se dividen en diversos servicios:

Gestor de grupos principales

El servicio de catálogo utiliza el High Availability Manager (Gestor HA) para agrupar los procesos para la supervisión de la disponibilidad. Cada grupo de procesos es un grupo principal. Con eXtreme Scale, el gestor de grupos principales agrupa dinámicamente los procesos juntos. Estos procesos son pequeños para favorecer la escalabilidad. Cada grupo principal elige un líder que tiene la responsabilidad añadida de enviar el estado al gestor de grupos principales cuando se produce una anomalía en los miembros individuales. Se utiliza el mismo mecanismo de estado para descubrir cuando fallan todos los miembros de un grupo, que provoca que falle la comunicación con el líder.

El gestor de grupos principales es un servicio totalmente automático responsable de organizar los contenedores en pequeños grupos de servidores que se federen automáticamente de manera ligera para conformar un ObjectGrid. Cuando un contenedor se pone en contacto por primera vez con el servicio de catálogo, espera a ser asignado a un grupo nuevo o existente. Un despliegue de eXtreme Scale se compone de varios de estos grupos, y este agrupamiento es un habilitador de escalabilidad clave. Cada grupo es un grupo de Mäquinas virtuales Java que utiliza la pulsación para supervisar la disponibilidad de los otros grupos. Uno de los miembros de estos grupos es elegido líder y tiene la responsabilidad añadida de transmitir información de disponibilidad al servicio de catálogo para permitir reaccionar ante anomalías mediante la reasignación y reenvío de rutas.

Servicio de colocación

El servicio de catálogo gestiona la colocación de fragmentos en el conjunto de servidores de contenedor disponibles. El servicio de colocación es responsable de mantener el equilibrio entre los recursos físicos. El servicio de colocación es responsable de asignar fragmentos individuales al

contenedor host. El servicio de colocación se ejecuta como uno de los N servicios elegidos en la cuadrícula de datos de modo que siempre hay exactamente una instancia del servicio en ejecución. Si la instancia falla, se elige otro proceso, que tomará el control. El estado del servicio de catálogo se replica en todos los servidores que alojan el servicio de catálogo para favorecer la redundancia.

Administración
El servicio de catálogo es también el punto de entrada lógico para la administración del sistema. El servicio de catálogo aloja un bean gestionado (MBean) y proporciona URL JMX (Java Management Extensions) para cualquiera de los servidores que gestiona el servicio de catálogo.

Servicio de ubicación
El servicio de ubicación actúa como un punto de contacto para tanto para los clientes que están buscando contenedores que alojan la aplicación que buscan, como para los servidores de contenedor que están registrando las aplicaciones alojadas en el servicio de colocación. El servidor de ubicación se ejecuta en todos los miembros de la cuadrícula de datos para escalar esta función.

Despliegue del dominio de servicio de catálogo

El servicio de catálogo contiene lógica que está inactiva durante los estados fijos. Como resultado, el servicio de catálogo influye mínimamente en la escalabilidad. El servicio se crea para dar servicio a cientos de contenedores que pasan a estar disponibles al mismo tiempo. Para la disponibilidad, configure el servicio de catálogo en una cuadrícula de datos.

Planificación

Después de iniciar un dominio de servicio de catálogo, los miembros de la cuadrícula de datos se enlazan juntos. Planifique con atención la topología del dominio de servicio de catálogo, porque no podrá modificar la configuración del dominio de servicio de catálogo durante la ejecución. Distribuya la cuadrícula de datos de la forma más diversificada posible para evitar errores.

Inicio de un dominio de servicio de catálogo

Para obtener más información sobre cómo crear un dominio de servicio de catálogo, consulte los detalles sobre cómo iniciar un dominio de servicio de catálogo en la *Guía de administración*.

Conexión de contenedores eXtreme Scale incorporados en WebSphere Application Server con un dominio de servicio de catálogo autónomo

Puede configurar contenedores eXtreme Scale que están incorporados en un entorno WebSphere Application Server para conectarse a un dominio de servicio de catálogo autónomo.

- Puede crear dominios de servicio de catálogo en la consola administrativa de WebSphere Application Server. Consulte Creación de dominios de servicio de catálogo en WebSphere Application Serverlos detalles sobre cómo crear dominios de servicio de catálogo en la *Guía de administración* para obtener más información.

-  (En desuso) En releases anteriores, ha conectado los servicios de catálogo en un dominio de servicio de catálogo creando una propiedad personalizada. Esta propiedad se puede utilizar todavía, pero está en desuso. Para obtener más información sobre esta propiedad personalizada, consulte Inicio y detención de servidores en un entorno de WebSphere Application Server la información sobre cómo iniciar el proceso de servicio de catálogo en un WebSphere Application Server en la *Guía de administración*..

  Nota: Colisión de nombre de servidor: puesto que esta propiedad se utiliza para iniciar el servidor de catálogo eXtreme Scale así como para conectarse, los servidores de catálogo no deben tener el mismo nombre que ningún servidor WebSphere Application Server.

Consulte "Quórums del servidor de catálogos" para obtener más información.

Quórums del servidor de catálogos

Cuando el mecanismo de quórum está habilitado, todos los servidores de catálogo del quórum deben estar disponibles para que se produzcan las operaciones de colocación en la cuadrícula de datos.

- "Términos importantes "
- "Pulsaciones y detección de anomalías"
- "Comportamiento de quórum" en la página 105
  - "Comportamiento del contenedor durante la pérdida de quórum" en la página 108
- "Comportamiento del cliente durante la pérdida de quórum" en la página 109

Términos importantes

- **Pulsación**: señal que se envía entre servidores para transmitir que están en ejecución.
- **Quórum**: grupo de servidores de catálogo que se comunican y realizan operaciones de colocación en la cuadrícula de datos. Este grupo consta de todos los servidores de catálogo de la cuadrícula de datos, a menos que se sustituya manualmente el mecanismo de quórum con acciones administrativas.
- **Caída de la red**: pérdida temporal de conectividad entre uno o más servidores.
- **Apagón**: pérdida permanente de conectividad entre uno o más servidores.
- **Centro de datos**: grupo localizado geográficamente de servidores que normalmente se conectan con una red de área local (LAN).
- **Zona**: una zona es una opción de configuración que se utiliza para agrupar servidores que comparten alguna característica física. Entre los ejemplos de zonas de grupos de servidores, se incluyen: un centro de datos, una red de área, un edificio o una planta de un edificio.

Pulsaciones y detección de anomalías

Servidores de contenedor y grupos principales

El servicio de catálogo coloca servidores de contenedor en grupos principales de tamaño limitado. Un grupo principal intenta detectar la anomalía de sus miembros. Se elige un solo miembro de un grupo principal para que sea el líder del grupo principal. El líder del grupo principal indica periódicamente al servicio de catálogo que el grupo principal está activo y notifica cualquier cambio en la pertenencia al

servicio de catálogo. Un cambio de pertenencia puede ser una JVM que falla o una JVM que se acaba de añadir y que se ha unido al grupo principal.

Si un socket de JVM está cerrado, se considera esa JVM como que ya no está disponible. Cada miembro de grupo principal también emite pulsaciones sobre estos sockets a una velocidad determinada por la configuración. Si una JVM no responde a estas pulsaciones en un periodo de tiempo máximo configurado, se considera que la JVM ya no está disponible, lo que desencadena una detección de anomalía.

Si el servicio de catálogo marca una JVM de contenedor como fallida y posteriormente se notifica el servidor de contenedor como disponible, se indica a la JVM de contenedor que concluya los servidores de contenedor WebSphere eXtreme Scale. Una JVM en este estado no se visualiza en las consultas del mandato del programa de utilidad **xscmd**. Los mensajes en los registros de la JVM de contenedor indican que esta última ha producido un error. Debe reiniciar manualmente estas JVM.

Si el líder del grupo principal no puede ponerse en contacto con ningún miembro, continúa intentando contactar con el miembro.

La anomalía completa de todos los miembros de un grupo principal también es una posibilidad. Si ha fallado todo el grupo principal, es responsabilidad del servicio de catálogo detectar esta pérdida.

Pulsación del dominio de servicio de catálogo

El dominio de servicio de catálogo tiene el aspecto de un grupo principal privado con una pertenencia estática y un mecanismo de quórum. Detecta las anomalías del mismo modo que un grupo principal normal. Sin embargo, el comportamiento se modifica para incluir la lógica del quórum. El servicio de catálogo utiliza también una configuración de pulsación menos agresiva.

Detección de errores

WebSphere eXtreme Scale detecta cuándo los procesos terminan mediante sucesos de cierre de socket anómalos. Se notifica inmediatamente al servicio de catálogo cuando un proceso termina.

Para obtener más información sobre la configuración de pulsación, consulte Ajuste del valor de intervalo de pulsación para la detección de migración tras errorla información sobre cómo configurar la detección de migración tras error en la *Guía de administración*.

Comportamiento de quórum

Normalmente, los miembros del servicio de catálogo tienen conectividad completa. El dominio de servicio de catálogo es un conjunto estático de JVM. WebSphere eXtreme Scale espera que todos los miembros del servicio de catálogo estén en línea. Cuando todos los miembros están en línea, el servicio de catálogo tiene quórum. El servicio de catálogo responde a sucesos de contenedor solo mientras el servicio de catálogo tiene quórum.

Razones para la pérdida de quórum

WebSphere eXtreme Scale espera perder quórum para los escenarios siguientes:

- Un miembro de la JVM del servicio de catálogo falla
- Se produce una caída de la red
- Se produce pérdida de un centro de datos

WebSphere eXtreme Scale no pierde quórum en el escenario siguiente:

- Detención de una instancia de servidor de catálogo con el mandato **stopOgServer** o cualquier otra acción administrativa. El sistema sabe que la instancia del servidor se ha detenido, lo que es distinto a una caída de la red o una anomalía de JVM.

Si el servicio de catálogo pierde quórum, espera a que se restablezca el quórum. Mientras el servicio de catálogo no tiene quórum, ignora los sucesos de los servidores de contenedor. Los servidores de contenedor continúan intentando las solicitudes rechazadas por el servidor de catálogo durante este tiempo. La pulsación se suspende hasta que se restablece un quórum.

Pérdida de quórum debida a una anomalía de JVM

Un servidor de catálogo que falla causa que se pierda el quórum. Si falla una JVM, se debe sustituir el quórum lo antes posible. El servicio de catálogo anómalo no puede volver a unirse a la cuadrícula de datos hasta que se haya sustituido el quórum.

Pérdida de quórum debido a la caída de la red

WebSphere eXtreme Scale está diseñado para esperar la posibilidad de caídas de la red. Una caída de la red es cuando se produce una pérdida temporal de conectividad entre los centros de datos. Las caídas de la red normalmente son transitorias y desaparecen en cuestión de segundos o minutos. Mientras WebSphere eXtreme Scale intenta mantener el funcionamiento normal durante el periodo de la caída de la red, una interrupción temporal se considera un único suceso de anomalía. Se espera arreglar la anomalía y a continuación se reanuda el funcionamiento normal sin que sea necesaria ninguna acción.

Una caída de la red de larga duración se puede clasificar como un apagón sólo a través de la intervención del usuario. Es necesario sustituir el quórum en un lado de la caída de la red para que el suceso se clasifique como apagón.

Ciclos de la JVM del servicio de catálogo

Si un servidor de catálogo se detiene utilizando el mandato **stopOgServer**, el quórum pierde un servidor. Los demás servidores aún tienen el quórum. Si se reinicia el servidor de catálogo, se restablecerá el quórum en el número anterior.

Consecuencias de la pérdida de quórum

Si una JVM de contenedor fallara mientras se ha perdido el quórum, la recuperación no se produciría hasta que finalizara la caída de la red. En un escenario de apagón, la recuperación no se produce hasta que se ejecuta el mandato de sustitución de quórum. La pérdida de quórum y una anomalía de contenedor se consideran una anomalía doble, lo que es un suceso inusual. Debido a la anomalía doble, es posible que las aplicaciones pierdan acceso de grabación a los datos almacenados en la JVM anómala. Cuando se restaura el quórum, se produce la recuperación normal.

De forma similar, si intenta iniciar un contenedor durante un suceso de pérdida de quórum, el contenedor no se inicia.

La conectividad total de cliente está autorizada durante la pérdida de quórum. Si no se produce ninguna anomalía de contenedor, ni ningún problema de conectividad durante el suceso de pérdida de quórum, los clientes pueden seguir interactuando de forma completa con los servidores de contenedor.

Si se produce una caída de la red, es posible que algunos clientes no tengan acceso a copias primarias o de réplica de los datos hasta que se haya resuelto la caída de la red.

Se pueden iniciar nuevos clientes porque debe existir una JVM de servicio de catálogo en cada centro de datos. Por lo tanto, como mínimo un cliente puede contactar con un servidor de catálogo durante un suceso de caída de la red.

Recuperación del quórum

Si se pierde el quórum por alguna razón, cuando este se restablece se ejecuta un protocolo de recuperación. Cuando se produce un suceso de pérdida de quórum, se suspenden todas las comprobaciones de integridad y concurrencia para los grupos principales y también se ignoran los informes de anomalías. Una vez recuperado el quórum, el servicio de catálogo comprueba todos los grupos principales para determinar inmediatamente su pertenencia. Los fragmentos alojados anteriormente en las JVM de contenedor notificadas como anómalas se recuperan. Si se perdierán fragmentos primarios, las réplicas supervivientes pasarían a ser fragmentos primarios. Si se perdieran fragmentos de réplica, se crearían fragmentos de réplica adicionales en los supervivientes.

Alteración temporal del quórum

Sustituya el quórum solo cuando se haya producido una anomalía del centro de datos. La pérdida de quórum debida a una anomalía de JVM de servicio de catálogo o una caída de la red se recupera automáticamente después de que se reinicie la JVM de servicio de catálogo o finalice la caída de la red.

Los administradores son los únicos con conocimiento de una anomalía de centro de datos. WebSphere eXtreme Scale trata una caída de la red y un apagón de forma similar. Debe informar al entorno de WebSphere eXtreme Scale de este tipo de anomalías con el mandato **xscmd -c overrideQuorum**. Este mandato indica al servicio de catálogo que suponga que el quórum se consigue con la pertenencia actual, y que tiene lugar la recuperación completa. Al emitir un mandato de sustitución de quórum, garantiza que las JVM del centro de datos anómalo han fallado realmente y no tienen oportunidades de recuperación.

La siguiente lista considera algunos escenarios para alterar temporalmente el quórum. En este escenario, tiene tres servidores de catálogo: A, B y C.

- **Caída de la red:** el servidor de catálogo C está aislado temporalmente. El servicio de catálogo pierde quórum y espera a que finalice la caída de la red. Una vez que se la solucionado la caída de la red, el servidor de catálogo C se vuelve a unir al dominio de servicio de catálogo y se restablece el quórum. La aplicación no verá ningún problema durante este momento.
- **Anomalía temporal:** durante una anomalía temporal, el servidor de catálogo C falla y el servicio de catálogo pierde el quórum. Debe sustituir el quórum. Después de que se restablezca el quórum, puede reiniciar el servidor de catálogo

C. El servidor de catálogo C se une de nuevo al dominio de servicio de catálogo cuando se reinicia. La aplicación no verá ningún problema durante este momento.

- **Anomalía del centro de atos:** verifica que el centro de datos ha fallado y que se ha aislado en la red. A continuación, emite el mandato **xscmd -c overrideQuorum**. Los dos centros de datos supervivientes ejecutan una recuperación completa sustituyendo los fragmentos que estaban alojados en el centro de datos anómalo. El servicio de catálogo ahora se ejecuta con un quórum completo de los servidores de catálogo A y B. La aplicación podría ver retardos y excepciones durante el intervalo entre el inicio del apagón y la sustitución del quórum. Una vez que se ha sustituido el quórum, la cuadrícula de datos se recupera y se reanuda el funcionamiento normal.
- **Recuperación del centro de datos:** los centros de datos supervivientes ya se están ejecutando con sustitución de quórum. Cuando el centro de datos que contiene el servidor de catálogo C se reinicia, todas las JVM del centro de datos se deben reiniciar. A continuación, el servidor de catálogo C se une de nuevo al dominio de servicio de catálogo existente y el valor de quórum vuelve a la situación normal sin intervención del usuario.
- **Anomalía de centro de datos y caída de la red:** el centro de datos que contiene el servidor de catálogo C falla. El quórum se sustituye y se recupera en los demás centros de datos. Si se produce una caída de la red entre los servidores de catálogo A y B, se aplican las reglas de recuperación de caída de red normales. Una vez que se soluciona la caída de la red, el quórum se restablece y se produce la recuperación necesaria de la pérdida de quórum.

Comportamiento del contenedor durante la pérdida de quórum

Los contenedores alojan uno o más fragmentos. Los fragmentos son primarios o réplicas para una partición específica. El servicio de catálogo asigna fragmentos a un contenedor y el servidor de contenedor utiliza esa asignación hasta que llegan nuevas instrucciones del servicio de catálogo. Por ejemplo, un fragmento primario continúa intentando la comunicación con sus fragmentos de réplica durante caídas de la red, hasta que el servicio de catálogo proporciona instrucciones adicionales al fragmento primario.

Comportamiento de réplica síncrona

El fragmento primario puede aceptar transacciones nuevas mientras la conexión está interrumpida si el número de réplicas en línea está como mínimo en el valor de la propiedad **minsync** para el conjunto de correlaciones. Si se procesa alguna transacción nueva en el fragmento primario mientras el enlace a la réplica síncrona está interrumpido, la réplica se vuelve a sincronizar con el estado actual del primario cuando se restablece el enlace.

No configure la réplica síncrona entre los centros de datos o mediante un enlace de estilo WAN.

Comportamiento de réplica asíncrona

Mientras la conexión está interrumpida, el fragmento primario puede aceptar nuevas transacciones. El fragmento primario almacena en el almacenamiento intermedio los cambios hasta un límite. Si la conexión con la réplica se restablece antes de que se alcance el límite, la réplica se actualiza con los cambios del almacenamiento intermedio. Si se ha alcanzado el límite, el primario destruye la

lista del almacenamiento intermedio y cuando se vuelve a conectar la réplica, se borra y se vuelve a sincronizar.

Comportamiento del cliente durante la pérdida de quórum

Los clientes siempre pueden conectarse al servidor de catálogo para realizar el programa de arranque de la cuadrícula de datos independientemente de si el dominio de servicio de catálogo tiene quórum o no. El cliente intenta conectarse a cualquier instancia de servicio de catálogo para obtener una tabla de direccionamiento y a continuación interactuar con la cuadrícula de datos. La conectividad de red puede impedir que el cliente interactúe con algunas particiones debido a la configuración de la red. El cliente podría conectarse a réplicas locales para datos remotos si se ha configurado para ello. Los clientes no pueden actualizar los datos si la partición primaria para estos datos no está disponible.

Réplicas y fragmentos

Con eXtreme Scale, una base de datos o fragmento en memoria puede duplicarse desde una Máquina virtual Java (JVM) en otra. Un fragmento representa una partición que se coloca en un contenedor. En un contenedor pueden existir varios fragmentos que representan distintas particiones. Cada partición tiene una instancia que es un fragmento primario y un número configurable de fragmentos de réplica. Los fragmentos de réplica son síncronos o asíncronos. Los tipos y la ubicación de los fragmentos de réplica los determina eXtreme Scale mediante una política de despliegue, que especifica el número mínimo y máximo de los fragmentos síncronos y asíncronos.

Tipos de fragmento

La réplica utiliza tres tipos de fragmentos:

- Fragmento primario
- Réplica síncrona
- Réplica asíncrona

El fragmento primario recibe todas las operaciones de inserción, actualización y eliminación. El fragmento primario añade y elimina réplicas, duplica datos en las réplicas y gestiona confirmaciones y retrotracciones de transacciones.

Las réplicas síncronas mantienen el mismo estado que el fragmento primario. Cuando un fragmento primario duplica datos en una réplica síncrona, la transacción no se confirma hasta que se confirma en la réplica síncrona.

Las réplicas asíncronas pueden o no estar en el mismo estado que el fragmento primario. Cuando un fragmento primario duplica datos en una réplica asíncrona, el fragmento primario no espera a que la réplica asíncrona se confirma.

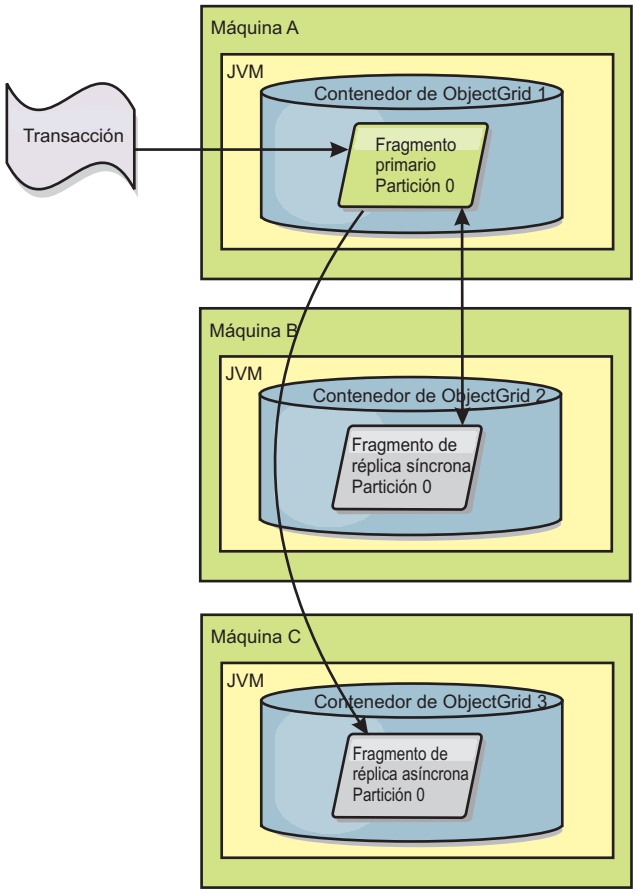


Figura 30. Vía de acceso de comunicación entre un fragmento primario y fragmentos de réplica

Fragmentos mínimos de réplicas síncronas

Cuando un fragmento primario se prepara para confirmar datos, comprueba cuántos fragmentos de réplicas síncronas han votado por confirmar la transacción. Si la transacción normalmente se procesa en la réplica, vota por confirmarse. Si se ha producido algún error en la réplica síncrona, se vota por no confirmarse. Antes de que un fragmento primario se confirme, el número de fragmentos de réplicas síncronas que votan por confirmarse deben satisfacer el valor minSyncReplica en la política de despliegue. Cuando el número de fragmentos de réplicas síncronas que votan por confirmarse es demasiado bajo, el fragmento primario no confirma la transacción y resulta en un error. Esta acción garantiza que la cantidad necesaria de réplicas síncronas esté disponible con los datos correctos. Las réplicas síncronas que han encontrado errores se han vuelto a registrar para corregir su estado. Si desea más información sobre cómo volver a realizar el registro, consulte Recuperación del fragmento de réplica.

El fragmento primario genera un error ReplicationVotedToRollbackTransactionException si muy pocas réplicas síncronas han votado por confirmarse.

Réplica y cargadores

Normalmente, un fragmento primario graba de forma síncrona en una base de datos los cambios a través del cargador. El cargador y la base de datos siempre

están sincronizados. Cuando el fragmento primario realiza una migración tras error en un fragmento de réplica, es posible que la base de datos y el cargador no estén sincronizados. Por ejemplo:

- El fragmento primario puede enviar la transacción a la réplica y luego sufrir una anomalía antes de confirmarse en la base de datos.
- El fragmento primario puede confirmarse en la base de datos y luego sufrir una anomalía antes de enviar la réplica.

Los dos enfoques llevan a que la réplica esté una transacción por delante o por detrás de la base de datos. Esta situación no es aceptable. eXtreme Scale utiliza un protocolo especial y un contrato con la implementación de cargador para resolver esta cuestión sin la confirmación de dos fases. El protocolo es el siguiente:

Lado del fragmento primario

- Enviar la transacción junto con resultados de transacciones anteriores.
- Grabar en la base de datos e intentar confirmar la transacción.
- Si la base de datos se confirma, confirmar en eXtreme Scale. Si la base de datos no está confirmada, retrotraer la transacción.
- Grabar el resultado.

Lado de la réplica

- Recibir una transacción y almacenarla en el almacenamiento intermedio.
- Para todos los resultados, enviar con la transacción, confirmar todas las transacciones almacenadas en el almacenamiento intermedio y descartar todas las transacciones retrotraídas.

Lado de réplica en la migración tras error

- Para todas las transacciones almacenadas en el almacenamiento intermedio, proporcionar las transacciones para el cargador y éste intenta confirmar las transacciones.
- Es necesario grabar el cargador para que cada transacción sea idempotente.
- Si la transacción ya está en la base de datos, el cargador no realizar ninguna operación.
- Si la transacción no está en la base de datos, el cargador aplica la transacción.
- Una vez que se han procesado todas las transacciones, el nuevo fragmento primario puede empezar a atender a solicitudes.

Este protocolo garantiza que al base de datos está en el mismo nivel que el estado del nuevo fragmento primario.

Colocación de fragmentos

El servicio de catálogo se ocupa de colocar los fragmentos. Cada ObjectGrid tiene varias particiones, cada una de las cuales tiene un fragmento primario y un conjunto opcional de fragmentos réplica. El servicio de catálogo asigna los fragmentos equilibrándolos de forma que se distribuyan equitativamente en los servidores de contenedor disponibles. Los fragmentos de réplica y primarios para la misma partición nunca se colocan en el mismo servidor de contenedor o en la misma dirección IP, a menos que la configuración esté en modalidad de desarrollo.

Si se inicia un nuevo servidor de contenedor, eXtreme Scale recupera fragmentos de servidores de contenedor relativamente sobrecargados en el nuevo servidor de contenedor vacío. Este movimiento de fragmentos habilita el escalado horizontal.

Escalar hacia fuera

Escalar hacia fuera significa que cuando se añaden servidores de contenedor adicionales a una cuadrícula de datos, eXtreme Scale intenta mover los fragmentos existentes, primarios o réplicas, del conjunto anterior de servidores de contenedor al nuevo conjunto. Este movimiento amplía la cuadrícula de datos para aprovechar el procesador, la red y la memoria de los servidores de contenedor que se acaban de añadir. Este movimiento también equilibra la cuadrícula de datos e intenta garantizar que cada JVM de la cuadrícula de datos aloja la misma cantidad de datos. Cuando la cuadrícula de datos se amplía, cada servidor contiene un subconjunto menor de la cuadrícula total. eXtreme Scale presupone que los datos se distribuyen uniformemente entre las particiones. Esta ampliación favorece la operación de escalar hacia fuera.

Escalar hacia dentro

Escalar hacia dentro significa que si falla una JVM , eXtreme Scale intenta reparar el daño. Si la JVM donde se ha producido la anomalía tenía una réplica, eXtreme Scale sustituye la réplica perdida mediante la creación de una nueva réplica en una JVM superviviente. Si la JVM donde se ha producido la anomalía tenía un fragmento primario, eXtreme Scale busca la mejor réplica entre los supervivientes y promociona la réplica para que sea el nuevo fragmento primario. eXtreme Scale sustituye la réplica promocionada por una nueva réplica creada en los servidores restantes. Para mantener la escalabilidad, eXtreme Scale conserva el recuento de réplicas para las particiones, a medida que se producen anomalías en los servidores.

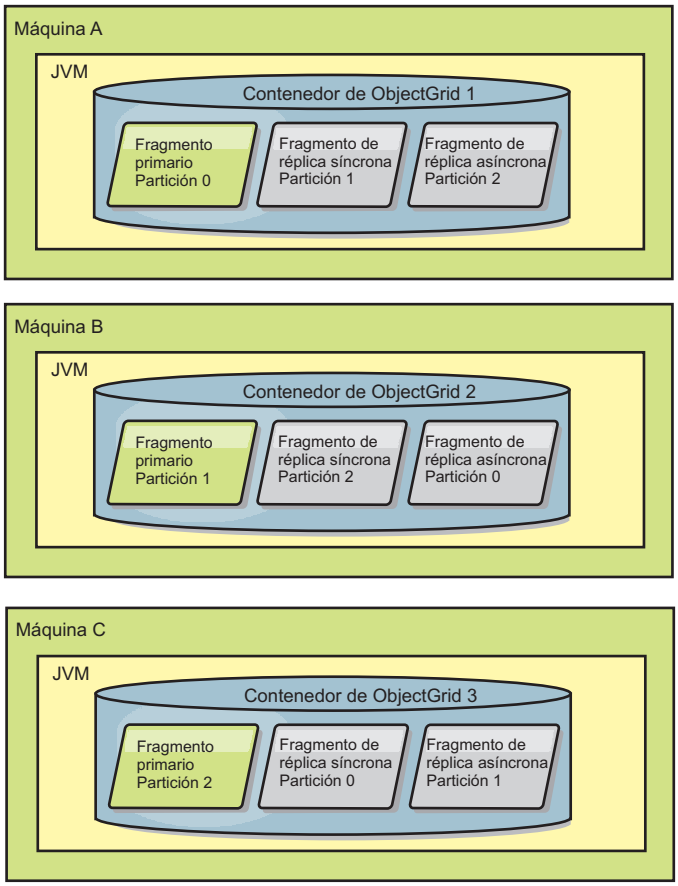


Figura 31. Colocación de un conjunto de correlaciones de ObjectGrid con una política de despliegue de 3 particiones con un valor minSyncReplicas de 1, un valor maxSyncReplicas de 1 y un valor maxAsyncReplicas de 1

Lectura de réplicas

Puede configurar conjuntos de correlaciones como, por ejemplo, que un cliente está autorizado para leer una réplica, en lugar de estar limitado sólo a los fragmentos primarios.

A menudo, puede resultar provechoso permitir a las réplicas actuar como más que unos primarios simplemente potentes en caso de anomalías. Por ejemplo, los conjuntos de correlaciones se pueden configurar para permitir que se direccionen operaciones de lectura a réplicas estableciendo la opción replicaReadEnabled de MapSet en true. El valor predeterminado es false.

Si desea más información sobre el elemento MapSet, consulte el tema sobre el archivo XML del descriptor de política de despliegue en *Guía de administración*.

La habilitación de lectura de réplicas puede mejorar el rendimiento distribuyendo las peticiones a más máquinas virtuales Java™. Si la opción no está habilitada, todas las peticiones de lectura como los métodos ObjectMap.get o Query.getResultIterator se direccionan al primario. Cuando replicaReadEnabled está definido en true, algunas peticiones get podrían devolver datos obsoletos, así pues una aplicación que utiliza esta opción debe poder tolerar esta posibilidad. Sin embargo, no se producirá ningún fallo de la memoria caché. Si los datos no están en la réplica, la petición get se direcciona al primario y se vuelve a intentar.

La opción replicaReadEnabled se puede utilizar tanto con la réplica síncrona como con la asíncrona.

Equilibrio de carga entre réplicas

El equilibrio de carga entre réplicas normalmente se utiliza sólo cuando los clientes almacenan en la memoria caché datos que almacenan siempre o cuando los clientes utilizan el bloqueo pesimista.

eXtreme Scale, salvo que se configure de otra manera, envía todas las solicitudes de lectura y grabación al servidor primario para un grupo de réplicas determinado. El fragmento primario debe atender todas las solicitudes de los clientes. Es posible que desee permitir que las solicitudes de lectura se envíen a las réplicas del fragmento primario. Si envía solicitudes de lectura a las réplicas la carga de las solicitudes de envío se podrá compartir entre varias JVM (Java Virtual Machines). No obstante, el uso de réplicas para las solicitudes de lectura puede generar repuestas incoherentes.

El equilibrio de carga entre réplicas normalmente se utiliza sólo cuando los clientes almacenan en la memoria caché datos que almacenan siempre o cuando los clientes utilizan el bloqueo pesimista.

Si lo datos se almacenan en la memoria caché constantemente y luego se invalidan en la memoria caché cercana del cliente, como resultado el fragmento primario debe detectar un índice de solicitudes get relativamente alto de los clientes. Asimismo, en modalidad de bloqueo pesimista, no existe memoria caché local, y por ello todas las solicitudes se envían al fragmento primario.

Si los datos son relativamente estáticos o si no se utiliza la modalidad pesimista, el envío de solicitudes de lectura a la réplica no tendrá un gran impacto en el rendimiento. La frecuencia de las solicitudes get de los clientes con memoria caché que están llenas de datos no es alta.

Cuando un cliente se inicia, su memoria caché cercana está vacía. Las solicitudes de memoria caché para la memoria caché vacía se remiten al fragmento primario. La memoria caché del cliente obtendrá datos con el tiempo, lo que provocará que la carga de solicitudes baje. Si una gran cantidad de clientes se inicia simultáneamente, la carga puede ser significativa y la lectura de réplicas puede ser una opción de rendimiento apropiada.

Ciclos de vida de los fragmentos

Los fragmentos pasan por estados y sucesos diferentes para poder admitir operaciones de réplica. El ciclo de vida de un fragmento incluye el paso a estar en línea, el tiempo de ejecución, la conclusión, la migración tras error y el manejo errores. Los fragmentos se pueden trasladar de un fragmento de réplica a un fragmento primario para manejar los cambios del estado del servidor.

Sucesos de ciclo de vida

Cuando se colocan y se inician los fragmentos réplica, pasan por una serie de sucesos hasta llegar a estar en línea y en modalidad de escucha.

Fragmento primario

El servicio de catálogo coloca un fragmento primario para una partición. El servicio de catálogo también equilibra las ubicaciones de los fragmentos primarios y e inicia la sustitución por anomalía para fragmentos primarios.

Cuando un fragmento se convierte en un fragmento primario, recibe una lista de réplicas del servicio de catálogo. El fragmento primario nuevo crea un grupo de réplicas y las registra.

Cuando el fragmento primario está listo, se muestra un mensaje de listo para operaciones empresariales en el archivo `SystemOut.log` del contenedor donde se esté ejecutando. El mensaje abierto o el mensaje CWOBJ1511I, lista el nombre de la correlación, el nombre del conjunto de correlaciones y el número de partición del fragmento primario que se ha iniciado.

```
CWOBJ1511I: mapName:mapSetName:partitionNumber (primary) is open for business.
```

Consulte el apartado "Colocación de fragmentos" en la página 111 para obtener más información sobre cómo coloca fragmentos el servicio de catálogo.

Fragmento réplica

Los fragmentos réplica los controla principalmente el fragmento primario a no ser que la réplica detecte un problema. Durante un ciclo de vida normal, el fragmento primario coloca, registra y elimina el registro de un fragmento de réplica.

Cuando el fragmento primario inicializa un fragmento réplica, un mensaje muestra el archivo de anotaciones cronológicas que describe dónde se ejecuta la réplica para indicar que el fragmento réplica está disponible. El mensaje abierto, o el mensaje CWOBJ1511I, lista el nombre de correlación, el nombre del conjunto de correlaciones y el número de partición del fragmento de réplica. Este mensaje es el siguiente:

```
CWOBJ1511I: mapName:mapSetName:partitionNumber (synchronous replica) is open for business.
```

o bien

```
CWOBJ1511I: mapName:mapSetName:partitionNumber (asynchronous replica) is open for business.
```

Fragmento réplica asíncrono: un fragmento réplica asíncrono sondea el fragmento primario para obtener los datos. La réplica ajustará automáticamente la temporización del sondeo si no recibe los datos del fragmento primario, que indica que se ha puesto al día con el fragmento primario. Se ajustará también si recibe un error que podría indicar que se ha producido un error en el fragmento primario o si hay un problema de red.

Cuando la réplica asíncrona comienza la réplica, imprime el mensaje siguiente en el archivo SystemOut.log para la réplica. Este mensaje podría imprimirse más de una vez por mensaje CWOBJ1511. Se imprimirá de nuevo si la réplica se conecta a un fragmento primario distinto o si se han añadido correlaciones de plantilla.

```
CWOBJ1543I: Se ha iniciado la réplica asíncrona objectGridName:mapsetName:partitionNumber o
ha seguido realizando la réplica desde el fragmento principal. Réplica de correlaciones: [mapName]
```

Fragmento réplica síncrono: cuando se inicia por primera vez el fragmento réplica asíncrono, no está todavía en modalidad de igual. Cuando un fragmento réplica está en modalidad de igual, recibe datos del fragmento primario a medida que los datos van entrando en el fragmento primario. Antes de pasar a la modalidad de igual, el fragmento réplica necesita una copia de todos los datos existentes en el fragmento primario.

La réplica síncrona copia los datos del fragmento primario similar a una réplica asíncrona sondeando los datos. Cuando copia los datos existentes del fragmento primario, cambia a modalidad de igual y comienza a recibir los datos a medida que el fragmento primario recibe los datos.

Cuando un fragmento réplica alcanza la modalidad de igual, imprime un mensaje en el archivo SystemOut.log de la réplica. El tiempo se refiere a la cantidad de tiempo que tardó el fragmento réplica en obtener todos los datos iniciales del fragmento primario. El valor de tiempo puede mostrarse como cero o muy bajo si el fragmento primario no tiene ningún dato que deba replicar.Puede que este mensaje se imprima más de una vez por mensaje CWOBJ1511. Se imprimirá de nuevo si la réplica se conecta a un fragmento primario distinto o si se han añadido correlaciones de plantilla.

```
CWOBJ1526I: Replica objectGridName:mapsetName:partitionNumber:mapName entering peer mode after X seconds.
```

Cuando el fragmento de réplica síncrono está en modalidad de igual, el fragmento primario debe hacer una réplica de las transacciones a todas las réplicas síncronas de modalidad de igual. Los datos del fragmento réplica síncrono permanecen al mismo nivel que los del fragmento primario. Si se establece un número mínimo de réplicas síncronas o minSync en la política de despliegue, ese número de réplicas síncronas debe votar confirmarse antes de que la transacción pueda confirmarse satisfactoriamente en el fragmento principal.

Sucesos de recuperación

Las operaciones de réplica se han diseñado para realizar recuperaciones a partir de sucesos de anomalías y errores. Si se produce una anomalía en un fragmento primario, otra réplica pasa a tener el control. Si las anomalías se producen en los fragmentos réplica, éstos intentarán recuperarse. El servicio de catálogo controla la colocación y las transacciones de fragmentos primarios nuevos o fragmentos réplica nuevos.

Un fragmento réplica se convierte en un fragmento primario

Un fragmento réplica se convierte en un fragmento primario por dos razones: el fragmento primario se ha detenido o ha fallado, o se ha realizado una decisión de equilibrio para mover el fragmento primario anterior a una nueva ubicación.

El servicio de catálogo selecciona un nuevo fragmento primario de los fragmentos réplica síncronos existentes. Si debe tener lugar un movimiento de fragmento primario y no hay réplicas, se colocará una réplica temporal para completar la transición. El fragmento primario nuevo registra todas las réplicas existentes y acepta transacciones como el nuevo fragmento primario. Si los fragmentos réplica existentes tienen el nivel correcto de datos, los datos actuales se conservan a medida que los fragmentos réplica se registran con el nuevo fragmento primario. Se sondearán las réplicas asíncronas con el nuevo fragmento primario.

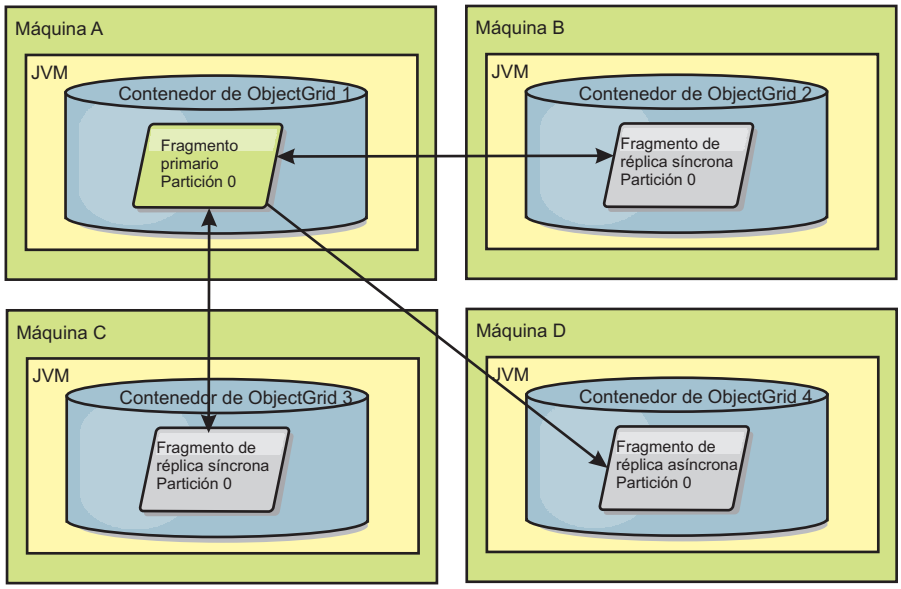


Figura 32. Colocación de ejemplo de un conjunto de correlaciones ObjectGrid para la partición partition0. La política del despliegue tiene un valor minSyncReplicas de 1, un valor maxSyncReplicas de 2 y un valor maxAsyncReplicas de 1.

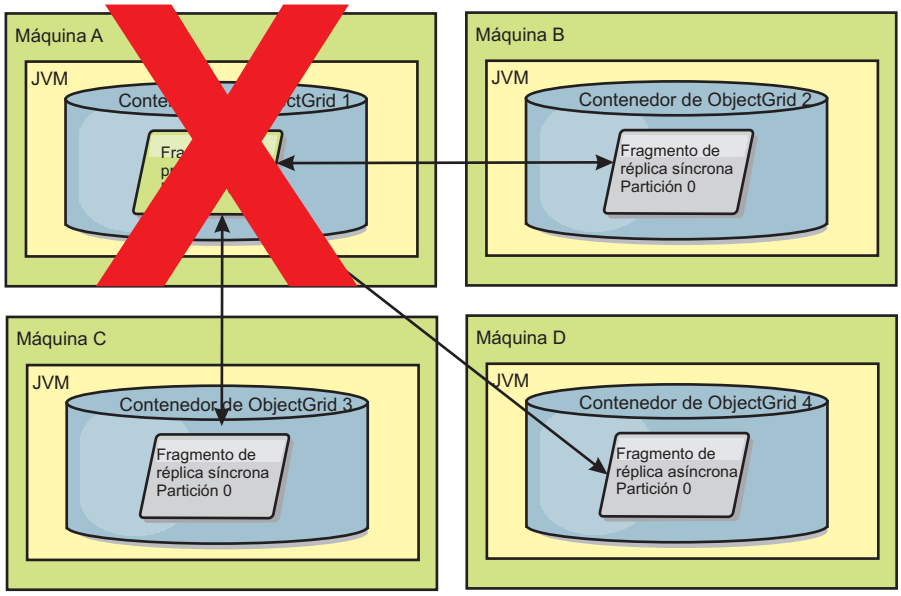


Figura 33. El contenedor del fragmento primario falla.

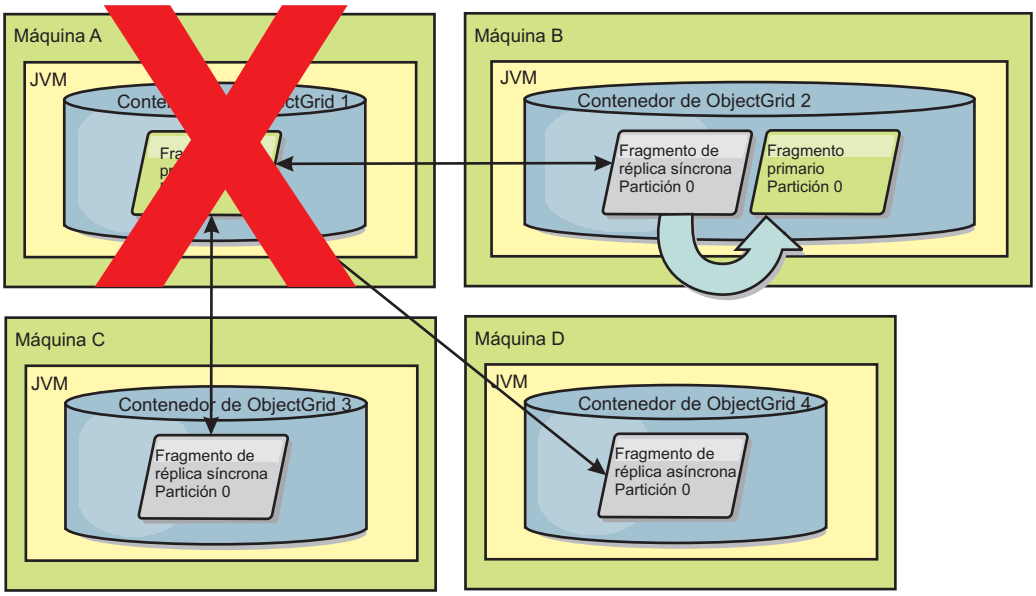


Figura 34. El fragmento de réplica síncrona en el contenedor 2 de ObjectGrid pasa a ser el fragmento primario.

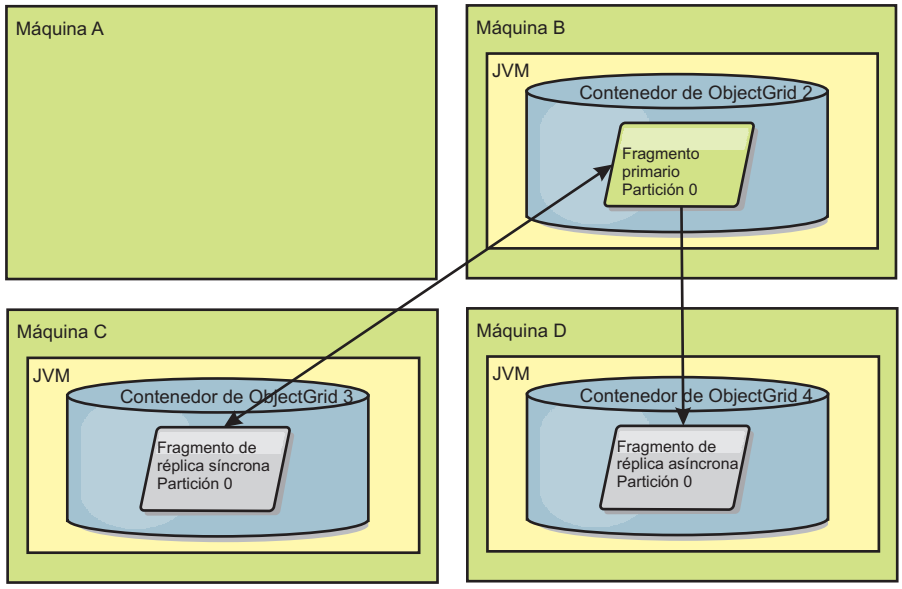


Figura 35. La máquina B contiene el fragmento primario. En función de cómo se establece la modalidad de reparación automática y la disponibilidad de los contenedores, un nuevo fragmento de réplica síncrona se podría colocar o no en una máquina.

Recuperación de un fragmento réplica

El fragmento primario controla al fragmento réplica síncrono. No obstante, si un fragmento réplica detecta un problema, puede desencadenar un suceso de volver a registrar para corregir el estado de los datos. La réplica borra los datos actuales y obtiene una nueva copia del fragmento primario.

Cuando un fragmento réplica inicia un suceso de volver a registrar, la réplica imprime un mensaje de anotaciones cronológicas.

```
CWOBJ1524I: Replica listener
objectGridName:mapSetName:partition must re-register with the primary.
Reason: Exception listed
```

Si una transacción provoca un error en un fragmento réplica durante el proceso, el fragmento réplica está en un estado desconocido. La transacción se ha procesado correctamente en el fragmento primario, pero se ha producido una anomalía en la réplica. Para corregir esta situación, la réplica inicia un suceso de volver a registrar. Con una nueva copia de los datos del fragmento primario, el fragmento réplica puede continuar. Si se vuelve a repetir el problema, el fragmento réplica no vuelve a registrarse de forma continuada. Si desea más información, consulte “Sucesos de errores”.

Sucesos de errores

Una réplica puede detener la réplica de datos si encuentra situaciones de error para las que no se puede recuperar la réplica.

Demasiados intentos de registro

Si una réplica desencadena un suceso de volver a registrar varias veces, pero no se confirman los datos correctamente, la réplica se detiene. Al detenerse, se evita que la réplica entre en un bucle infinito de sucesos de volver a registrarse. De manera predeterminada, un fragmento réplica intenta volver a registrarse tres veces seguidas antes de detenerse.

Si un fragmento réplica vuelve a registrarse demasiadas veces, imprimirá el siguiente mensaje en el archivo de anotaciones cronológicas.

```
CWOBJ1537E: objectGridName:mapSetName:partition exceeded the maximum number
of times to reregister (timesAllowed) without successful transactions..
```

Si la réplica no se recupera mediante operaciones de volver a registrarse, podría existir un problema generalizado con las transacciones relativas al fragmento réplica. Un problema posible podría ser que faltan recursos en la variable CLASSPATH si se produce un error al inflar las claves o valores de la transacción.

Anomalía al entrar en modalidad de igual

Si una réplica intenta entrar en modalidad de igual y se produce un error al procesar los datos en bloque del fragmento primario (datos del punto de control), la réplica concluye. Esto impide que la réplica se inicie con datos iniciales incorrectos. Puesto que recibe los mismos datos del primario si se vuelve a registrar, no se vuelve a intentar la réplica.

Si un fragmento réplica no puede entrar en modalidad de igual, imprimirá el siguiente mensaje en el archivo de anotaciones cronológicas:

```
CWOBJ1527W Replica objectGridName:mapSetName:partition:mapName failed to enter peer mode after numSeconds seconds.
```

En el archivo de anotaciones cronológicas se muestra otro mensaje que explica por qué la réplica no ha podido entrar en modalidad de igual.

Recuperación después de una anomalía al volver a registrarse o de la modalidad de igual

Si una réplica falla al volver a registrarse o al entrar en la modalidad de igual, la réplica está en un estado inactivo hasta que se produce un nuevo suceso de colocación. Un nuevo suceso de colocación podría ser un nuevo servidor que se

inicia o detiene. También puede iniciar un suceso de colocación utilizando el método triggerPlacement en el MBean PlacementServiceMBean.

Conjuntos de correlaciones para réplica

La réplica se habilita asociando BackingMaps a un conjunto de correlaciones.

Un conjunto de correlaciones es una colección de correlaciones que se categorizan por clave de partición. Esta clave de partición se obtiene de la clave del correlación individual efectuando una operación módulo entre hash y el número de particiones. Si un grupo de correlaciones de un conjunto de particiones tiene la clave de partición X, estas correlaciones se almacenarán en una partición X correspondiente de la cuadrícula de datos. Si otro grupo tiene una clave de partición Y, todas las correlaciones se almacenarán en una partición Y, y así sucesivamente. Además, los datos de la correlación se replican en función de la política definida en el conjunto de correlaciones, que solo se utiliza para topologías distribuidas de eXtreme Scale (innecesario para instancias locales).

Si desea más información, consulte “Particionamiento” en la página 78.

Se asigna a los conjuntos de correlaciones el número de particiones que tendrán y una política de réplica. La configuración de réplica del conjunto de correlaciones simplemente identifica el número de fragmentos de réplica síncronos y asíncronos que debería tener un conjunto de correlaciones además del fragmento primario. Por ejemplo, si va a haber una réplica síncrona y una réplica asíncrona, cada una de las BackingMaps asignadas al conjunto de correlaciones tendrá un fragmento de réplica distribuido automáticamente en el conjunto de contenedores disponibles para eXtreme Scale. La configuración de réplica también puede permitir que los clientes lean datos de servidores duplicados de forma síncrona. Esto puede esparcir la carga de las solicitudes de lectura entre servidores adicionales en eXtreme Scale. La réplica sólo tiene un impacto de modelo de programación cuando se realiza la precarga de BackingMaps.

Visión general del proceso de transacciones

WebSphere eXtreme Scale utiliza las transacciones como su mecanismo para la interacción con datos.

Para interactuar con los datos, la hebra de la aplicación requiere su propia sesión. Si la aplicación desea utilizar el ObjectGrid en una hebra, llame a uno de los métodos ObjectGrid.getSession para obtener una hebra. Con la sesión, la aplicación puede trabajar con los datos almacenados en las correlaciones de ObjectGrid.

Cuando una aplicación utiliza un objeto Session, la sesión debe estar en el contexto de una transacción. Una transacción empieza o se confirma y retrotrae mediante los métodos begin, commit y rollback en el objeto Session. Las aplicaciones también pueden funcionar en la modalidad de confirmación automática, en la que Session empieza automáticamente y confirma una transacción, siempre que se realiza una operación en la correlación. Una modalidad de confirmación automática no puede agrupar varias operaciones en una única transacción, de forma que es la opción más lenta si crea un proceso por lotes de varias operaciones en una única transacción. Sin embargo, para las transacciones que sólo contienen una operación, la confirmación automática es la opción más rápida.

Transacciones

Las transacciones tienen muchas ventajas para el almacenamiento de datos y la manipulación. Puede utilizar las transacciones para proteger la cuadrícula de datos

de los cambios simultáneos, para aplicar varios cambios como una unidad simultánea, para replicar datos y para implementar un ciclo de vida para los bloqueos en los cambios.

Cuando se inicia una transacción, WebSphere eXtreme Scale asigna una correlación de diferencias especial para mantener los cambios o copias actuales de pares de clave y valor que la transacción utiliza. Normalmente, cuando se accede a un par de clave y valor, el valor se copia antes de que la aplicación reciba el valor. La correlación de diferencias rastrea todos los cambios para las operaciones como, por ejemplo, insert, update, get, remove, etc. Las claves no se copian porque se da por supuesto que son inmutables. Si se especifica un objeto ObjectTransformer, este objeto se utiliza para copiar el valor. Si la transacción utiliza el bloqueo optimista, también se realiza un seguimiento de las imágenes anteriores de los valores para su comparación cuando se confirma la transacción.

Si se retrotrae una transacción, se descarta la información de correlación de diferencias y se liberan los bloqueos de las entradas. Cuando se confirma una transacción, los cambios se aplican a las correlaciones y se liberan los bloqueos. Si se utiliza el bloqueo optimista, eXtreme Scale compara las versiones de imágenes anteriores de los valores con los valores incluidos en la correlación. Estos valores deben coincidir para que la transacción se confirme. Esta comparación permite un esquema de bloqueo de varias versiones, pero a costa de que se realicen dos copias cuando la transacción accede a la entrada. Se vuelven a copiar todos los valores y se almacena la nueva copia en la correlación. WebSphere eXtreme Scale realiza esta copia para evitar que la aplicación cambie la referencia de la aplicación por el valor después de una confirmación.

Puede evitar utilizar varias copias de la información. La aplicación puede guardar una copia utilizando el bloqueo pesimista en lugar del bloqueo optimista como coste de limitar la concurrencia. También se puede evitar la copia del valor durante la confirmación si la aplicación acepta no cambiar un valor después de la confirmación.

Ventajas de las transacciones

Utilice transacciones por las siguientes razones:

Mediante el uso de transacciones, puede:

- Retrotraer cambios si se produce una excepción o si la lógica empresarial necesita deshacer los cambios de estado.
- Para aplicar varios cambios como una unidad atómica durante la confirmación.
- Mantener y liberar bloqueos en los datos para aplicar varios cambios como una unidad atómica durante la confirmación.
- Proteger una hebra de los cambios simultáneos.
- Implementar un ciclo de vida para los bloqueos en cambios.
- Producir una unidad atómica de duplicación.

Tamaño de transacción

Las transacciones de mayor tamaño son más eficaces, especialmente para la réplica. Sin embargo, las transacciones de mayor tamaño pueden afectar de forma adversa a la concurrencia porque se mantienen durante más tiempo los bloqueos sobre entradas. Si utiliza transacciones de mayor tamaño, puede aumentar el rendimiento

de la réplica. El aumento de este rendimiento es importante cuando se precarga una correlación. Pruebe con distintos tamaños de lotes para determinar lo que funciona mejor en cada caso.

Las transacciones de mayor tamaño también son útiles con los cargadores. Si se está utilizando un cargador que puede realizar el proceso por lotes de SQL, son posibles aumentos significativos de rendimiento en función de la transacción y las reducciones significativas de la carga en el lado de la base de datos. Esta ganancia en el rendimiento dependerá de la implementación del cargador.

Modalidad de confirmación automática

Si no se ha iniciado de forma activa ninguna transacción, cuando una aplicación interactúa con un objeto ObjectMap, empieza una operación automática de inicio y confirmación en nombre de la aplicación. Esta operación automática de inicio y confirmación funciona, pero impide que la retrotracción y el bloqueo funcionen de forma eficaz. La velocidad de réplica síncrona se ve afectado debido al tamaño de transacción muy pequeño. Si utiliza una aplicación de gestor de entidades, no utilice la modalidad de confirmación automática porque los objetos que busca el método EntityManager.find se convierten inmediatamente en no gestionados en la devolución del método y dejan de poderse utilizar.

Coordinadores de transacciones externos

Normalmente, las transacciones se inician con el método session.begin y finalizan con el método session.commit. Sin embargo, cuando se incorpora eXtreme Scale, las transacciones podrían iniciarse y terminarse a través de un coordinador de transacciones externo. Si utiliza un coordinador de transacciones externas, no tendrá que llamar al método session.begin y finalizar el método session.commit. Si utiliza WebSphere Application Server, puede utilizar el plug-in WebSphereTranscationCallback.

Atributo CopyMode

Puede ajustar el número de copias definiendo el atributo CopyMode de los objetos BackingMap u ObjectMap en el archivo XML de descriptor de ObjectGrid.

Puede ajustar el número de copias definiendo el atributo CopyMode de los objetos BackingMap u ObjectMap. La modalidad de copia tiene los siguientes valores:

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

El valor COPY_ON_READ_AND_COMMIT es el valor predeterminado. El valor COPY_ON_READ copia los datos iniciales recuperados, pero no copia durante la confirmación. Esta modalidad es segura si la aplicación no modifica un valor después de confirmar una transacción. El valor NO_COPY no copia datos, que sólo es seguro para los datos de sólo lectura. Si los datos nunca cambian, no tendrá que copiarlos por razones de aislamiento.

Tenga cuidado cuando utilice el valor del atributo NO_COPY con las correlaciones que se pueden actualizar. WebSphere eXtreme Scale utiliza la copia en el primer toque para permitir la retrotracción de la transacción. La aplicación sólo ha

cambiado la copia y, como resultado, eXtreme Scale descarta la copia. Si se utiliza el valor de atributo NO_COPY, y la aplicación modifica el valor confirmado, no es posible completar una retrotracción. Si se modifica el valor confirmado comportará problemas con índices, réplica, etc, porque los índices y las réplicas se actualizan cuando se confirma la transacción. Si modifica los datos confirmados y, a continuación, retrotrae la transacción, que en realidad no se retrotrae, los índices no se actualizan y la réplica no tiene lugar. Otras hebras pueden ver los cambios no confirmados inmediatamente, incluso si tienen bloqueos. Utilice el valor de atributo NO_COPY para las correlaciones de sólo lectura o para aplicaciones que completan la copia apropiada antes de modificar el valor. Si utiliza el valor de atributo NO_COPY y llama al soporte de IBM con un problema de integridad de datos, se le solicitará que reproduzca el problema con la modalidad de copia establecida en COPY_ON_READ_AND_COMMIT.

El valor COPY_TO_BYTES almacena valores en la correlación de un formato serializado. En el momento de lectura, eXtreme Scale infla el valor a partir de un formato serializado y en el momento de confirmación almacena el valor en un formato serializado. Con este método, se produce una copia durante la lectura y la confirmación.

La modalidad de copia predeterminada para una correlación se puede configurar en el objeto BackingMap. También puede cambiar la modalidad de copia en las correlaciones antes de iniciar una transacción mediante el uso del método ObjectMap.setCopyMode.

A continuación, aparece un ejemplo de un fragmento de código de la correlación de respaldo de un archivo `objectgrid.xml` que muestra cómo establecer la modalidad de copia para una correlación de respaldo dada. Este ejemplo da por supuesto que utiliza `cc` como espacio de nombres de `objectgrid/config`.

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

Gestor de bloqueo

Al configurar una estrategia de bloqueo, se crea un gestor de bloqueo que la correlación de respaldo mantenga la coherencia de entradas de la memoria caché.

Configuración del gestor de bloqueos

Cuando se utiliza una estrategia de bloqueo PESSIMISTIC u OPTIMISTIC, se crea un gestor de bloqueos para BackingMap. El gestor de bloqueos utiliza una correlación hash para realizar un seguimiento de las entradas bloqueadas por una o más transacciones. Cuantas más entradas de correlación existan en la correlación hash, mayor será el grupo de bloqueos con un buen rendimiento. El riesgo de las colisiones de sincronización de Java es menor a medida que crece el número de grupos. Un número mayor de grupos también implica mayor simultaneidad. Los ejemplos anteriores muestran cómo una aplicación puede establecer el número de grupos de bloqueos que se deben utilizar en una instancia determinada de BackingMap.

Para evitar una excepción java.lang.IllegalStateException, debe llamarse al método setNumberOfLockBuckets antes que a los métodos initialize o getSession en la instancia de ObjectGrid. El parámetro del método setNumberOfLockBuckets es un entero primitivo de Java que especifica el número de grupos de bloqueo para utilizar. El uso de un número primo puede permitir una distribución uniforme de entradas de correlación en los grupos de bloqueos. Un buen punto de partida para obtener un mejor rendimiento es establecer el número de grupos de bloqueos en un 10 por ciento del número esperado de entradas de BackingMap.

Estrategias de bloqueo

Las estrategias de bloqueo pueden ser de tipo pesimista, optimista o ninguno. Para elegir la estrategia de bloqueo, debe tener en cuenta cuestiones como el porcentaje de cada tipo de operaciones que realizará, si utilizará un cargador o no, etc.

Los bloqueos son enlazados por transacciones. Puede especificar los siguientes valores de bloqueo:

- **Sin bloqueo**: la ejecución sin el valor de bloqueo es la más rápida. Si utiliza datos de sólo lectura, es posible que no necesite el bloqueo.
- **Bloqueo pesimista**: adquiere bloqueos sobre entradas y luego mantiene los bloqueos hasta que se realiza la confirmación. Esta estrategia de bloqueo proporciona una mayor coherencia a costa del rendimiento.
- **Bloqueo optimista**: toma una imagen anterior de cada registro que toca la transacción y compara la imagen con los valores de entrada actuales cuando se confirma la transacción. Si los valores de entrada cambian, la transacción se retrotrae. No se mantiene ningún bloqueo hasta el momento de la confirmación. Esta estrategia de bloqueo proporciona una mejor concurrencia que la estrategia pesimista, con el riesgo de que la transacción se retrotraiga y el coste de memoria de realizar una copia adicional de la entrada.

Establezca la estrategia de bloqueo en la BackingMap. No puede cambiar la estrategia de bloqueo para cada transacción. A continuación, aparece un fragmento de código XML de ejemplo que muestra cómo establecer la modalidad de bloqueo en una correlación utilizando el archivo XML, que da por supuesto que `cc` es el espacio de nombres para el espacio de nombres de `objectgrid/config`:

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

Bloqueo pesimista

Utilice la estrategia de bloqueo pesimista en operaciones de correlación de lectura y grabación cuando no es posible utilizar otra estrategia de bloqueo. Cuando se configura una correlación ObjectGrid para utilizar la estrategia de bloqueo pesimista, se obtiene un bloqueo de transacción pesimista para una entrada de correlación cuando una transacción obtiene por primera vez la entrada de BackingMap. El bloqueo pesimista se mantiene hasta que la aplicación completa la transacción. Por lo general, la estrategia de bloqueo pesimista se utiliza en las situaciones siguientes:

- Cuando BackingMap se configura con o sin un cargador y la información de creación de versiones no está disponible.
- Cuando BackingMap se utiliza directamente en una aplicación que necesita ayuda de eXtreme Scale para el control de simultaneidad.
- Cuando la información de creación de versiones está disponible, pero las transacciones de actualización colisionan con frecuencia en las entradas de respaldo, lo cual produce anomalías optimistas de actualización.

Como la estrategia de bloqueo pesimista tiene el mayor impacto sobre el rendimiento y la escalabilidad, esta estrategia sólo debe utilizarse para correlaciones de lectura y grabación, cuando no es viable ninguna otra estrategia de bloqueo. Por ejemplo, estas situaciones podrían incluir cuando se producen con frecuencia anomalías optimistas de actualización, o cuando es difícil para una aplicación gestionar la recuperación de una anomalía optimista.

Bloqueo optimista

En la estrategia de bloqueo optimista, se presupone que dos transacciones no pueden intentar actualizar la misma entrada de correlación mientras se ejecutan simultáneamente. Por este motivo, la modalidad de bloqueo no necesita mantenerse para el ciclo de vida de la transacción, porque ya que es improbable que más de una transacción actualice la entrada de correlación simultáneamente. Por lo general, la estrategia de bloqueo optimista se utiliza en las situaciones siguientes:

- Cuando BackingMap se configura con o sin un cargador y la información de creación de versiones está disponible.
- Cuando BackingMap tiene mayoritariamente transacciones que realizan operaciones de lectura. En BackingMap, las operaciones insert, update o remove no se producen con frecuencia en las entradas de correlaciones.
- Cuando una correlación BackingMap se inserta, actualiza o elimina con más frecuencia de lo que se lee, pero las transacciones rara vez colisionan en la misma entrada de correlación.

Al igual que en la estrategia de bloqueo pesimista, los métodos de la interfaz ObjectMap determinan cómo eXtreme Scale intenta automáticamente adquirir una modalidad de bloqueo para una entrada de correlación a la que se accede. No obstante, las diferencias entre las estrategias optimistas y pesimistas son:

- Al igual que la estrategia de bloqueo pesimista, los métodos get y getAll adquieren una modalidad de bloqueo S cuando se invoca el método. Sin embargo, con el bloqueo optimista, la modalidad de bloqueo S no se mantiene hasta que finaliza la transacción, sino que se libera antes de que el método vuelva a la aplicación. El propósito de adquirir la modalidad de bloqueo es que eXtreme Scale pueda garantizar que sólo los datos confirmados de otras transacciones sean visibles a la transacción actual. Después de que eXtreme Scale haya comprobado que los datos se han confirmado, la modalidad de bloqueo S se libera. Durante el ciclo de confirmación, se realiza una comprobación de la creación de versiones optimista para garantizar que ninguna otra transacción haya modificado la entrada de correlación después de que la transacción actual haya liberado su modalidad de bloqueo S. Si no se capta una entrada de la correlación antes de que se actualice, invalide o suprima, el tiempo de ejecución de eXtreme Scale capta de forma implícita la entrada de la correlación. Esta operación get implícita se realiza para obtener el valor actual en el momento en que la entrada se solicitó para modificarse.
- A diferencia de la estrategia de bloqueo pesimista, los métodos getForUpdate y getAllForUpdate se manejan exactamente igual que los métodos get y getAll cuando se utiliza la estrategia de bloqueo optimista. Es decir, se adquiere una modalidad de bloqueo S al inicio del método y se libera la modalidad de bloqueo S antes de que se devuelva a la aplicación.

Todos los otros métodos ObjectMap se manejan exactamente igual que si se manejaran para la estrategia de bloqueo pesimista. Es decir, cuando se invoca el método commit, se obtiene una modalidad de bloqueo X para cualquier entrada de correlación que se inserte, actualice, elimine, manipule o invalide, y la modalidad de bloqueo X se mantiene hasta que la transacción complete el proceso de confirmación.

En la estrategia de bloqueo optimista, se presupone que ninguna transacción que se ejecute simultáneamente con otra intentará actualizar la misma entrada de correlación. Por este motivo, la modalidad de bloqueo no necesita mantenerse durante toda la vida de la transacción ya que es improbable que más de una

transacción actualice la entrada de correlación simultáneamente. Sin embargo, como no se ha mantenido la modalidad de bloqueo, otra transacción simultánea podría actualizar de forma potencial la entrada de la correlación, después de que la transacción actual haya liberado su modalidad de bloqueo S.

Para manejar esta posibilidad, eXtreme Scale obtiene un bloqueo X durante el ciclo de confirmación y realiza una comprobación de la creación de versiones optimista para verificar que ninguna otra transacción haya modificado la entrada de correlación después de que la transacción actual haya leído la entrada de correlación en BackingMap. Si otra transacción ha modificado la entrada de correlación, se produce una anomalía en la comprobación de versiones y se muestra una excepción OptimisticCollisionException. Esta excepción obliga a la transacción actual retrotraerse y la aplicación debe volver a intentar toda la transacción. La estrategia de bloqueo optimista es muy útil cuando se producen sobre todo lecturas de una correlación y rara vez se producen actualizaciones de la misma entrada de correlación.

Sin bloqueo

Cuando un objeto BackingMap se configura para que no use ninguna estrategia de bloqueo, no se obtiene ningún bloqueo de transacción para una entrada de correlación.

No usar ninguna estrategia de bloqueo es útil cuando una aplicación es un gestor de persistencia como, por ejemplo, un contenedor EJB (Enterprise JavaBeans) o cuando una aplicación utiliza Hibernate para obtener los datos persistentes. En este escenario, BackingMap se configura sin cargador y el gestor de persistencia utiliza BackingMap como memoria caché de datos. En este escenario, el gestor de persistencia proporciona control de simultaneidad entre las transacciones que están accediendo a las mismas entradas de correlación.

WebSphere eXtreme Scale no necesita obtener ningún bloqueo de transacción para el control de simultaneidad. Esta situación presupone que el gestor de persistencia no libera sus bloqueos de transacción antes de actualizar la correlación de ObjectGrid con los cambios confirmados. Si el gestor de persistencia libera sus bloqueos, debe utilizarse una estrategia de bloqueo optimista o pesimista. Por ejemplo, suponga que el gestor de persistencia de un contenedor EJB está actualizando una correlación de ObjectGrid con los datos que se confirmaron en la transacción gestionada por el contenedor EJB. Si la actualización de la correlación de ObjectGrid se produce antes de que se liberen los bloqueos de transacción del gestor de persistencia, podrá utilizar la estrategia sin bloqueos. Si la actualización de la correlación de ObjectGrid se produce después de que se liberen los bloqueos de transacción del gestor de persistencia, debe utilizar la estrategia de bloqueo optimista o pesimista.

Otro escenario en el que se puede utilizar una estrategia sin bloqueos es cuando la aplicación utiliza BackingMap directamente y se ha configurado un cargador para la correlación. En este escenario, el cargador utiliza el soporte de control de simultaneidad proporcionado por un sistema de gestión de bases de datos relacionales (RDBMS) mediante el uso de Java Database Connectivity (JDBC) o Hibernate para acceder a los datos de la base de datos relacional. La implementación de cargador puede utilizar un acercamiento optimista o pesimista. Un cargador que utiliza un bloqueo optimista o un procedimiento de creación de versiones favorece un alto nivel de simultaneidad y rendimiento. Para obtener más información sobre cómo implementar un enfoque de bloqueo optimista, consulte la sección OptimisticCallback en la información sobre las consideraciones del

cargador en la *Guía de administración*. Si utiliza un cargador que utiliza el soporte de bloqueo pesimista de una programa de fondo subyacente, es posible que desee utilizar el parámetro forUpdate que se pasa en el método get de la interfaz Loader. Establezca este parámetro en true si el método getForUpdate de la interfaz ObjectMap ha sido utilizado por la aplicación para obtener los datos. El cargador puede utilizar este parámetro para determinar si debe solicitar un bloqueo actualizable en la fila que se está leyendo. Por ejemplo, DB2 obtiene un bloqueo que se puede actualizar si una sentencia SQL select contiene una cláusula `FOR UPDATE`. Este acercamiento ofrece la misma prevención de situaciones de punto muerto que la descrita en el apartado "Bloqueo pesimista" en la página 124.

Para obtener más información, consulte el tema sobre el manejo de bloqueos en la *Guía de programación* o el tema sobre el bloqueo de entrada de correlación en la *Guía de administración*.

Distribución de transacciones

Utilice JMS (Java Message Service) para los cambios de transacciones distribuidas entre las distintas capas o en entornos en plataformas combinadas.

JMS es un protocolo ideal para distribuir cambios entre distintos niveles o en entornos con diferentes plataformas. Por ejemplo, algunas aplicaciones que utilizan eXtreme Scale se podrían desplegar en IBM WebSphere Application Server Community Edition, Apache Geronimo o Apache Tomcat, mientras que otras aplicaciones se podrían ejecutar en WebSphere Application Server versión 6.x. JMS es ideal para los cambios distribuidos entre los iguales de eXtreme Scale en estos distintos entornos. El transporte de mensajes de High Availability Manager es muy rápido, pero sólo puede distribuir cambios a Máquinas virtuales Java que estén en un único grupo principal. JMS es más lento, pero permite que conjuntos más grandes y más diversos de clientes de aplicación puedan compartir un ObjectGrid. JMS es ideal si se comparten datos en un ObjectGrid entre un cliente grueso de Swing y una aplicación desplegada en WebSphere Extended Deployment.

El mecanismo de invalidación de clientes incorporado y la réplica de igual a igual son ejemplos de distribución de cambios transaccionales basados en JMS. Consulte la información sobre cómo configurar la réplica de igual a igual con JMS en la *Guía de administración* para obtener más información.

Implementación de JMS

JMS se implementa para distribuir los cambios de transacciones utilizando un objeto Java que se comporta como un ObjectGridEventListener. Este objeto puede propagar el estado de las cuatro formas siguientes:

1. Invalidación: las entradas desalojadas, actualizadas o suprimidas se eliminan de todas las Mäquinas virtuales Java de iguales al recibir el mensaje.
2. Invalidación condicional: la entrada sólo se desaloja si la versión local es la misma o más antigua que la versión del editor.
3. Envío: las entradas desalojadas, actualizadas, suprimidas o insertadas se añaden o se sobrescriben en todas las Máquinas virtuales Java de iguales al recibir el mensaje JMS.
4. Envío condicional: la entrada sólo se actualiza o se añade en el lado del receptor si la entrada local es menos reciente que la versión que se va a publicar.

Escuchar cambios de publicación

El plug-in implementa la interfaz ObjectGridEventListener para interceptar el suceso transactionEnd. Cuando eXtreme Scale invoca este método, el plug-in intenta convertir la lista LogSequence de cada correlación manipulada por la transacción a un mensaje JMS, que intentará publicar. El plug-in puede haberse configurado para publicar cambios de todas las correlaciones o de un subconjunto. Los objetos LogSequence se procesan para las correlaciones que tienen habilitada la publicación. La clase LogSequenceTransformer ObjectGrid serializa un objeto filtrado LogSequence de cada correlación en una corriente. Después de que todos los objetos LogSequences se serialicen en una corriente, se crea un objeto JMS ObjectMessage y se publica para un tema conocido.

Escuchar mensajes JMS y aplicarlos al objeto ObjectGrid local

El mismo plug-in también inicia una hebra que forma un bucle y recibe todos los mensajes publicados para un tema conocido. Cuando llega un mensaje, se pasa el contenido del mensaje a la clase LogSequenceTransformer, donde se convierte a un conjunto de objetos LogSequence. A continuación, se inicia una transacción de no escritura a través. Cada objeto LogSequence se proporciona al método Session.processLogSequence, que actualiza las correlaciones locales con los cambios. El método processLogSequence entiende la modalidad de distribución. La transacción se confirma y la memoria caché local refleja los cambios. Para obtener más información sobre cómo utilizar JMS para distribuir cambios de transacción, consulte la información sobre cómo distribuir cambios entre máquinas virtuales Java iguales en la *Guía de administración*.

Transacciones de partición única y transacciones entre cuadrículas de datos

La diferencia principal entre WebSphere eXtreme Scale y las soluciones de almacenamiento de datos tradicionales como las bases de datos relacionales o las bases de datos en memoria es el uso del particionamiento, que permite a la memoria caché realizar las escaladas de forma lineal. Los tipos importantes de transacciones a tener en cuenta son transacciones de partición única y transacciones de cada partición (entre cuadrículas de datos).

En general, las interacciones con la memoria caché se pueden categorizar como transacciones de una partición único o transacciones entre cuadrículas de datos, tal como se describe en la sección siguiente.

Transacciones de partición única

Las transacciones de partición única son el método preferible para interactuar con las memorias caché alojadas por WebSphere eXtreme Scale. Cuando una transacción está limitada a una única partición, de forma predeterminada, está limitada a una única Máquina virtual Java y, por lo tanto, un único sistema de servidor. Un servidor puede completar *M* número de estas transacciones por segundo y si tiene *N* sistemas, puede completar `M*N` transacciones por segundo. Si el negocio aumenta y debe doblar el rendimiento respecto a muchas de estas transacciones por segundo, puede doblar el valor *N* comprando más sistemas. Puede cumplir las demandas de capacidad sin modificar la aplicación, actualizar el hardware o, incluso, colocando la aplicación fuera de línea.

Además de permitir a la memoria caché realizar escaladas de forma significativa, las transacciones de partición única también maximizan la disponibilidad de la memoria caché. Cada transacción sólo depende de un sistema. Cualquiera de los

otros (N-1) sistemas puede falla sin que esto afecte al éxito o al tiempo de respuesta de la transacción. Por lo tanto, si ejecuta 100 sistemas y uno de ellas falla, sólo el 1 por ciento de las transacciones en curso en el momento en que falla el servidor se retrotrae. Después de que el servidor falle, WebSphere eXtreme Scale reubica las particiones alojadas por el servidor anómalo en los otros 99 sistemas. Durante este breve periodo, antes de que se complete la operación, los otros 99 sistemas pueden seguir completando transacciones. Sólo las transacciones que podrían implicar que las particiones que se están reubicando se bloqueen. Después de que se complete el proceso de migración tras error, la memoria caché puede seguir ejecutándose, plenamente operativa a un 99 por ciento de su capacidad de rendimiento original. Después de que se sustituya un servidor anómalo y se devuelva a la cuadrícula de datos, la memoria caché vuelva al 100 por cien de la capacidad de rendimiento.

Transacciones entre cuadrículas de datos

En términos de rendimiento, disponibilidad y escalabilidad, las transacciones entre cuadrículas de datos son lo contrario a la transacciones de una partición única. Las transacciones entre cuadrículas de datos acceden a cada partición y por lo tanto cada sistema de la configuración. Se solicita a cada sistema de la cuadrícula de datos que busque algunos datos y que a continuación devuelva el resultado. La transacción no se puede completar hasta que han respondido todos los sistemas y, por lo tanto, el rendimiento de toda la cuadrícula de datos está limitado por el sistema más lento. Añadir sistemas no hace que el sistema más lento sea más rápido y, por lo tanto, no mejora el rendimiento de la memoria caché.

Las transacciones entre cuadrículas de datos tiene un efecto similar en la disponibilidad. Ampliando el ejemplo anterior, si ejecuta 100 servidores y uno falla, el 100 por ciento de las transacciones que están en curso en el momento en el que falló el servidor se retrotraen. Después de que falle el servidor, WebSphere eXtreme Scale empieza a reubicar las particiones alojadas por dicho servidor a los otros 99 sistemas. Durante este tiempo, antes de que se complete el proceso de migración tras error, la cuadrícula de datos no puede procesar ninguna de estas transacciones. Después de que se complete el proceso de migración tras error, la memoria caché puede seguir ejecutándose, pero a una capacidad reducida. Si cada sistema de la cuadrícula de datos presta servicio a 10 particiones, 10 de los 99 sistemas restantes recibirán como mínimo una partición adicional como parte del proceso de migración tas error. Añadir una partición adicional aumentar la carga de trabajo de dicho sistema en un 10 por ciento, como mínimo. Debido a que el rendimiento de la cuadrícula de datos está limitado al rendimiento del sistema más lento en una transacción entre cuadrículas de datos, de promedio el rendimiento se reduce en un 10 por ciento.

Las transacciones de partición única son preferibles a las transacciones entre cuadrículas de datos para el escalado con una memoria caché de objetos distribuida con alta disponibilidad, como WebSphere eXtreme Scale. La maximización del rendimiento de estas clases de sistemas requiere el uso de técnicas distintas a las metodologías relacionales tradicionales, pero puede convertir las transacciones entre cuadrículas de datos en transacciones escalables de una partición única.

Procedimientos recomendados para crear modelos de datos escalables

Los procedimientos recomendados para crear aplicaciones escalables con productos como WebSphere eXtreme Scale incluyen dos categorías: los principios

fundacionales y las sugerencias de implementación. Los principios fundacionales son ideas principales que se deben capturar en el diseño de los propios datos. Una aplicación que no observa estos principios probablemente no realizará bien las escaladas, incluso para sus transacciones principales. Se aplican las sugerencias de implementación para las transacciones problemáticas en una aplicación bien diseñada de otra forma que observa los principios generales para los modelos de datos escalables.

Principios fundacionales

Algunos de los métodos importantes para optimizar la escalabilidad son conceptos o principios básicos que se deben tener en cuenta.

Duplicar en lugar de normalizar

El concepto clave para recordar sobre los productos como WebSphere eXtreme Scale es que se han diseñado para distribuir los datos entre un gran número de sistemas. Si el objetivo es completar la mayoría o todas las transacciones en una única partición, el diseño del modelo de datos debe garantizar que todos los datos que podría necesitar la transacción se encuentran en la partición. La mayoría del tiempo, la única forma de conseguir esto es duplicando los datos.

Por ejemplo, considere una aplicación como un tablón de mensajes. Dos transacciones muy importantes para un tablón de mensajes son mostrar todas las publicaciones de un usuario proporcionado y todas las publicaciones sobre un tema determinado. En primer lugar, considere cómo estas transacciones funcionarían con un modelo de datos normalizado que contiene un registro de usuarios, un registro de temas y un registro de publicaciones que contiene el texto real. Si las publicaciones se particionan con registros de usuarios, la visualización del tema pasa a ser una transacción entre cuadrícula y viceversa. Los temas y los registros no se pueden particionar juntos porque tienen una relación de muchos a muchos.

El mejor método para realizar esta escalada del tablón de mensajes es duplicar las publicaciones, almacenando una copia con el registro de temas y una copia con el registro de usuarios. A continuación, la visualización de las publicaciones de un usuario es una transacción de partición única, la visualización de las publicaciones sobre un tema es una transacción de partición única y la actualización o la supresión de una publicación es una transacción de dos particiones. Estas tres transacciones se escalarán de forma lineal, ya que el número de sistemas de la cuadrícula de datos aumenta.

Escalabilidad en lugar de recursos

El mayor obstáculo para superar cuando se considera eliminar la normalización de los modelos de datos es el impacto que estos modelos tendrían en los recursos. Podría parecer que conservar dos, tres o más copias de algunos datos utiliza demasiados recursos para que sea práctico. Cuando lo confronta con este escenario, recuerde los siguientes hechos: los recursos de hardware son más baratos cada año. En segundo lugar, y más importante, WebSphere eXtreme Scale elimina los costes más ocultos asociados al despliegue de más recursos.

Medir los recursos en términos de coste, en lugar de en términos de sistema como, por ejemplo, megabytes y procesadores. Generalmente, los almacenes de datos que funcionan con datos relacionales normalizados deben estar situados en el mismo sistema. Esta ubicación necesaria

significa que se debe adquirir un único gran sistema empresarial, en lugar de varios sistemas pequeños. Con el hardware de empresa, no es raro que un sistema capaz de completar un millón de transacciones por segundo cueste muchos más que el coste combinado de 10 sistemas capaces de realizar 100.000 transacciones por segundos cada uno.

También existe un coste empresarial en la adición de recursos. Una negocio creciente acaba por agotar la capacidad. Cuando se agota la capacidad, debe concluir mientras se traslada a un sistema mayor y más rápido, o bien crear un segundo entorno de producción al que se puede pasar. De cualquier modo, los costes adicionales vendrán en forma de pérdidas de negocio o en el mantenimiento de casi el doble de la capacidad necesaria durante el periodo de transacción.

Con WebSphere eXtreme Scale, no es necesario concluir la aplicación para añadir capacidad. Si la empresa proyecta que se requiere un 10 por ciento más de capacidad para el próxima año, aumente el número de sistemas de la cuadrícula de datos en un 10 por ciento. Puede aumentar este porcentaje sin tiempo de inactividad de la aplicación y sin adquirir excesiva capacidad.

Evitar transformaciones de datos

Cuando se utiliza WebSphere eXtreme Scale, los datos se deben almacenar en un formato que pueda consumir directamente la lógica empresarial. Desglosar los datos en un formato más primitivo es costoso. La transformación se debe realizar cuando los datos se escriben y cuando los datos se leen. Con las bases de datos relacionales, esta transformación se realiza por necesidad, porque los datos se persisten de forma última en el disco con bastante frecuencia, pero con WebSphere eXtreme Scale, no es necesario que realice estas transformaciones. Para la mayoría de las partes, los datos se almacenan en la memoria y, por lo tanto, se almacenan en el formato exacto que necesita la aplicación.

Observar esta regla simple le ayuda a eliminar la normalización de los datos de acuerdo con el primer principio. El tipo más común de transformación para los datos empresariales es las operaciones JOIN que son necesarias para convertir los datos normalizados en un conjunto de resultados que se ajuste a las necesidades de la aplicación. Almacenar los datos en el formato correcto impide de forma implícita realizar estas operaciones JOIN y genera un modelo de datos no normalizados.

Eliminar consultas no enlazadas

Independientemente de cómo se estructuren los datos, las consultas no enlazadas no se escalan bien. Por ejemplo, no se tiene una transacción que solicite una lista de todos los elementos ordenados por un valor. Esta transacción podría funcionar a la primera, si el número total de elementos es 1000, pero si el número total de elementos llega a 10 millones, la transacción devuelve todos los 10 millones de elementos. Si ejecuta esta transacción, los dos resultados más probables son que la transacción agote el tiempo o que el cliente encuentre un error de memoria agotada.

La mejor opción es alterar la lógica empresarial de forma que sólo se puedan devolver los 10 o 20 primeros elementos. La alteración de la lógica mantiene el tamaño de la transacción gestionable, independientemente de cuántos elementos contenga la memoria caché.

Definir esquema

La principal ventaja de normalizar los datos es que el sistema de la base de datos puede ocuparse de la coherencia de los datos de forma interna. Cuando se elimina la normalización de los datos para la escalabilidad, deja de existir esta gestión automática de la coherencia de los datos. Debe implementar un modelo de datos que puede funcionar en la capa de la aplicación o como plug-in en la cuadrícula de datos distribuida para garantizar la coherencia de los datos.

Considere el ejemplo del tablón de mensajes. Si una transacción elimina una publicación de un tema, se debe eliminar la publicación duplicada del registro de usuarios. Sin un modelo de datos, es posible que un desarrollador escriba el código de la aplicación para eliminar la publicación del tema y olvide eliminar la publicación del registro de usuarios. Sin embargo, si el desarrollador estuviera utilizando un modelo de datos, en lugar de interactuando directamente con la memoria caché, el método removePost en el modelo de datos podría extraer el ID de usuario de la publicación, buscar el registro de usuarios y eliminar la publicación duplicada de forma interna.

De forma alternativa, puede implementar un receptor que se ejecuta en la partición real que detecta el cambio en el tema y ajusta automáticamente el registro de usuarios. Un receptor podría ser beneficioso porque el ajuste del registro de usuarios se podría realizar de forma local si la partición parece tener el registro de usuarios, o incluso si el registro de usuarios está en una partición distinta, la transacción se produce entre los servidores, en lugar de entre el cliente y el servidor. Probablemente, la conexión de red entre los servidores es más rápida que la conexión de red entre el cliente y el servidor.

Impedir la competencia

Se impiden escenarios como tener un contador global. La cuadrícula de datos no se escalará si un único registro se está utilizando un número desproporcionado de veces en comparación con los demás registros. El rendimiento de la cuadrícula de datos se limitará al rendimiento del sistema que aloja un registro determinado.

En estas situaciones, intente dividir el registro para que sea gestionado por partición. Por ejemplo, considere una transacción que devuelve el número total de entradas en la memoria caché distribuida. En lugar de tener cada acceso de la operación insert y remove en un único registro que aumenta, tener un receptor en cada partición rastrea las operaciones insert y remove. Con este rastreo del receptor, las operaciones insert y remove se pueden convertir en operaciones de partición única.

La lectura de un contador pasará a ser una operación entre cuadrículas de datos, pero para la mayoría, ya era ineficaz como operación entre cuadrículas de datos porque su rendimiento estaba ligado al rendimiento del sistema que alojaba el registro.

Sugerencias de implementación

También puede considerar las siguientes sugerencias para conseguir la mejor escalabilidad.

Utilizar los índices de búsqueda inversa

Considere un modelo de datos que ha eliminado la normalización correctamente donde los registros de clientes se particionan basándose en el número del ID de cliente. Este método de particionamiento es la opción

lógica porque casi todas las operaciones empresariales realizadas con el registro de clientes utilizan el número del ID del cliente. Sin embargo, una transacción importante que no utiliza el número del ID del cliente es la transacción de inicio de sesión. Es más común tener nombres de usuario o direcciones de correo electrónico para el inicio de sesión, en lugar de números de ID de cliente.

El enfoque sencillo al escenario de inicio de sesión es utilizar una transacción entre cuadrículas de datos para buscar el registro de cliente. Tal como se ha explicado previamente, este enfoque no realiza escaladas.

La siguiente opción podría ser la partición en el nombre del usuario o el correo electrónico. Esta opción no es práctica porque todas las operaciones basadas en ID de cliente pasan a ser transacciones entre cuadrículas de datos. Asimismo, los clientes del sitio podrían desear cambiar su nombre de usuario o dirección de correo electrónico. Los productos como WebSphere eXtreme Scale necesitan el valor que se utiliza para la partición de datos en constantes de permanencia.

La solución correcta es utilizar un índice de búsqueda inversa. Con WebSphere eXtreme Scale, se puede crear una memoria caché en la misma cuadrícula distribuida que la memoria caché que aloja todos los registros de usuarios. Esta memoria caché es altamente disponible, está particionada y es escalable. Se puede utilizar esta memoria caché para correlacionar un nombre de usuario o dirección de correo electrónico con un ID de cliente. Esta memoria caché convierte el inicio de sesión en una operación de dos particiones, en lugar de una operación entre cuadrícula. Este escenario no es tan bueno como una transacción de partición única, pero el rendimiento se sigue escalando de forma lineal a medida que el número de sistemas aumenta.

Calcular en el momento de la escritura

Los valores calculados comúnmente como promedios o totales pueden resultar caros para generarse, porque normalmente estas operaciones requieren leer un gran número de entradas. Puesto que leer es más comunes que escribir en la mayoría de las aplicaciones, es eficaz calcular estos valores en el momento de escribir y, a continuación, almacenar el resultado en la memoria caché. Esta práctica hace que las operaciones de lectura sean más rápidas y más escalables.

Campos opcionales

Considere un registro de usuarios que incluya una empresa, un lugar y un número de teléfono. Un usuario puede tener todos estos números definidos, o ninguno o alguna combinación de éstos. Si los datos se normalizaron, podría existir una tabla de usuarios y una tabla de números de teléfono. Los números de teléfono para un usuario determinado se podrían encontrar utilizando una operación JOIN entre las dos tablas.

Eliminar la normalización de este registro no requiera la duplicación de datos, porque la mayoría de los usuarios no comparten los números de teléfono. En lugar de esto, debe estar permitido vaciar las ranuras del registro de usuarios. En lugar de tener una tabla de números de teléfono, añada tres atributos a cada registro de usuarios, uno para cada tipo de número de teléfono. Esta adición de atributos elimina la operación JOIN y realiza una búsqueda de número de teléfono para un usuario y una operación de partición única.

Colocación de relaciones de muchos a muchos

Considere una aplicación que rastrea los productos y las tiendas en las que se venden los productos. Un único producto se vende en muchas tiendas y una sola tienda vende muchos productos. Suponga que esta aplicación rastrea 50 tiendas grandes. Cada producto se vende en un máximo de 50 tiendas, con cada tienda que vende miles de productos.

Conservar una lista de tiendas dentro de la entidad de producto (disposición A), en lugar de conservar una lista de productos dentro de cada entidad de tienda (disposición B). Consultando algunas de las transacciones, esta aplicación podría realizar ilustraciones que expliquen por qué la disposición A es más escalable.

En primer lugar, consulte las actualizaciones. Con la disposición A, eliminar un producto del inventario de una tienda bloquea la entidad del producto. Si la cuadrícula de datos aloja 10000 productos, solo será necesario bloquear el 1/10000 de la cuadrícula para realizar la actualización. Con la disposición B, la cuadrícula de datos solo contiene 50 tiendas, por lo que se debe bloquear el 1/50 de la cuadrícula para comletar la actualización. Así pues, aunque ambas disposiciones se podrían considerar operaciones de partición única, la disposición A se escala de forma horizontal de forma más eficaz.

Ahora, si se consideran las lecturas con la disposición A, buscar las tiendas en las que se vende un producto es una transacción de partición única que se escala y es rápida porque la transacción sólo transmite una pequeña cantidad de datos. Con la disposición B, esta transacción pasa a ser una transacción entre cuadrículas de datos porque se debe acceder a cada entidad de tienda para ver si el producto se vende en esa tienda, lo que implica una gran ventaja de rendimiento respecto a la disposición A.

Escalar con datos normalizados

Un uso legítimo de las transacción entre cuadrícula de datos es escalar el proceso de datos. Si una cuadrícula d e datos tiene 5 sistemas y se envía una transacción entre cuadrículas de datos que clasificar unos 100.000 registros e n cada sistema, esa transacción ordenará unos 500.000 registros. Si el sistema más lento de la cuadrícula de datos pueden realizar 10 de estas transacciones por segundo, la cuadrícula de datos puede ordenar unos 5.000.000 registros por segundo. Si los datos de la cuadrícula se doblan, cada sistema realiza una clasificación entre 200.000 registros y cada transacción realiza una clasificación entre 1.000.000 de registros. Este aumento de datos disminuye el rendimiento del sistema más lento a 5 transacciones por segundo, reducendo de esta forma el rendimiento de la cuadrícula de datos a 5 transacciones por segundo. Sin embargo, la cuadrícula de datos ordena unos 5.000.000 registros por segundo.

En este escenario, doblar el número de sistemas permite a cada sistema volver a su carga previa de clasificación entre 100.000 registros, lo que permite al sistema más lento procesar 10 de estas transacciones por segundo. El rendimiento de la cuadrícula de datos continúa siendo el mismo en 10 solicitudes por segundo, pero ahora cada transacción procesa 1.000.000 registros, así que la cuadrícula ha doblado su capacidad de proceso de registros a 10.000.000 por segundo.

Las aplicaciones como por ejemplo un motor de búsqueda que es necesario escalar en términos de proceso de datos para alojar el tamaño creciente de Internet y el rendimiento para acomodar el crecimiento en el número de usuarios, debe crear varias cuadrículas de datos, con un turno circular de las solicitudes entre cuadrículas. Si debe escalar el rendimiento, añada

sistemas y añada otra cuadrícula de datos a las solicitudes de servicio. Si es necesario escalar el proceso de datos, añada más sistemas y mantenga constante el número de cuadrículas de datos.

Visión general de seguridad

WebSphere eXtreme Scale puede proteger el acceso a los datos, incluida la posibilidad de integración con proveedores de datos externos.

Nota: En un almacén de datos no almacenado en memoria caché existente, como una base de datos, probablemente, tendrá características de seguridad incorporadas que podría necesitar para configurar o habilitar de forma activa. No obstante, después de haber almacenado en memoria caché los datos con eXtreme Scale, debe considerar la situación resultante importante de que las características de seguridad del programa de fondo ya no están en vigor. Puede configurar la seguridad de eXtreme Scale en los niveles necesarios de modo que la nueva arquitectura almacenada en memoria caché para los datos también esté protegida.
A continuación, aparece un breve resumen de las características de seguridad de eXtreme Scale. Si desea más información detallada sobre cómo configurar la seguridad, consulte *Guía de administración* y *Guía de programación*.

Conceptos básicos de la seguridad distribuida

La seguridad distribuida de eXtreme Scale se basa en tres conceptos clave:

Autenticación de confianza
: La capacidad de determinar la identidad del solicitante. WebSphere eXtreme Scale da soporte a la autenticación de cliente a servidor y servidor a servidor.

Autorización
: La capacidad de dar permisos para otorgar derechos de acceso al solicitante. WebSphere eXtreme Scale da soporte a distintas autorizaciones para diversas operaciones.

Transporte seguro
: La transmisión segura de datos a través de una red. WebSphere eXtreme Scale soporta los protocolos TLS/SSL (Transport Layer Security/Secure Sockets Layer).

Autenticación

WebSphere eXtreme Scale da soporte a la infraestructura distribuida de cliente-servidor. La infraestructura de seguridad de cliente-servidor existe para proteger el acceso a los servidores de eXtreme Scale. Por ejemplo, cuando el servidor eXtreme Scale requiere una autenticación, el cliente de eXtreme Scale debe proporcionar las credenciales para autenticar el servidor. Estas credenciales pueden ser un par de nombre de usuario y contraseña, un certificado de cliente, un ticket de Kerberos o datos que se presentan en un formato acordado por el cliente y el servidor.

Autorización

Las autorizaciones de WebSphere eXtreme Scale se basan en sujetos y permisos. Puede utilizar JAAS (Java Authentication and Authorization Services) para autoriza el acceso, o puede conectar un método personalizado, como Tivoli Access Manager (TAM), para manejar las autorizaciones. Pueden otorgarse las siguientes autorizaciones a un cliente o grupo:

Autorización de correlaciones
Realizar operaciones de inserción, lectura, actualización o supresión en correlaciones.

Autorización de ObjectGrid
Realizar consultas de objetos o entidades y consultas de secuencias en objetos ObjectGrid.

Autorización de agentes de DataGrid
Permitir que los agentes de DataGrid se desplieguen en un ObjectGrid.

Autorización de correlaciones del lado del servidor
Duplicar una correlación de servidor con el lado del cliente o crear un índice dinámico con la correlación de servidor.

Autorización de administración
Realizar tareas de administración.

Seguridad de transporte

Para proteger la comunicación cliente-servidor, WebSphere eXtreme Scale soporta TLS/SSL. Estos protocolos proporcionan el nivel de seguridad de la capa de transporte con la autenticidad, integridad y confidencialidad para una conexión segura entre un cliente y un servidor de eXtreme Scale.

Seguridad de la cuadrícula

En un entorno seguro, un servidor debe poder comprobar la autenticidad de otro servidor. Para ello WebSphere eXtreme Scale utiliza un mecanismo de serie de clave secreta compartida. Este mecanismo de clave secreta es parecido a una contraseña secreta. Todos los servidores de eXtreme Scale acuerdan una serie secreta compartida. Cuando un servidor se une a la cuadrícula de datos, el servidor se ve obligado a presentar la serie secreta. Si la serie secreta del servidor que se une coincide con una del servidor maestro, este servidor se puede unir a la cuadrícula. De lo contrario, la solicitud se rechaza.

El envío de una serie secreta en texto normal no es seguro. La infraestructura de seguridad de eXtreme Scale proporciona un plug-in SecureTokenManager para permitir al servidor proteger este secreto antes de enviarlo. Puede elegir cómo implementar la operación segura. WebSphere eXtreme Scale proporciona una implementación, en la que se implementa la operación segura para cifrar y firmar la serie secreta.

Seguridad JMX (Java Management Extensions) en una topología de despliegue dinámico

La seguridad de JMX MBean recibe soporte en todas las versiones de eXtreme Scale. Los clientes de MBeans de servidor de catálogo y MBeans de servidor de contenedor pueden autenticarse, y se puede forzar el acceso a operaciones de MBean.

Seguridad de eXtreme Scale local

La seguridad de eXtreme Scale local es distinta del modelo de eXtreme Scale distribuido porque la aplicación crea una instancia y utiliza una instancia de ObjectGrid directamente. La aplicación y las instancias de eXtreme Scale están en la misma JVM (Java Virtual Machine). Puesto que no hay ningún concepto de cliente-servidor en este modelo, no se da soporte a la autenticación. Las

aplicaciones deben gestionar su propia autenticación y, a continuación, pasar el objeto Subject autenticado aeXtreme Scale. Sin embargo, el mecanismo de autorización que se utiliza para el modelo de programación de eXtreme Scale local es el mismo que se ha utilizado para el modelo cliente-servidor.

Configuración y programación

Para obtener más información sobre cómo configurar y programar la seguridad, consulte Integración de la seguridad con proveedores externos y API de seguridad.

Visión general de los servicios de datos REST

El servicio de datos REST de WebSphere eXtreme Scale es un servicio HTTP Java compatible con Microsoft WCF Data Services (anteriormente ADO.NET Data Services) e implementa el Protocolo de datos abierto (OData). Microsoft WCF Data Services es compatible con esta especificación al utilizar Visual Studio 2008 SP1 y .NET Framework 3.5 SP1.

Requisitos de compatibilidad

El servicio de datos REST permite a cualquier cliente HTTP acceder a una cuadrícula de datos. El servicio de datos REST es compatible con el soporte de WCF Data Services que se proporciona con Microsoft .NET Framework 3.5 SP1. Se pueden desarrollar aplicaciones RESTful con las útiles herramientas proporcionadas por Microsoft Visual Studio 2008 SP1. En la figura se proporciona una visión general de cómo interactúa WCF Data Services con clientes y bases de datos.

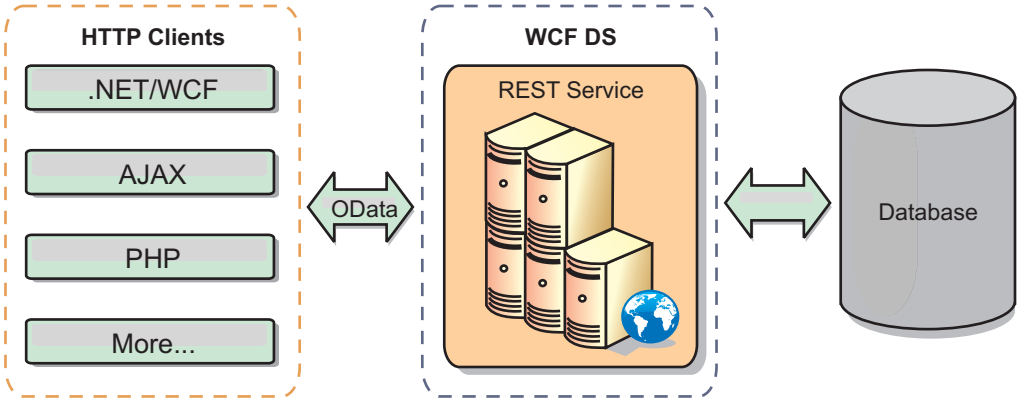


Figura 36. Microsoft WCF Data Services

WebSphere eXtreme Scale incluye una API con muchas funciones para clientes Java. Como se muestra en la figura siguiente, el servicio de datos REST es una pasarela entre clientes HTTP y la cuadrícula de datos de WebSphere eXtreme Scale, comunicando con la cuadrícula mediante un cliente de WebSphere eXtreme Scale. El servicio de datos REST es un servlet Java, que permite despliegues flexibles para plataformas Java Platform, Enterprise Edition (JEE) comunes, como WebSphere Application Server. El servicio de datos REST se comunica con la cuadrícula de datos de WebSphere eXtreme Scale utilizando las API Java de WebSphere eXtreme Scale. Permite los clientes de WCF Data Services o cualquier otro cliente que pueda comunicarse con HTTP y XML.

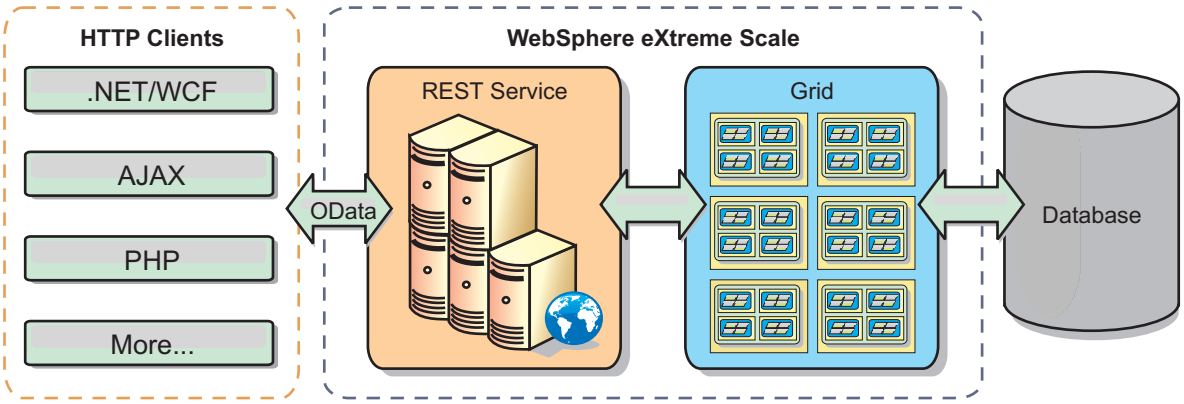


Figura 37. Servicio de datos REST de WebSphere eXtreme Scale

Consulte Configuración de servicios de datos REST, o utilice los enlaces siguientes para obtener más información sobre WCF Data Services.

- Microsoft WCF Data Services Developer Center
- Visión general de ADO.NET Data Services en MSDN
- Libro blanco: Uso de ADO.NET Data Services
- Protocolo de publicación Atom: URI de servicios de datos y ampliaciones de la carga útil
- Formato de archivo de definición de esquema conceptual
- Formato de empaquetado del modelo de datos de entidad para servicios de datos
- Protocolo de datos abierto
- Preguntas frecuentes sobre el protocolo de datos abierto

Características

Esta versión del servicio de datos REST de eXtreme Scale soporta las características siguientes:

- Modelado automático de entidades de API EntityManager de eXtreme Scale como entidades de WCF Data Services que incluye el soporte siguiente:
  - Conversión del tipo de datos Java al tipo de modelo de datos de entidad
  - Soporte para la asociación de entidades
  - Soporte para la asociación de raíces de esquema y claves, necesario para cuadrículas de datos particionadas

  Si desea más información, consulte Modelo de entidad.
- Atom Publish Protocol (AtomPub o APP) XML y formato de carga útil de datos JavaScript Object Notation (JSON).
- Operaciones de creación, lectura, actualización y supresión (CRUD) utilizando los respectivos métodos de solicitud HTTP: POST, GET, PUT y DELETE. Además, la ampliación de Microsoft: MERGE está soportada.
- Consultas simples, con filtros
- Solicitudes de recuperación por lotes y conjuntos de cambios
- Soporte de cuadrícula de datos particionada para alta disponibilidad
- Interoperatividad con clientes de la API EntityManager de eXtreme Scale
- Soporte para servidores web JEE estándar
- Simultaneidad optimista

- Autorización y autenticación de usuarios entre el servicio de datos REST y la cuadrícula de datos de eXtreme Scale

Problemas y limitaciones conocidos

- Las solicitudes a través de túnel no están soportadas.

Capítulo 2. Planificación

Antes de instalar WebSphere eXtreme Scale y desplegar las aplicaciones de cuadrícula de datos, debe decidir sobre la topología de almacenamiento en memoria caché, completar la planificación de capacidad, revisar los requisitos de hardware y software, valores de red y ajuste, etc. ambién puede utilizar la lista de comprobación operacional para asegurarse de que el entorno está preparado para tener una aplicación desplegada.

Para obtener una descripción de los métodos recomendados que puede utilizar al diseñar las aplicaciones WebSphere eXtreme Scale, lea el artículo siguiente en developerWorks: Principles and best practices for building high performing and highly resilient WebSphere eXtreme Scale applications (Principios y métodos recomendados para crear aplicaciones de WebSphere eXtreme Scale muy flexibles y de alto rendimiento).

Planificación de la topología

Con WebSphere eXtreme Scale, la arquitectura puede utilizar el almacenamiento en memoria caché de datos en memoria local o el almacenamiento en memoria caché de datos de cliente-servidor distribuido. La arquitectura puede tener distintas relaciones con las bases de datos. También puede configurar la topología para que abarque varios centros de datos.

Para poder funcionar, WebSphere eXtreme Scale necesita una mínima infraestructura adicional. La infraestructura se compone de scripts que instalan, inician y detienen una aplicación Java Platform, Enterprise Edition en un servidor. Los datos colocados en memoria caché se almacenan en servidores de contenedor, y los clientes se conectan de forma remota al servidor.

Entornos en memoria

Cuando realiza un despliegue en un entorno local en memoria, WebSphere eXtreme Scale se ejecuta en una única Máquina virtual Java y no se replica. Para configurar un entorno local puede utilizar un archivo XML de ObjectGrid o las API de ObjectGrid.

Entornos distribuidos

Cuando realiza un despliegue en un entorno distribuido, WebSphere eXtreme Scale se ejecuta en un conjunto de Máquinas virtuales Java, aumentando el rendimiento, disponibilidad y escalabilidad. Con esta configuración, puede utilizar el particionamiento y la réplica de datos. También puede añadir servidores adicionales sin reiniciar los servidores eXtreme Scale existentes. Igual que en el entorno local, en el entorno distribuido se necesita un archivo XML ObjectGrid, o una configuración equivalente mediante programa. Debe también proporcionar un archivo XML de política de despliegue con detalles de configuración

Puede crear despliegues sencillos o grandes despliegues con terabytes en los que son necesarios miles de servidores.

Almacenamiento local de memoria caché en memoria

En el caso más sencillo, WebSphere eXtreme Scale se puede utilizar como una memoria caché de cuadrícula de datos en memoria local (no distribuida). El caso local beneficia especialmente a las aplicaciones de simultaneidad alta donde varias hebras necesitan acceder y modificar los datos transitorios. Los datos que se mantienen en una cuadrícula de datos local se pueden indexar y recuperar mediante consultas. Las consultas le ayudan a utilizar conjuntos de datos en memoria grandes. El soporte proporcionado con Máquina virtual Java (JVM), aunque está listo para su uso, tiene una estructura de datos limitada.

La topología de la memoria caché en memoria local para WebSphere eXtreme Scale se utiliza para proporcionar un acceso coherente y transaccional a los datos temporales de una única máquina virtual Java.

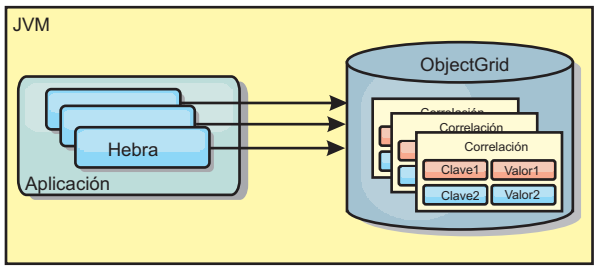


Figura 38. Escenario de memoria caché en memoria local

Ventajas

- Fácil configuración: se puede crear un ObjectGrid a través de un programa o de forma declarativa con el archivo XML descriptor de ObjectGrid o con otras infraestructuras como, por ejemplo, Spring.
- Rápido: cada BackingMap puede adaptarse de forma independiente de modo que la utilización de la memoria y la simultaneidad sean óptimas.
- Es ideal para las topologías de máquina virtual Java única con conjuntos de datos pequeños o para almacenar en memoria caché los datos de acceso frecuente.
- Es transaccional. Las actualizaciones de BackingMap se pueden agrupar en una única unidad de trabajo y se pueden integrar como último participante en transacciones de 2 fases como, por ejemplo, transacciones JTA (Java Transaction Architecture).

Desventajas

- No es tolerante a errores.
- Los datos no se replican. Las memorias caché en memoria son la mejor solución para los datos de referencia de sólo lectura.
- No es escalable. La cantidad de memoria necesaria para la base de datos podría desbordar la máquina virtual Java.
- Se producen problemas al añadir máquinas virtuales Java:
  - Los datos no se pueden particionar fácilmente.
  - Se debe replicar manualmente el estado entre las máquinas virtuales Java o cada instancia podría tener distintas versiones de los mismos datos.
  - La operación de invalidación es muy costosa.

– Cada memoria caché se debe calentar de forma independientemente. El calentamiento es el periodo de carga de un conjunto de datos, de forma que la memoria caché se rellena con datos válidos.

Cuándo se debe utilizar

La topología de despliegue de la memoria caché en memoria local sólo se debe utilizar cuando la cantidad de datos que se deben almacenar en memoria caché es pequeña (cabe en una única máquina virtual Java) y es relativamente estable. Los datos obsoletos deben tolerarse con este acercamiento. El uso de desalojadores para mantener en la memoria caché los datos usados con más frecuencia o los más recientes puede ayudar a mantener pequeño el tamaño de la memoria caché y a aumentar la relevancia de los datos.

Memoria caché local replicada de igual

Debe asegurarse de que la memoria caché esté sincronizada si existen varios procesos con instancias de memoria caché independientes. Para asegurarse de que las instancias de memoria caché están sincronizadas, habilite una memoria caché replicada por un igual con JMS (Java Message Service).

WebSphere eXtreme Scale incluye dos plug-ins que propagan automáticamente los cambios de las transacciones entre instancias de ObjectGrid de un igual. El plug-in JMSObjectGridEventListener propaga automáticamente los cambios de eXtreme Scale mediante JMS.

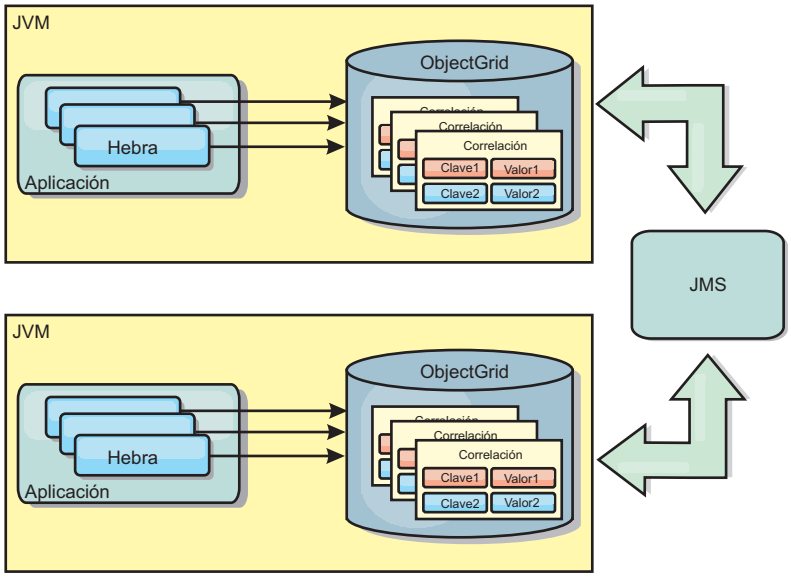


Figura 39. La memoria caché duplicada por un igual con los cambios que se propagan con JMS

Si ejecuta un entorno WebSphere Application Server, el plug-in TranPropListener también está disponible. El plug-in TranPropListener utiliza el gestor de alta disponibilidad (HA) para propagar los cambios a cada instancia de memoria caché de igual.

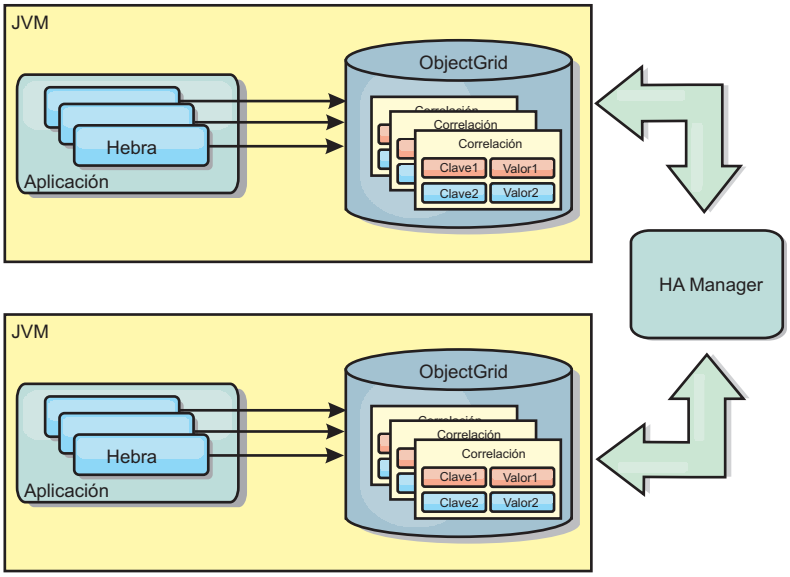


Figura 40. La memoria caché duplicada por un igual con los cambios propagados con el High Availability Manager.

Ventajas

- Los datos son más válidos porque se actualizan con más frecuencia.
- Con el plug-in TranPropListener, igual que el entorno local, eXtreme Scale se puede crear a través de programa o de forma declarativa con el archivo XML de descriptor de despliegue de eXtreme Scale o con otras infraestructuras como, por ejemplo, Spring. La integración con el High Availability Manager se realiza de forma automática.
- Cada BackingMap se puede ajustar independientemente para obtener un uso y una simultaneidad óptimos de la memoria.
- Las actualizaciones de BackingMap se pueden agrupar en una única unidad de trabajo y se pueden integrar como último participante en transacciones de 2 fases como, por ejemplo, transacciones JTA (Java Transaction Architecture).
- Ideal para topologías de pocas JVM con un conjunto de datos razonablemente pequeño o para almacenar en memoria caché datos de acceso frecuente.
- Los cambios en eXtreme Scale se duplican en todas las instancias de eXtreme Scale de igual. Los cambios son coherentes mientras se utilice una suscripción duradera.

Desventajas

- La configuración y el mantenimiento de JMSObjectGridEventListener pueden ser complejos. eXtreme Scale puede crearse mediante programación o de forma declarativa con el archivo XML de descriptor de despliegue de eXtreme Scale o con otras infraestructuras como Spring.
- No es escalable: el volumen de memoria que requiere la base de datos puede desbordar la JVM.
- Funciona de forma incorrecta cuando se añade Máquinas virtuales Java:
  - Los datos no se pueden particionar fácilmente.
  - La operación de invalidación es muy costosa.
  - Cada memoria caché debe calentarse de manera independiente.

Cuándo se debe utilizar

Utilice topología de despliegue solo cuando la cantidad de datos que se deben almacenar en memoria caché sea pequeña, pueda caber en una única JVM y sea relativamente estable.

Memoria caché incorporada

Las cuadrículas de WebSphere eXtreme Scale pueden ejecutarse en procesos existentes como servidores eXtreme Scale incorporados o bien puede gestionarlas como procesos externos.

Las cuadrículas incorporadas son útiles cuando se ejecutan en un servidor de aplicaciones como, por ejemplo, WebSphere Application Server. Puede iniciar los servidores eXtreme Scale que no están incorporados utilizando los scripts de la línea de mandatos y ejecutarlos en un proceso Java.

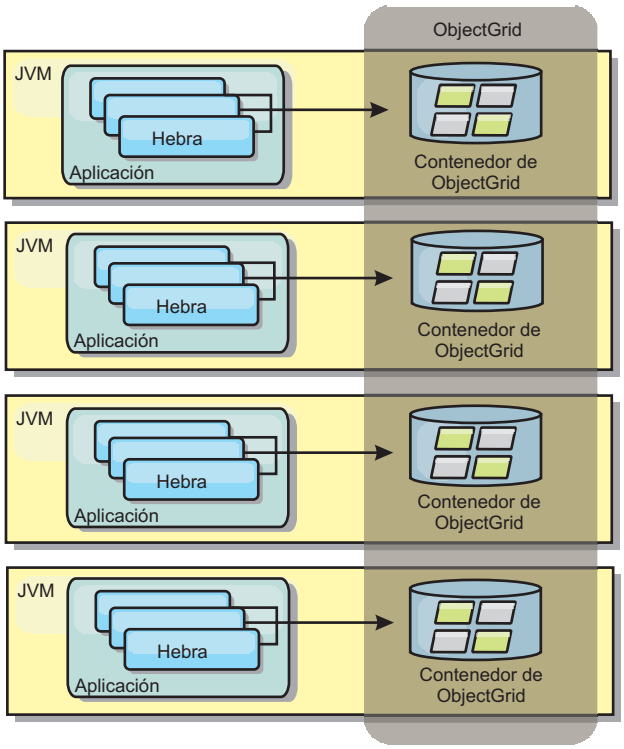


Figura 41. Memoria caché incorporada

Ventajas

- Administración simplificada ya que hay menos procesos que deban gestionarse.
- Despliegue de aplicaciones simplificado ya que la cuadrícula utiliza el cargador de clases de la aplicación cliente.
- Admite particionamiento y alta disponibilidad.

Desventajas

- Aumenta el uso de la memoria en procesos de cliente ya que todos los datos se colocan en el proceso.
- Aumenta el uso de la CPU para dar servicio a las solicitudes de los clientes.

- Es más difícil manejar las actualizaciones de las aplicaciones ya que los clientes utilizan los mismos archivos JAR (Java Archive) de aplicación que los servidores.
- Menos flexible. Escalar clientes y servidores de cuadrícula no puede aumentar a la misma velocidad. Si los servidores se definen externamente, puede tener más flexibilidad al gestionar el número de procesos.

Cuándo se debe utilizar

Utilice cuadrículas incorporadas cuando haya suficiente memoria libre en el proceso de cliente para datos de cuadrícula y posibles datos de sustitución por anomalía.

Para obtener más información, consulte el tema sobre la habilitación del mecanismo de invalidación de cliente en la*Guía de administración*.

Memoria caché distribuida

WebSphere eXtreme Scale se usa con más frecuencia como una memoria caché compartida, para proporcionar acceso transaccional a los datos en varios componentes donde, de lo contrario, se utilizará una base de datos tradicional. La memoria caché compartida elimina la necesidad de configurar una base de datos.

Coherencia de la memoria caché

La memoria caché es coherente porque todos los clientes ven los mismos datos en la memoria caché. Cada dato se almacena exactamente en un servidor de la memoria caché, lo que evita tener copias innecesarias que podrían contener posiblemente distintas versiones de los datos. Una memoria caché coherente también puede contener más datos a medida que se añadan más servidores a la cuadrícula de datos, y se amplía de forma lineal a medida que crece el tamaño de la cuadrícula. Puesto que los clientes acceden a los datos desde esta cuadrícula de datos con llamadas a procedimiento remotas, también se conoce como memoria caché remota, o memoria caché lejana). A través de la partición de datos, cada proceso contiene un subconjunto exclusivo del conjunto de datos total. Las cuadrículas de datos más grandes pueden contener más datos y dar servicio a más solicitudes de esos datos. La coherencia también elimina la necesidad de pasar datos de invalidación por la cuadrícula de datos porque no hay datos obsoletos. La memoria caché coherente sólo contiene la copia más reciente de cada dato.

Si ejecuta un entorno WebSphere Application Server, el plug-in TranPropListener también está disponible. El plug-in TranPropListener utiliza el componente de alta disponibilidad (HA Manager) de WebSphere Application Server para propagar los cambios en cada instancia de memoria caché de ObjectGrid de igual.

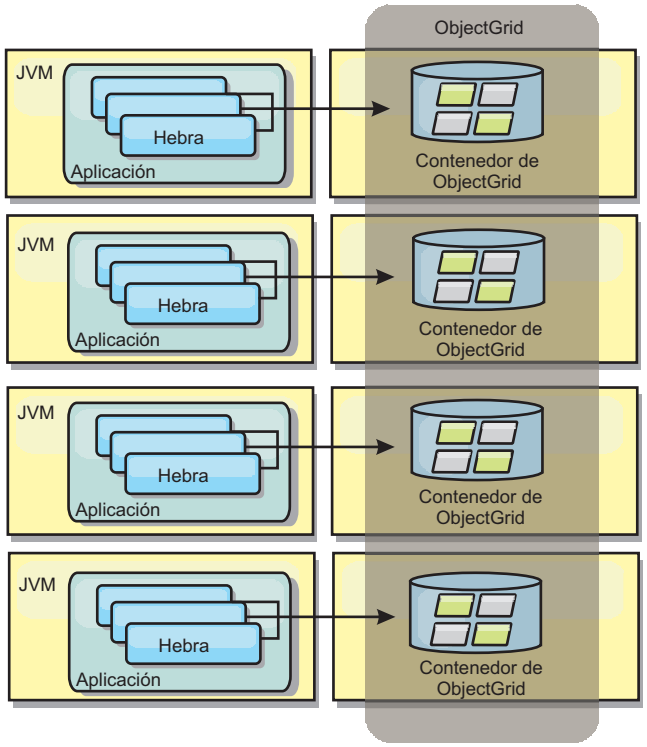


Figura 42. Memoria caché distribuida

Memoria caché cercana

De forma opcional, los clientes pueden tener una memoria caché local en línea cuando se utiliza eXtreme Scale en una topología distribuida. Esta memoria caché opcional se llama memoria caché cercana, es un ObjectGrid independiente en cada cliente, que sirve como memoria caché para la memoria caché remota del lado del servidor. La memoria caché cercana se habilita de manera predeterminada al configurar el bloqueo como optimista o ninguno, y no puede utilizarse si se configura como pesimista.

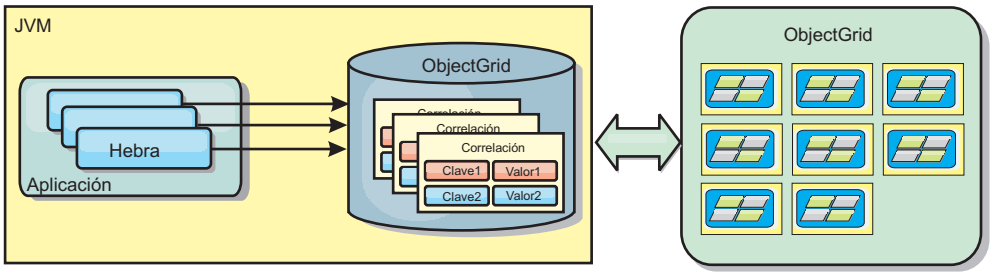


Figura 43. Memoria caché cercana

Una memoria caché cercana es muy rápida porque proporciona un acceso en memoria a un subconjunto de todos los conjuntos de datos almacenados en memoria caché que se almacenan de forma remota en los servidores eXtreme Scale. La memoria caché cercana no está particionada y contiene datos de cualquiera de las particiones eXtreme Scale remotas. WebSphere eXtreme Scale puede tener hasta tres niveles de memoria caché del modo siguiente.

1. La memoria caché de nivel de transacción contiene todos los cambios de una única transacción. La memoria caché de transacción contiene una copia de

trabajo de los datos hasta que la transacción se confirma. Cuando una transacción de cliente solicita datos de un objeto ObjectMap, primero se comprueba la transacción.

2. La memoria caché cercana en el nivel de cliente contiene un subconjunto de datos del nivel de servidor. Cuando el nivel de transacción no tiene los datos, los datos se captan de la capa de cliente, si están disponibles, y se insertan en la memoria caché de transacción
3. La cuadrícula de datos del nivel del servidor contiene la mayoría de los datos y se comparte entre todos los clientes. El nivel de servidor puede particionarse, lo que permite almacenar en memoria caché un gran volumen de datos. Cuando la memoria caché cercana de cliente no tiene los datos, éstos se captan del nivel de servidor y se insertan en la memoria caché de cliente. El nivel de servidor también tiene un plug-in Loader. Si la cuadrícula no tiene los datos solicitados, se invoca el Loader y los datos resultantes se insertan del almacén de datos de proceso de fondo en la cuadrícula.

Para inhabilitar la memoria caché cercana, establezca el atributo numberOfBuckets en 0 en la configuración de descriptor de eXtreme Scale de alteración temporal del cliente. Consulte el tema sobre el bloqueo de entrada de correlación para ver detalles sobre las estrategias de bloqueo de eXtreme Scale. La memoria caché cercana también se puede configurar para tener una política de desalojo separada y distintos plug-ins mediante una configuración de descriptor de eXtreme Scale de alteración temporal del cliente.

Ventaja

- Un tiempo de respuesta rápido porque todos los accesos a los datos son locales. Buscando los datos en la memoria caché cercana primero se guarda un recorrido a la cuadrícula de los servidores, por lo hace que incluso los datos remotos se puedan acceder de forma local.

Desventajas

- Aumenta la duración de los datos obsoletos debido a que la memoria caché cercana en cada nivel puede no estar sincronizada con los datos actuales de la cuadrícula de datos.
- Se basa en un desalojador para invalidar los datos a fin de evitar quedarse sin memoria.

Cuándo se debe utilizar

Debe usarse cuando el tiempo de respuesta sea importante y puedan tolerarse los datos obsoletos.

Integración de base de datos: almacenamiento en memoria caché de grabación diferida, en línea y complementaria

WebSphere eXtreme Scale se utiliza para atender una base de datos tradicional y eliminar la actividad de lectura que normalmente se envía a la base de datos. Puede utilizarse una memoria caché coherente con una aplicación mediante el uso directo o indirecto de un correlacionador de objetos relacionales. La memoria caché coherente puede después descargar de lecturas la base de datos o el programa de fondo. En un escenario ligeramente más complejo, como por ejemplo un acceso transaccional a un conjunto de datos donde sólo algunos de los datos necesitan garantías de persistencia tradicional, puede usarse el filtrado para descargar incluso transacciones de grabación.

Puede configurar WebSphere eXtreme Scale para que funcione como un espacio de proceso de base de datos en memoria muy flexible. No obstante, WebSphere eXtreme Scale no es un correlacionador de objetos relacionales (ORM). No sabe de dónde proceden los datos de la cuadrícula de datos. Una aplicación o un ORM puede colocar datos en un servidor eXtreme Scale. Es responsabilidad del origen de datos garantiza que son coherentes con la base de datos de la que proceden los datos. Esto significa que eXtreme Scale no puede invalidar los datos extraídos de una base de datos automáticamente. La aplicación o el correlacionador debe proporcionar esta función y gestionar los datos almacenados en eXtreme Scale.

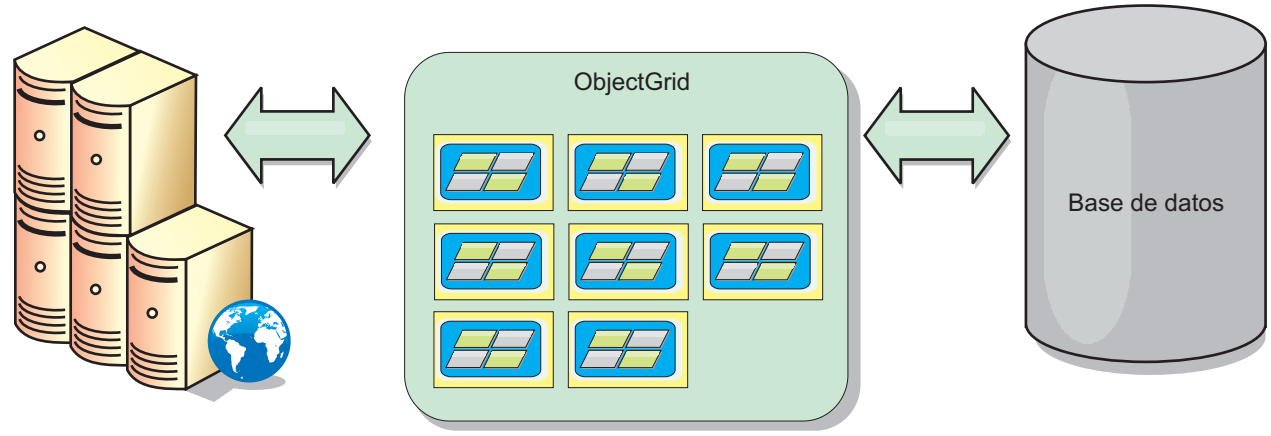


Figura 44. ObjectGrid como un almacenamiento intermedio de base de datos

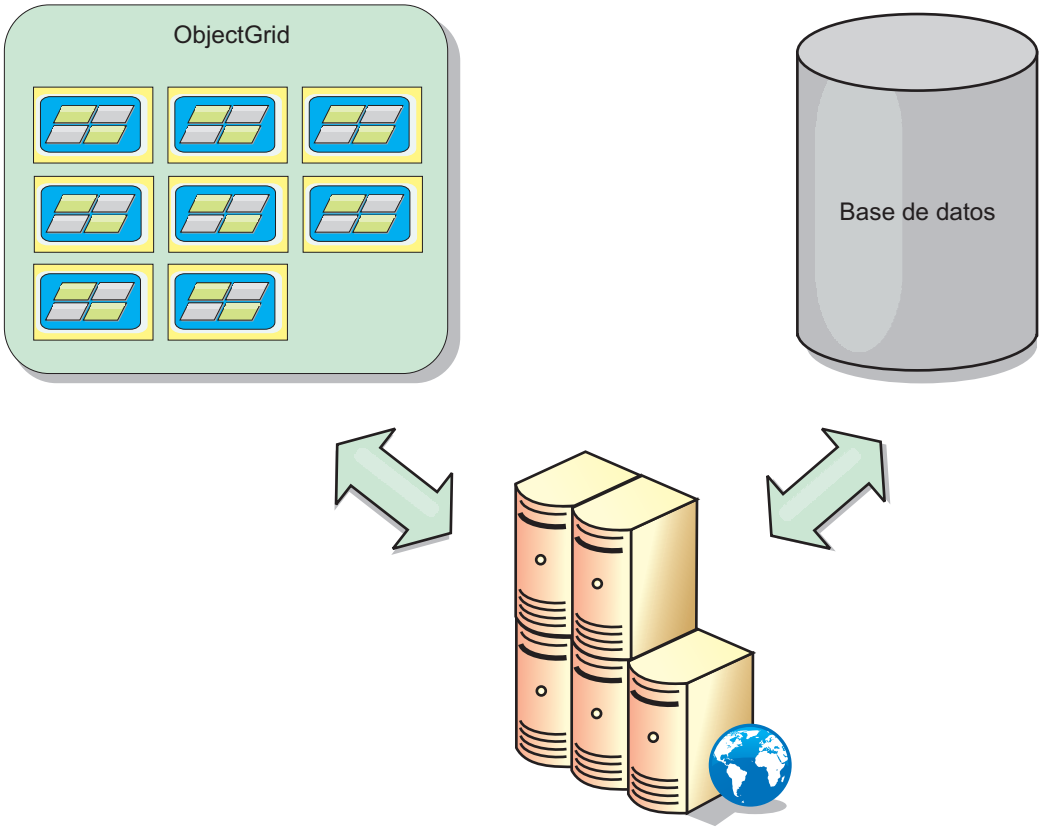


Figura 45. ObjectGrid como una memoria caché secundaria

Memoria caché escasa y completa

WebSphere eXtreme Scale puede utilizarse como una memoria caché escasa o una memoria caché completa. Una memoria caché escasa sólo mantiene un subconjunto de los datos totales, mientras que una memoria caché completa conserva todos los datos y se puede llenar de forma poca activa, conforme se requieran los datos. A las memorias caché escasas normalmente se accede utilizando claves (en lugar de índices o consultas) puesto que los datos sólo están parcialmente disponibles.

memoria caché escasa

Cuando una clave no está presente en una memoria caché escasa, o los datos no están disponibles y se produce una falta de coincidencia de memoria caché, se invoca el siguiente nivel. Los datos se captan, desde una base de datos, por ejemplo, y se insertan en el nivel de la memoria caché de cuadrícula de datos. Si utiliza una consulta o un índice, sólo se accede a los valores cargados actualmente y las solicitudes no se remiten a los demás niveles.

Memoria caché completa

Una memoria caché completa contiene todos los datos necesarios y se puede acceder a la misma utilizando atributos que no son de clave con índices o consultas. Una memoria caché completa se precarga con datos de la base de atos antes de que la aplicación intente acceder a los datos. Una memoria caché completa puede funcionar como una sustitución de base de datos después de que se carguen los datos. Puesto que están disponibles todos los datos, las consultas y los índices se pueden utilizar para encontrar y agregar datos.

Memoria caché complementaria

Cuando se utiliza WebSphere eXtreme Scale como memoria caché complementaria, se utiliza el programa de fondo con la cuadrícula de datos.

Memoria caché complementaria

Puede configurar el producto como una memoria caché complementaria para la capa de acceso a datos de una aplicación. En este escenario, WebSphere eXtreme Scale se utiliza para almacenar temporalmente objetos que normalmente se recuperarían de una base de datos de programa de fondo. Las aplicaciones comprueban si la cuadrícula de datos contiene los datos. Si los datos están en la cuadrícula de datos, los datos se devuelven al emisor. Si los datos no existen, los datos se recuperan de la base de datos de fondo. A continuación, los datos se insertan en la cuadrícula de datos de forma que la siguiente solicitud pueda utilizar la copia almacenada en memoria caché. El diagrama siguiente muestra cómo se puede utilizar WebSphere eXtreme Scale como una memoria caché complementaria con una capa de acceso a datos arbitrarios como por ejemplo OpenJPA o Hibernate.

Plug-ins de memoria caché para Hibernate y OpenJPA

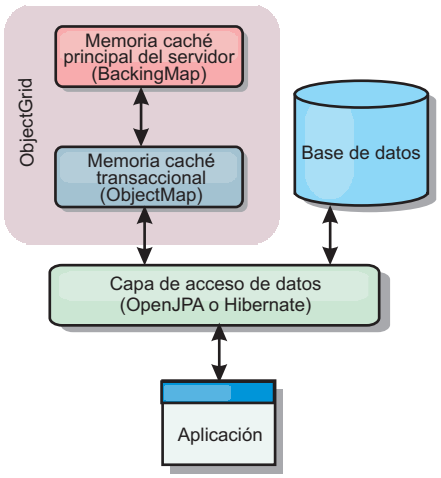


Figura 46. Memoria caché complementaria

Los plug-ins de memoria caché para OpenJPA e Hibernate se incluyen en WebSphere eXtreme Scale, de forma que puede utilizar el producto como una memoria caché complementaria automática. El uso de WebSphere eXtreme Scale como un proveedor de memoria caché aumenta el rendimiento cuando se leen y consultan datos y reduce la carga de la base de datos. WebSphere eXtreme Scale presenta algunas ventajas sobre las implementaciones de memoria caché incorporada ya que la memoria caché se replica automáticamente entre procesos. Cuando un cliente almacena en memoria caché un valor, todos los demás clientes pueden utilizar el valor almacenado en la memoria.

Memoria caché en línea

Puede configurar almacenamiento en memoria caché en línea para un programa de fondo de base de datos o como una memoria complementaria para una base de datos. El almacenamiento en memoria caché en línea utiliza eXtreme Scale como el medio principal para interactuar con los datos. Cuando se utiliza eXtreme Scale como una memoria caché en línea, la aplicación interactúa con el programa de fondo mediante un plug-in Loader.

Memoria caché en línea

Cuando se utiliza como una memoria caché en línea, WebSphere eXtreme Scale interactúa con el programa de fondo utilizando un plug-in Loader. Este escenario puede simplificar el acceso a datos porque las aplicaciones pueden acceder a las API eXtreme Scale directamente. Se da soporte a distintos escenarios de almacenamiento en memoria caché en eXtreme Scale para garantizar que los datos de la memoria caché y los datos del programa de fondo estarán sincronizados. El diagrama siguiente ilustra cómo una memoria caché en línea interactúa con la aplicación y el programa de fondo.

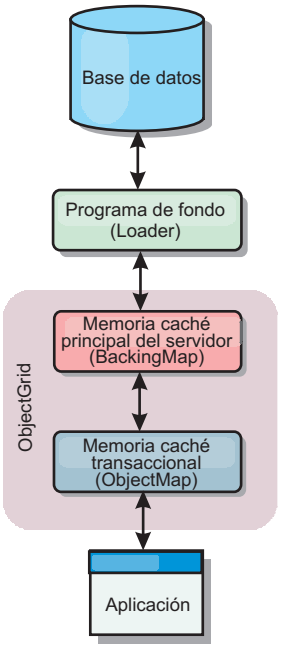


Figura 47. Memoria caché en línea

La opción de memoria caché en línea simplifica el acceso de datos, porque permite a las aplicaciones acceder a las API de eXtreme Scale directamente. WebSphere eXtreme Scale soporta varios escenarios de memoria caché en línea, del modo siguiente.

- Lectura directa
- Grabación directa
- Grabación diferida

Caso de ejemplo de almacenamiento en memoria caché de lectura directa

Una memoria caché de lectura directa es una memoria caché escasa que carga de forma poco activa entradas de datos por clave cuando se solicitan. Esto se lleva a cabo sin que el solicitante sepa cómo se llenan las entradas. Si los datos no se pueden encontrar en la memoria caché de eXtreme Scale, eXtreme Scale recuperará los datos que faltan del plug-in Loader, que carga los datos de la base de datos de programa de fondo y los inserta en la memoria caché. Las solicitudes subsiguientes para la misma clave de datos se encontrarán en la memoria caché hasta que se elimina, anula o desaloja.

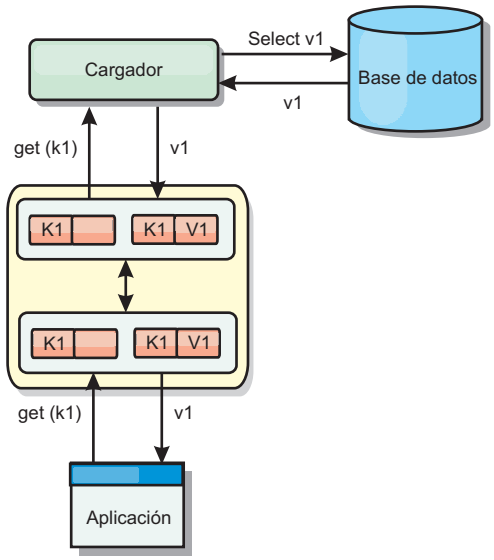


Figura 48. Almacenamiento en memoria caché de lectura directa

Caso de ejemplo de almacenamiento en memoria caché de grabación directa

En una memoria caché de grabación directa, cada grabación en la memoria caché graba de forma síncrona en la base de datos mediante el cargador. Este método proporciona coherencia con el programa de fondo, pero reduce el rendimiento de grabación porque la operación de la base de datos es síncrona. Como que la memoria caché y la base de datos están actualizadas, las lecturas subsiguientes para los mismos datos se encontrarán en la memoria caché, evitando la llamada a la base de datos. Una memoria caché de grabación directa suele utilizarse junto con una memoria caché de lectura directa.

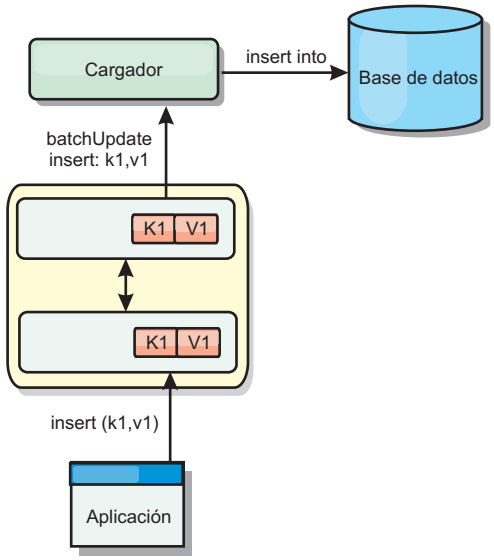


Figura 49. Almacenamiento en memoria caché de grabación directa

Caso de ejemplo de almacenamiento en memoria caché de grabación anticipada

La sincronización de base de datos se puede mejorar grabando los cambios de forma asíncrona. Esto se conoce como memoria caché de grabación diferida o de grabación aplazada. En su lugar, los cambios que normalmente se grabarían de forma síncrona en el cargador se colocarán en el almacenamiento intermedio de eXtreme Scale y se grabarán en la base de datos utilizando una hebra de subordinada. El rendimiento de grabación se mejora de forma significativa porque la operación de la base de datos se elimina de la transacción del cliente y se pueden comprimir las grabaciones de la base de datos.

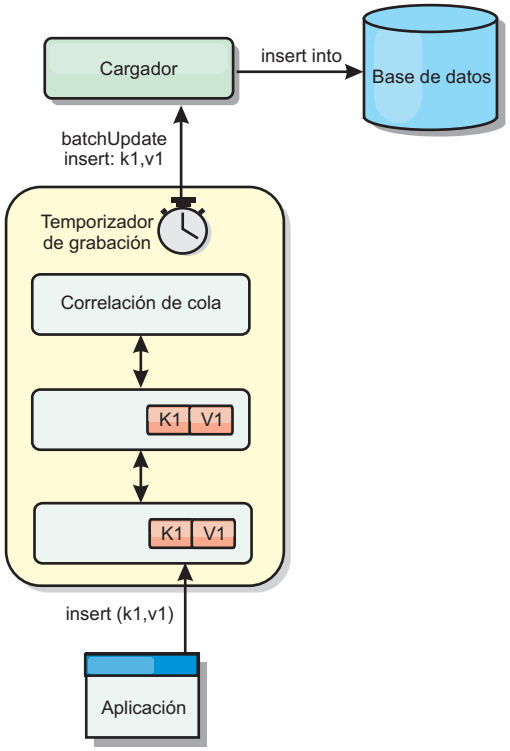


Figura 50. Almacenamiento en memoria caché de grabación diferida

Almacenamiento en memoria caché de grabación diferida

Puede utilizar el almacenamiento en la memoria caché de grabación diferida para reducir la sobrecarga que se produce al actualizar una base de datos utilizada como programa de fondo.

Visión general del almacenamiento en memoria caché con grabación diferida

El almacenamiento en memoria caché de grabación diferida pone en cola de forma asíncrona actualizaciones del plug-in de cargador (Loader). Puede mejorar el rendimiento mediante la desconexión de actualizaciones, inserciones y eliminaciones de una correlación, la sobrecarga de la actualización de la base de datos de programa de fondo. La actualización asíncrona se realiza después de un retardo basado en la hora (por ejemplo, cinco minutos) o un retardo basado en entradas (1000 entradas).

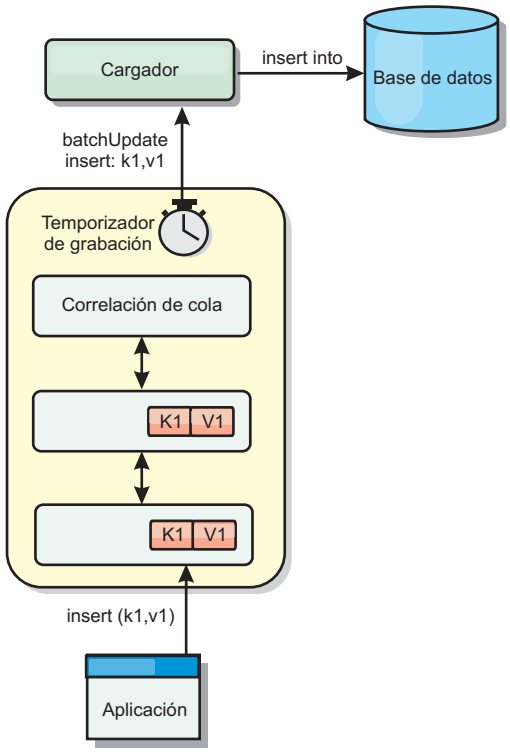


Figura 51. Almacenamiento en memoria caché de grabación diferida

La configuración de la grabación diferida en BackingMap crea una hebra entre el cargador y la correlación. El cargador delega las solicitudes de datos a través de la hebra de acuerdo con los valores de configuración del método BackingMap.setWriteBehind. Cuando una transacción de eXtreme Scale inserta, actualiza o elimina una entrada de una correlación, se crea un objeto LogElement para cada uno de estos registros. Estos elementos se envían al cargador de grabación diferida y se ponen en cola en un objeto ObjectMap especial llamado correlación de cola. Cada correlación de respaldo con el valor de grabación diferida habilitado tiene sus propias correlaciones de cola. Una hebra de grabación diferida elimina periódicamente los datos en cola de las correlaciones de cola y los envía al cargador de programa de fondo real.

El cargador de grabación diferida sólo envía los tipos de inserción, actualización y eliminación de objetos LogElement al cargador real. Todos los demás tipos de objetos LogElement, por ejemplo el tipo EVICT, se pasan por alto.

El soporte de grabación diferida *es* una ampliación del plug-in Loader, que puede utilizar para integrar eXtreme Scale con la base de datos. Por ejemplo, consulte la información del apartado Configuración de cargadores JPA sobre cómo configurar un cargador JPA.

Ventajas

La habilitación del soporte de grabación diferida tiene las ventajas siguientes:

- **Aislamiento de anomalía de programa de fondo:** el almacenamiento de grabación diferida proporciona una capa de aislamiento de las anomalías de programa de fondo. Cuando la base de datos de programa de fondo falla, las actualizaciones se ponen en cola en la correlación de cola. Las aplicaciones

pueden continuar con las transacciones a eXtreme Scale. Cuando se recupera el programa de fondo, los datos de la correlación de cola se envían al programa de fondo.

- **Carga reducida de programa de fondo** el cargador de grabación diferida fusiona las actualizaciones según una clave, de forma que sólo existe una actualización fusionada por clave en la correlación de cola. Este procedimiento reduce el número de actualizaciones en la base de datos de programa de fondo.
- **Rendimiento mejorado de transacciones:** los tiempos individuales de las transacciones de eXtreme Scale se reducen porque la transacción no necesita esperar a que los datos se sincronicen con el programa de fondo.

Consideraciones sobre el diseño de aplicaciones

Habilitar el soporte de grabación diferida es sencillo, pero diseñar una aplicación que funcione con el soporte de grabación diferida requiere un cuidado especial. Sin el soporte de grabación diferida, la transacción ObjectGrid encierra la transacción del programa de fondo. La transacción ObjectGrid se inicia antes de que se inicie la transacción de programa de fondo, pero termina después de que termine la transacción de programa de fondo.

Con el soporte de grabación diferida habilitado, la transacción ObjectGrid finaliza antes de que se inicie la transacción de programa de fondo. La transacción ObjectGrid y la transacción del programa de fondo se desacoplan.

Restricciones de la integridad referencial

Cada correlación de respaldo que se configura con soporte de grabación diferida tiene su propia hebra de grabación diferida que empuja los datos al programa de fondo. Por lo tanto, los datos que se actualizan en correlaciones diferentes de una transacción ObjectGrid se actualizan en el programa de fondo en diferentes transacciones de programa de fondo. Por ejemplo, la transacción T1 actualiza la clave key1 en la correlación Map1 y la clave key2 en la correlación Map2. La actualización de key1 en la correlación Map1 se actualiza en el programa de fondo en una transacción de programa de fondo, y la clave key2 actualizada en la correlación Map2 se actualiza en el programa de fondo en otra transacción de programa de fondo mediante distintas hebras de grabación diferida. Si los datos almacenados en Map1 y Map2 tienen relaciones, como restricciones de clave foránea en el programa de fondo, puede que se produzca un error en las actualizaciones.

Al diseñar las restricciones de la integridad referencial en la base de datos de programa de fondo, asegúrese de que se permiten las actualizaciones que no funcionan.

Comportamiento de bloqueo de correlaciones de cola

Otra diferencia principal en el comportamiento de las transacciones es el comportamiento de bloqueo. ObjectGrid admite tres estrategias de bloqueo distintas: pesimista (PESSIMISTIC), optimista (OPTIMISTIC) y ninguno (NONE). Las correlaciones de cola de grabación diferida utilizan la estrategia de bloqueo pesimista independientemente de la estrategia de bloqueo configurada en el mapa de respaldo. Existen dos tipos diferentes de operaciones que adquieren un bloqueo en la correlación de cola:

- Cuando se confirma una transacción ObjectGrid, o se produce un vaciado (vaciado de correlación o vaciado de sesión), la transacción lee la clave de la correlación de cola y coloca un bloqueo S en la clave.
- Cuando se confirma una transacción ObjectGrid, la transacción intenta actualizar el bloqueo S a un bloqueo X en la clave.

Debido a este comportamiento de correlación de colas adicional, puede ver algunas diferencias en el comportamiento del bloqueo.

- Si la correlación de usuarios está configurada como estrategia de bloqueo pesimista (PESSIMISTIC), no hay mucha diferencia de comportamiento en el bloqueo. Cada vez que se llama a una operación de desecho o confirmación, se coloca un bloqueo S en la misma clave de la misma correlación de colas. Durante la confirmación, no sólo se adquiere un bloqueo X para la clave en la correlación de usuarios, sino que además de adquiere para la clave en la correlación de colas.
- Si la correlación de usuarios está configurada como estrategia de bloqueo optimista (OPTIMISTIC) o ninguna (NONE), la transacción de usuario seguirá el patrón de estrategia de bloqueo pesimista (PESSIMISTIC). Cada vez que se llama a una operación de desecho o confirmación, se adquiere un bloqueo S en la misma clave de la misma correlación de colas. Durante la confirmación se adquiere un bloqueo X para la clave en la correlación de colas utilizando la misma transacción.

Reintentos de transacción de cargador

ObjectGrid no admite transacciones XA o en dos fases. La hebra de grabación diferida elimina los registros de la correlación de cola y actualiza los registros del programa de fondo. Si se produce una anomalía en el servidor durante la transacción, puede que se pierdan algunas actualizaciones del programa de fondo.

El cargador de grabación diferida reintentará automáticamente la grabación de las transacciones con anomalías y enviará un objeto LogSequence en duda al programa de fondo para evitar la pérdida de datos. Esta acción requiere que el cargador sea idempotente, que significa que cuando Loader.batchUpdate(TxId, LogSequence) se llama dos veces con el mismo valor, el resultado es como si se aplicara sólo una vez. Las implementaciones de cargador deben implementar la interfaz RetryableLoader para habilitar esta característica. Consulte la documentación de la API para obtener información detallada.

Anomalías del cargador

El plug-in de cargador puede fallar cuando no puede comunicarse con el programa de fondo de la base de datos. Esto puede suceder si el servidor de bases de datos o la conexión de red está inactivo. El cargador de grabación diferida pondrá en cola las actualizaciones e intentará empujar los cambios de los datos al cargador de forma periódica. El cargador debe notificar al tiempo de ejecución de ObjectGrid que hay un problema de conectividad de base de datos; para ello, emitirá una excepción LoaderNotAvailableException.

Por lo tanto, la implementación del cargador debe distinguir entre una anomalía de datos o un anomalía física del cargador. La anomalía de datos debe emitirse o volver a emitirse como excepción LoaderException o OptimisticCollisionException, pero una anomalía física del cargador debe emitirse o volver a emitirse como excepción LoaderNotAvailableException. ObjectGrid maneja estas dos excepciones de manera diferente:

- Si el cargador de grabación diferida obtiene una excepción LoaderException, el cargador de grabación diferida considerará la anomalía como un error de los datos, como por ejemplo un error de clave duplicada. El cargador de grabación diferida anulará el proceso por lotes de la actualización, e intentará actualizar un registro cada vez para aislar la anomalía de los datos. Si se vuelve a obtener una excepción {{LoaderException}} durante la actualización de un registro, se crea un registro de actualización con errores y se anota en la correlación de actualizaciones con errores.
- Si el cargador de grabación diferida obtiene una excepción LoaderNotAvailableException, el cargador de grabación diferida la considerará como un error porque no puede conectase a la base de datos, por ejemplo, el programa de fondo de la base de datos está inactivo, una conexión de base de datos no está disponible, o la red no está activa. El cargador de grabación diferida esperará 15 segundos y después volverá a intentar realizar la actualización por lotes en la base de datos.

El error habitual es emitir una excepción LoaderException cuando debería emitirse una excepción LoaderNotAvailableException. Todos los registros puestos en cola en el cargador de grabación diferida pasan a ser registros de actualizaciones con anomalías, que anula el propósito del aislamiento de anomalías de programa de fondo.

Consideraciones sobre el rendimiento

El soporte de almacenamiento en memoria caché de grabación diferida aumenta el tiempo de respuesta al eliminar la actualización del cargador de la transacción. También aumenta el rendimiento de base de datos ya que las actualizaciones de base de datos se combinan. Es importante comprender la sobrecarga que supone la hebra de grabación diferida, que extrae los datos de la correlación de cola y los envía al cargador.

El número máximo de actualizaciones o el tiempo máximo de actualización debe ajustarse en función del entorno y de los patrones de uso esperados. Si el valor del número máximo de actualizaciones o el tiempo máximo de actualización es demasiado pequeño, la sobrecarga de la hebra de grabación diferida puede sobrepasar las ventajas. Si se especifica un valor elevado para estos dos parámetros, podría aumentarse el uso de memoria al poner en cola los datos y aumentarse el tiempo obsoleto de los registros de la base de datos.

Para obtener un rendimiento óptimo, ajuste los parámetros de grabación diferida de acuerdo con los factores siguientes:

- Índice de transacciones de lectura y grabación.
- Misma frecuencia de actualización de registros.
- Latencia de actualización de la base de datos.

Cargadores

Con un plug-in Loader plug-in, una correlación de cuadrícula de datos puede actuar como una memoria caché de datos para los datos que se mantienen normalmente en un almacén persistente en el mismo sistema o en otro sistema. Generalmente, se utiliza una base de datos o un sistema de archivos como almacenamiento persistente. Una máquina virtual Java (JVM) remota también se puede utilizar como el origen de datos, lo que permite crear memorias caché basadas en hub utilizando eXtreme Scale. Un cargador tiene la lógica para leer y escribir datos en un almacén persistente.

Visión general

Los cargadores son plug-ins de correlaciones de respaldo que se invocan cuando se realizan cambios en la correlación de respaldo o ésta no puede satisfacer una solicitud de datos (una falta de memoria caché). Se invoca el cargador cuando la memoria caché no puede satisfacer la solicitud de una clave, proporcionando la capacidad de lectura a través y el relleno poco activo de la memoria caché. Un cargador también permite actualizar la base de datos cuando los valores de la memoria caché cambian. Todos los cambios de una transacción se agrupan para minimizar el número de interacciones de la base de datos. Se utiliza un plug-in TransactionCallback junto con el cargador para desencadenar la demarcación de la transacción de fondo. Utilizar este plug-in es importante cuando se incluyen varias correlaciones en una única transacción, o cuando se desechan los datos de una transacción en la memoria caché sin confirmar.

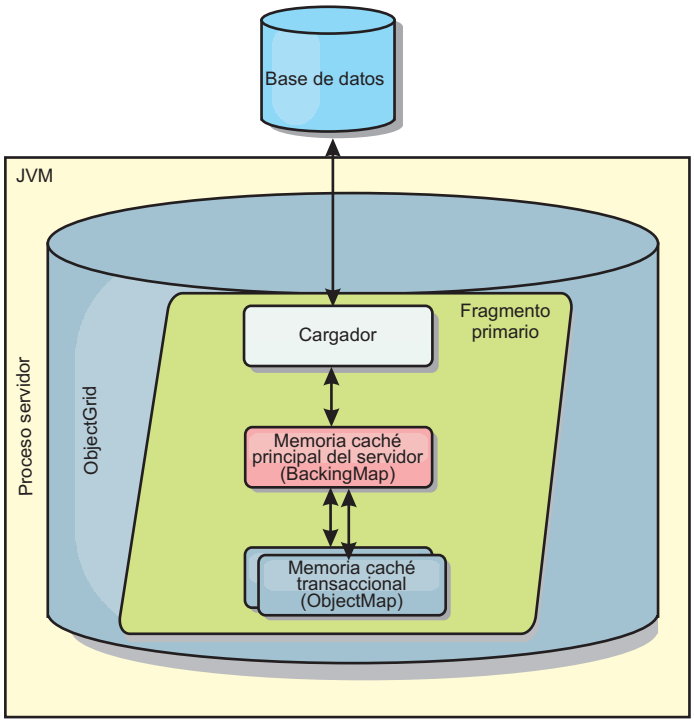


Figura 52. Cargador

El cargador también puede utilizar las actualizaciones sobrecualificadas para evitar mantener los bloqueos de base de datos. Al almacenar un atributo de versión en el valor de memoria caché, el cargador puede ver la imagen antes y después del valor tal como se actualiza en la memoria caché. Este valor se puede utilizar cuando se actualiza la base de datos o cuando se realiza un programa de fondo para verificar que los datos no se han actualizado. Un cargador también se puede configurar para precargar la cuadrícula de datos cuando se inicia. Cuando se realizan particiones, se asocia una instancia de cargador con cada partición. Si la correlación "Company" tiene diez particiones, hay diez instancias de cargador, una por partición primaria. Cuando se activa el fragmento primario de la correlación, se invoca el método preloadMap para el cargador de forma síncrona o asíncrona, que permite cargar automáticamente la partición de la correlación con los datos procedentes del programa de fondo. Cuando se invocan de forma síncrona, todas las transacciones de cliente se bloquean, lo que impide el acceso incoherente a la

cuadrícula de datos. De forma alternativa, se puede utilizar un precargador de cliente para cargar toda la cuadrícula de datos.

Dos cargadores incorporados pueden simplificar en gran medida la integración con los programas de fondo de la base de datos relacional. Los cargadores JPA utilizan las funciones de correlación de objetos relacionales (ORM) de ambas implementaciones, OpenJPA e Hibernate, de la especificación de JPA (Java Persistence API). Si desea más información, consulte "Cargadores JPA" en la página 66.

Si utiliza cargadores en una configuración de varios centros de datos, debe considerar cómo se mantiene la coherencia de los datos y la memoria caché entre las cuadrículas de datos. Para obtener más información, consulte "Consideraciones sobre el cargador en una topología multimaestro" en la página 172.

Configuración de cargador

Para añadir un cargador a la configuración de BackingMap, puede utilizar la configuración mediante programa o la configuración del archivo XML. Un cargador tiene la siguiente relación con una correlación de respaldo.

- Una correlación de respaldo sólo puede tener un cargador.
- Una correlación de respaldo de cliente (memoria caché cercana) no puede tener un cargador.
- Una definición de cargador se puede aplicar a varias correlaciones de respaldo, pero cada una de éstas tiene su propia instancia de cargador.

Precarga de datos y calentamiento

En muchos escenarios que incorporan el uso de un cargador, puede preparar la cuadrícula de datos precargándola con datos.

Cuando se utiliza como una memoria caché completa, la cuadrícula de datos debe alojar todos los datos y se debe cargar antes de que los clientes se puedan conectar a ella. Cuando se utiliza una memoria caché escasa, puede preparar la memoria caché con datos de forma que los clientes tengan acceso inmediato a los datos cuando estos se conecten.

Existen dos enfoques para la precarga de datos en la cuadrícula de datos: mediante un plug-in Loader o mediante un cargador de clientes, tal como se describe en las secciones siguientes.

Plug-in Loader

El plug-in Loader está asociado con cada correlación y es responsable de sincronizar un fragmento de partición primaria con la base de datos. El método preloadMap del plug-in Loader se invoca automáticamente cuando se activa un fragmento. Por ejemplo, si tiene 100 particiones, existen 100 instancias de cargador, y cada una carga los datos para su partición. Si se ejecuta de forma síncrona, todos los clientes se bloquean hasta que se complete la precarga.

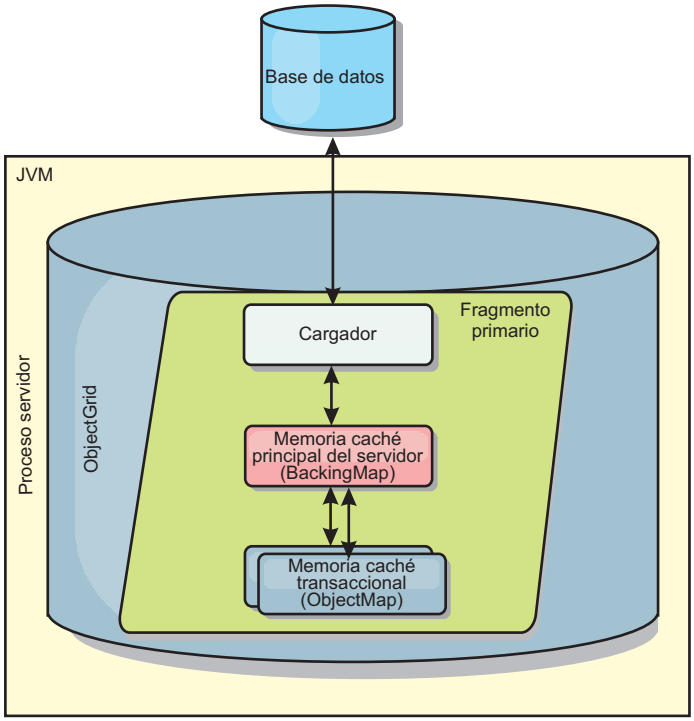


Figura 53. Plug-in Loader

Cargador de clientes

Un cargador de clientes es un patrón para utilizar uno o más clientes para carga la cuadrícula con datos. El uso de varios clientes para cargar los datos de cuadrícula puede ser eficaz cuando el esquema de partición no se almacena en la base de datos. Puede invocar los cargadores de clientes manual o automáticamente cuando se inicia la cuadrícula de datos. De forma opcional, los cargadores de clientes pueden utilizar StateManager para establecer el estado de la cuadrícula de datos en la modalidad de precarga, de forma que los clientes no pueden acceder a la cuadrícula mientras está precargando los datos. WebSphere eXtreme Scale incluye un cargador basado en JPA (Java Persistence API) que puede utilizar para cargar automáticamente la cuadrícula de datos con los proveedores OpenJPA o Hibernate JPA. Para obtener más información sobre los proveedores de memoria caché, consulte “Plug-in de memoria caché de nivel 2 (L2) JPA” en la página 26.

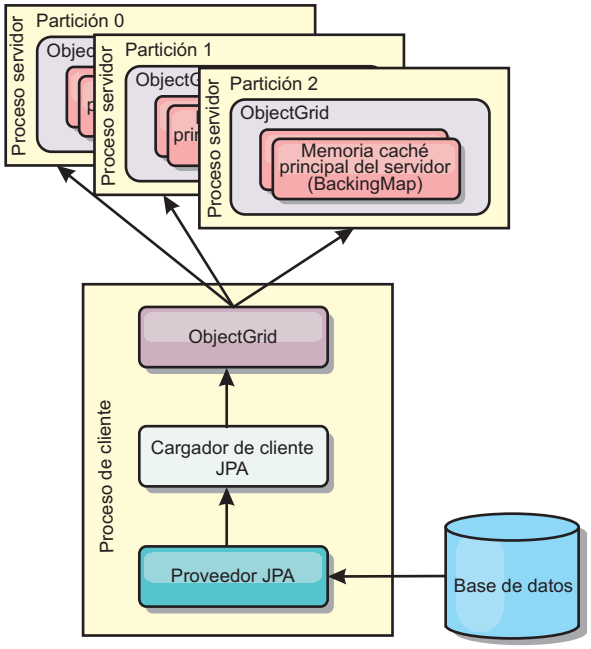


Figura 54. Cargador de clientes

Técnicas de sincronización de base de datos

Cuando se utiliza WebSphere eXtreme Scale como memoria caché, se deben escribir aplicaciones que admitan datos obsoletos si la base de datos puede actualizarse de forma independiente a una transacción de eXtreme Scale. Para servir como un espacio de proceso de base de datos en memoria sincronizado, eXtreme Scale proporciona distintos métodos para mantener la memoria caché actualizada.

Técnicas de sincronización de base de datos

Renovación periódica

La memoria caché se puede invalidar o actualizar de forma automática y periódica utilizando el actualizados de base de datos basado en el tiempo de JPA (Java Persistence API). El actualizador consulta periódicamente la base de datos utilizando un proveedor JPA para cualquier actualización o inserción que se haya producido desde la actualización anterior. Todos los cambios identificados se anulan o actualizan automáticamente cuando se utilizan con una memoria caché escasa. Si se utilizan con una memoria caché completa, las entradas se pueden descubrir e insertar en la memoria caché. Las entradas nunca se eliminan de la memoria caché.

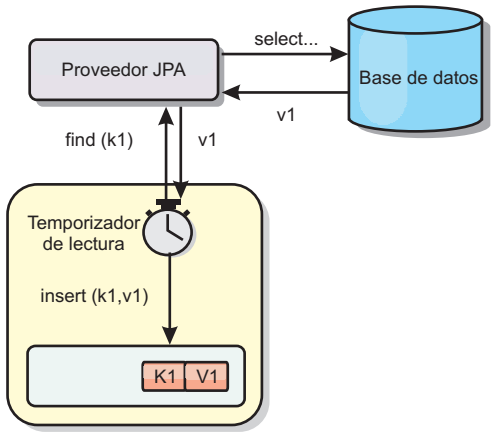


Figura 55. Renovación periódica

Desalojo

Las memorias caché escasas pueden utilizar políticas de desalojo para eliminar automáticamente datos de la memoria caché sin afectar a la base de datos. Existen tres políticas incorporadas incluidas en eXtreme Scale: tiempo de vida, menos usada recientemente y usada con menos frecuencia. Las tres políticas pueden, de forma opcional, desalojar datos de forma más agresiva a medida que la memoria pasa a estar limitada habilitando la opción de desalojo basado en memoria.

Anulación basada en sucesos

Las memorias caché escasas y completas se pueden invalidar o actualizar utilizando un generador de sucesos como, por ejemplo, JMS (Java Message Service). La anulación utilizando JMS puede unirse manualmente a cualquier proceso que actualiza el programa de fondo utilizando un desencadenante de base de datos. Se proporciona un plug-in JMS ObjectGridEventListener en eXtreme Scale que puede notificar a los clientes cuando la memoria caché del servidor tiene algún cambio. Esto puede disminuir la cantidad de tiempo que el cliente puede ver los datos obsoletos.

Anulación programática

Las API eXtreme Scale permiten la interacción manual de la memoria caché cercana y de servidor utilizando los métodos de API Session.beginNoWriteThrough(), ObjectMap.invalidate() y EntityManager.invalidate(). Si un proceso de cliente o servidor ya no necesita una parte de los datos, los métodos de anulación se pueden utilizar para eliminar datos de la memoria caché cercana o del servidor. El método beginNoWriteThrough se aplica cualquier operación ObjectMap o EntityManager a la memoria caché local sin llamar al cargador. Si se invoca desde un cliente, la operación sólo se aplica a la memoria caché cercana (el cargador remoto no se invoca). Si se invoca en el servidor, la operación sólo se aplica a la memoria caché principal del servidor sin invocar el cargador.

Invalidación de datos

Para eliminar los datos de memoria caché de escala, puede utilizar un mecanismo de invalidación basado en suceso o mediante programa.

Invalidación basada en sucesos

Las memorias caché escasas y completas se pueden invalidar o actualizar utilizando un generador de sucesos como, por ejemplo, JMS (Java Message Service). La anulación utilizando JMS puede unirse manualmente a cualquier proceso que actualiza el programa de fondo utilizando un desencadenante de base de datos. Se proporciona un plug-in JMS ObjectGridEventListener en eXtreme Scale que puede notificar a los clientes cuando la memoria caché de servidor cambia. Este tipo de notificación disminuye la cantidad de tiempo que el cliente puede ver los datos obsoletos.

La invalidación basada en sucesos consta normalmente de los tres componentes siguientes.

- **Cola de sucesos:** Una cola de sucesos almacena los sucesos de cambio de datos. Puede ser una cola JMS, una base de datos, una cola FIFO o cualquier clase de siempre que pueda gestionar los sucesos de cambio de datos.
- **Editor de sucesos:** Un editor de sucesos publica los sucesos de cambio de datos en la cola de sucesos. Un editor de sucesos es normalmente una aplicación que usted mismo crea o una implementación de plug-in de eXtreme Scale. El editor de sucesos sabe cuándo se cambian los datos o cambia los datos por sí mismo. Cuando se confirma una transacción, se generan los sucesos para los datos cambiados y el editor de sucesos publica estos sucesos en la cola de sucesos.
- **Consumidor de sucesos:** Un consumidor de sucesos consume sucesos de cambio de datos. El consumidor de sucesos es por lo general una aplicación para garantizar que los datos de la cuadrícula de destino se actualizan con el cambio más reciente de otras cuadrículas. Este consumidor de sucesos interactúa con la cola de sucesos para obtener los cambios de datos más recientes y aplica los cambios de datos en la cuadrícula de destino. Los consumidores de sucesos pueden utilizar las API de eXtreme Scale para invalidar datos obsoletos o actualizar la cuadrícula con los datos más recientes.

Por ejemplo, JMSObjectGridEventListener tiene una opción para un modelo cliente-servidor, en el cual la cola de sucesos es un destino de JMS designado. Todos los procesos del servidor son editores de sucesos. Cuando se confirma una transacción, el servidor obtiene los cambios de datos y los publica en la JMS de destino designada. Todos los procesos de cliente son consumidores de sucesos. Reciben los cambios de datos del destino de JMS designado y aplican los cambios en la memoria caché cercana del cliente.

Consulte el tema sobre la habilitación del mecanismo de invalidación del cliente en la *Guía de administración* si desea más información.

Anulación programática

Las API WebSphere eXtreme Scale permiten la interacción manual de la memoria caché cercana y de servidor utilizando los métodos de API Session.beginNoWriteThrough(), ObjectMap.invalidate() y EntityManager.invalidate(). Si un proceso de cliente o servidor ya no necesita una parte de los datos, los métodos de anulación se pueden utilizar para eliminar datos de la memoria caché cercana o del servidor. El método beginNoWriteThrough se aplica cualquier operación ObjectMap o EntityManager a la memoria caché local sin llamar al cargador. Si se invoca desde un cliente, la operación sólo se aplica a la memoria caché cercana (el cargador remoto no se invoca). Si se invoca en el servidor, la operación sólo se aplica a la memoria caché principal del servidor sin invocar el cargador.

Puede utilizar la anulación mediante programa con otras técnicas para determinar cuándo invalidar los datos. Por ejemplo, este método de invalidación utiliza mecanismos de invalidación basados en sucesos para recibir los sucesos de cambio de datos y luego utiliza interfaces de programación de aplicaciones para invalidar los datos obsoletos.

Índices

Utilice el plug-in MapIndexPlugin para crear un índice o varios índice en una BackingMap para dar soporte al acceso a datos no de clave.

Tipos de índices y configuración

La característica de indexación la representa el plug-in MapIndexPlugin, o Index de forma abreviada. Index es un plug-in BackingMap. Una BackingMap puede tener varios plug-ins Index configurados, siempre que cada uno de ellos siga las normas de configuración de Index.

Puede utilizar la característica de indexación para crear uno o más índices en una BackingMap. Un índice se crea a partir de un atributo o una lista de atributos de un objeto en la BackingMap. De esta manera, las aplicaciones pueden encontrar rápidamente determinados objetos. Con la característica de índices, las aplicaciones pueden encontrar objetos con un valor específico o dentro de un intervalo de valores de atributos indizados.

Existen dos tipos de índice: estático y dinámico. Con el índice estático, debe configurar el plug-in de índices en BackingMap antes de inicializar la instancia de ObjectGrid. Puede realizar esta configuración con una configuración de XML o mediante programa de la BackingMap. Los índices estáticos empiezan a construir un índice durante la inicialización de ObjectGrid. El índice siempre está sincronizado con la BackingMap y listo para ser utilizado. Después de que se inicie el proceso de indexación estática, el mantenimiento del índice forma parte del proceso de gestión de transacciones de eXtreme Scale. Cuando las transacciones confirman cambios, estos cambios también actualizan el índice estático y los cambios de índice se retrotraen si la transacción se retrotrae.

Con el índice dinámico, puede crear un índice en una correlación BackingMap antes o después de la inicialización de la instancia de ObjectGrid que contiene. Las aplicaciones tienen un control del ciclo de vida sobre el proceso de indexación dinámica, de forma que pueda eliminar un índice dinámico, cuando ya no sea necesario. Cuando una aplicación crea un índice dinámico, éste podría no estar listo para su uso inmediato debido al tiempo que tarda en completarse el proceso de creación del índice. Puesto que la cantidad de tiempo depende de la cantidad de datos indexados, se proporciona la interfaz DynamicIndexCallback para aplicaciones que desean recibir notificaciones cuando se produzcan determinados sucesos de indexación. Estos sucesos pueden incluir sucesos de error, destrucción y preparado. Las aplicaciones pueden implementar esta interfaz de devolución de llamada y registrarla con el proceso de índices dinámicos.

Si una BackingMap tiene un plug-in de índice configurado, podrá obtener el proxy de índice de aplicaciones de la ObjectMap correspondiente. Si se llama al método getIndex en la ObjectMap y se proporciona el nombre del plug-in de índice, se devolverá el objeto de proxy de índice. Debe difundir el objeto de proxy de índice en una interfaz apropiada de índice de aplicaciones como, por ejemplo, MapIndex, MapRangeIndex, o una interfaz personalizada de índices. Después de obtener el objeto de proxy de índice, puede utilizar los métodos definidos en la interfaz de índices de aplicación para buscar objetos almacenados en memoria caché.

En la lista siguiente se resumen los pasos que debe seguir para utilizar los índices:

- Añada plug-ins de índices estáticos o dinámicos a BackingMap.
- Obtenga el objeto de proxy de índice de aplicación; para ello, emita el método getIndex de ObjectMap.
- Difunda el objeto de proxy de índice a una interfaz de índices de aplicación apropiada, como MapIndex, MapRangeIndex, o a una interfaz de índices personalizada.
- Utilice los métodos definidos en una interfaz de índices de aplicación para buscar los objetos almacenados en memoria caché.

La clase HashIndex es la implementación de plug-in de índice que puede soportar ambas interfaces de índice de aplicación incorporadas: MapIndex y MapRangeIndex. También puede crear sus propios índices. Puede añadir HashIndex como un índice estático o dinámico en BackingMap, obtener un objeto proxy de índice MapIndex o MapRangeIndex y utilizar el objeto proxy de índice para encontrar los objetos almacenados en memoria caché.

Índice predeterminado

Si desea iterar a través de las claves en una correlación local, puede utilizar el índice predeterminado. Este índice no requiere ninguna configuración, pero se debe utilizar en el fragmento, utilizando una instancia de ObjectGrid o agente recuperada del método ShardEvents.shardActivated(ObjectGrid shard).

Consideraciones sobre la calidad de los datos

Los resultados de los métodos de consulta de índice sólo representan una instantánea de los datos en un momento puntual. No se obtiene ningún bloqueo contra la entrada de datos después de que los resultados vuelvan a la aplicación. La aplicación tiene que ser consciente de que se pueden producir actualizaciones de datos en un conjunto de datos devuelto. Por ejemplo, la aplicación obtiene la clave de un objeto almacenado en memoria caché ejecutando el método findAll de MapIndex. Este objeto de clave devuelto se asocia a una entrada de datos de la memoria caché. La aplicación debe poder ejecutar el método get en ObjectMap para encontrar un objeto proporcionando el objeto de clave. Si otra transacción elimina el objeto de datos de la memoria caché, justo antes de que se llame al método get, el resultado devuelto será nulo.

Consideraciones sobre el rendimiento de los índices

Uno de los principales objetivos de la característica de índices es mejorar el rendimiento global de BackingMap. Si los índices no se utilizan correctamente, podría verse afectado el rendimiento de la aplicación. Tenga en cuenta los siguientes factores antes de utilizar esta característica.

- **El número de transacciones de escritura simultáneas:** el proceso de índices se puede producir cada vez que una transacción escribe datos en una BackingMap. El rendimiento disminuye si hay muchas transacciones grabando datos en una correlación al mismo tiempo que una aplicación realiza operaciones de consulta de índices.
- **El tamaño del conjunto de resultados devuelto por una operación de consulta:** a medida que el tamaño del conjunto de resultados aumenta, el rendimiento de la consulta disminuye. El rendimiento tiene tendencia a disminuir si el tamaño del conjunto de resultados es un 15% o más de la BackingMap.

- **El número de índices creados sobre la misma BackingMap:** cada índice consume recursos del sistema. A medida que el número de índices creados sobre la BackingMap aumenta, disminuye el rendimiento.

La función de indexación puede mejorar el rendimiento de BackingMap de forma drástica. Los casos ideales se producen cuando la BackingMap tiene operaciones básicamente de lectura, el conjunto de resultados de la consulta es un pequeño porcentaje de las entradas de BackingMap, y sólo se crean unos pocos índices sobre la BackingMap.

Planificación de topologías de varios centros de datos

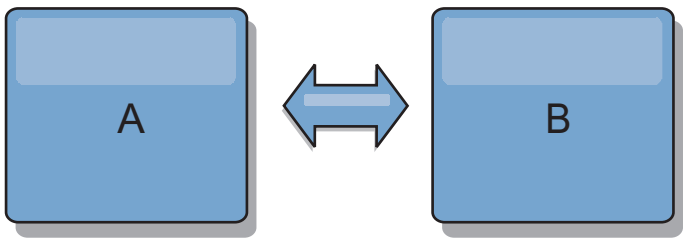
Mediante la utilización de la réplica asíncrona multimaestro, dos o más cuadrículas de datos pueden convertirse en copias exactas entre ellas. Cada cuadrícula de datos está alojada en un dominio de servicio de catálogo independiente, con su propio servicio de catálogo, servidores de contenedor y un nombre exclusivo. Con la réplica asíncrona multimaestro, puede utilizar enlaces para conectar una colección de dominios de servicio de catálogo. A continuación, los dominios de servicio de catálogo se sincronizan utilizando la réplica mediante los enlaces. Puede construir casi cada topología mediante la definición de enlaces entre los dominios de servicio de catálogo.

Topologías para réplica multimaestro

Tiene diversas opciones cuando elige una topología para el despliegue que incorpora réplica multimaestro.

Enlaces que conectan dominios de servicio de catálogo

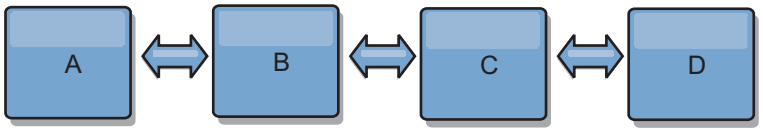
Una infraestructura de cuadrícula de datos de réplica es un gráfico conectado de dominios de servicio de catálogo con enlaces bidireccionales entre ellos. Con un enlace, dos dominios de servicio de catálogo pueden comunicar los cambios en los datos. Por ejemplo, la topología más sencilla es un par de dominios de servicio de catálogo con un único enlace entre ellos. Los dominios de servicio de catálogo se nombran alfabéticamente: A, B, C y así sucesivamente, desde la izquierda. Un enlace puede cruzar una red de área amplia (WAN), abarcando distancias grandes. Incluso si se interrumpe el enlace, aún puede cambiar los datos en cualquiera de los dos dominios de servicio de catálogo. La topología reconcilia los cambios cuando el enlace reconecta los dominios de servicio de catálogo. Los enlaces intentan volverse a conectar automáticamente si se interrumpe la conexión de red.



Después de haber configurado los enlaces, eXtreme Scale en primer lugar intenta hacer que cada dominio de servicio de catálogo sea idéntico. A continuación, eXtreme Scale intenta mantener las condiciones idénticas a los cambios producidos en cualquier dominio de servicio de catálogo. El objetivo es que cada dominio de servicio de catálogo sea un reflejo exacto de cada uno de los otros dominios de servicio de catálogo conectados mediante los enlaces. Los enlaces de réplica entre los dominios de servicio de catálogo ayudan a garantizar que los cambios realizados en un dominio se copian en los otros dominios.

Topologías de línea

Aunque es un despliegue muy simple, una topología de línea muestra algunas cualidades de los enlaces. En primer lugar, no es necesario que un dominio de servicio de catálogo esté conectado directamente a cada uno de los otros dominios de servicio de catálogo para que reciba los cambios. El Dominio B obtiene los cambios del Dominio A. El Dominio C recibe los cambios del Dominio A a través del Dominio B, que se conecta a los Dominios A y C. De forma similar, el Dominio D recibe los cambios de los otros dominios mediante el Dominio C. Esta capacidad esparce la carga de distribuir los cambios lejos del origen de los cambios.



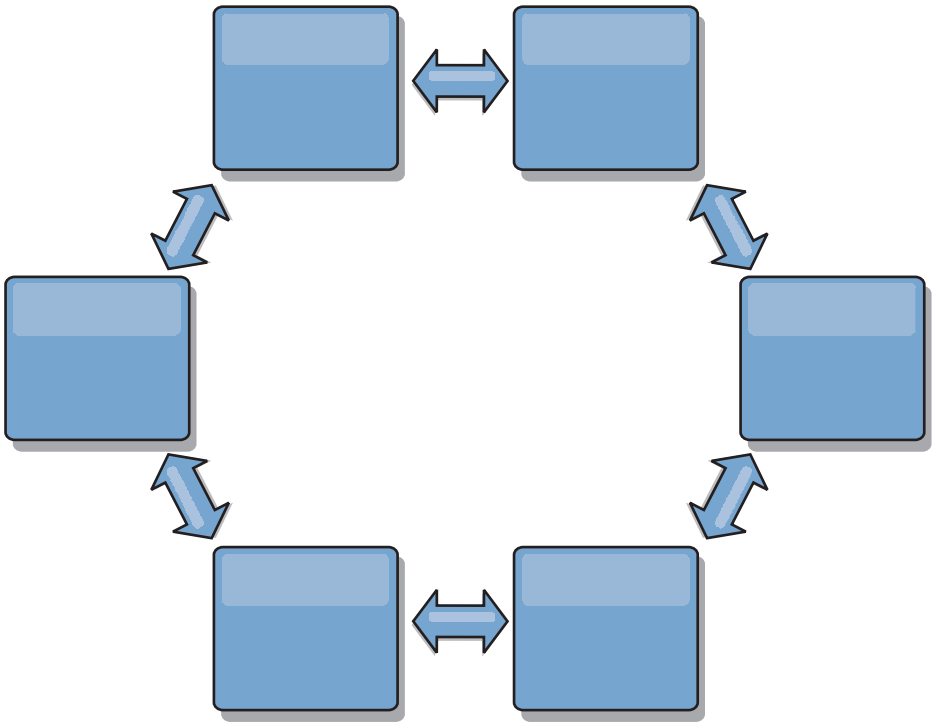
Tenga en cuenta que si el Dominio C falla, se producirán las acciones siguientes:

1. El Dominio D se quedará huérfano hasta que se reinicie el Dominio C
2. El Dominio C se sincronizará a sí mismo con el Dominio B, que es una copia del Dominio A
3. El Dominio D utilizará el Dominio C para sincronizarse a sí mismo con los cambios de los Dominios A y B. Estos cambios inicialmente se han producido mientras el Dominio D estaba huérfano (mientras el Dominio C estaba inactivo).

En última instancia, los Dominios A, B, C y D serán todos ellos idénticos entre ellos de nuevo.

Topologías de anillo

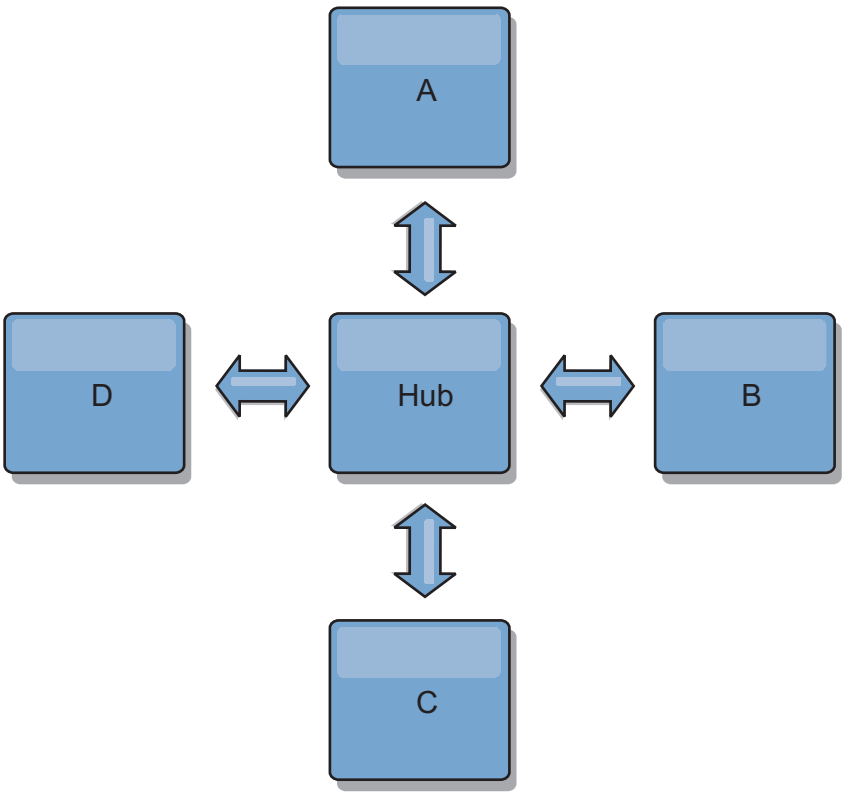
Las topologías de anillo son un ejemplo de una topología más flexible. Cuando un dominio de servicio de catálogo o un único enlace falla, los dominios de servicio de catálogo supervivientes todavía pueden obtener los cambios. Los dominios de servicio de catálogo viajan alrededor del anillo, alejándose de la anomalía. Cada dominio de servicio de catálogo tiene como máximo dos enlaces a otros dominios de servicio de catálogo, independientemente del tamaño de la topología de anillo. La latencia para propagar los cambios puede ser grande. Podría ser necesario que los cambios de un dominio de servicio de catálogo determinado viajaran a través de varios enlaces antes de que todos los dominios de servicio de catálogo tengan los cambios. Una topología de línea tiene la misma característica.



También puede desplegar una topología de anillo más sofisticada, con un dominio de servicio de catálogo raíz en el centro del anillo. El dominio de servicio de catálogo raíz funciona como el punto central de reconciliación. Los otros dominios de servicio de catálogo actúan como puntos remotos de reconciliación para los cambios que se producen en el dominio de servicio de catálogo raíz. El dominio de servicio de catálogo raíz puede arbitrar los cambios entre los dominios de servicio de catálogo. Si una topología de anillo contiene más de un anillo alrededor de un dominio de servicio de catálogo raíz, el dominio sólo puede arbitrar los cambios entre el anillo más interno. Sin embargo, los resultados del arbitraje se distribuyen por los dominios de servicio de catálogo de los otros anillos.

Topologías de hub y radio

Con una topología de hub y radio, los cambios viajan a través de un dominio de servicio de catálogo de hub. Debido a que el hub es el único dominio de servicio de catálogo intermedio especificado, las topologías de hub y radio tienen una latencia menor. El dominio de hub está conectado a cada dominio de radio mediante un enlace. El hub distribuye los cambios entre los dominios de servicio de catálogo. El hub actúa como un punto de reconciliación para las colisiones. En un entorno con una tasa de actualización alta, es posible que el hub necesite ejecutarse en más hardware que los radios para permanecer sincronizado. WebSphere eXtreme Scale está diseñado para escalar de forma lineal, lo que significa que puede ampliar el hub, según sea necesario, sin dificultad. Sin embargo, si el hub falla, los cambios no se distribuyen hasta que se reinicia el hub. Los cambios en los dominios de servicio de catálogo de radio se distribuirán una vez que se vuelva a conectar el hub.



También puede utilizar una estrategia con clientes completamente replicados, una variación de la topología que utiliza un par de servidores eXtreme Scale en ejecución como hub. Cada cliente crea una cuadrícula de datos de un solo contenedor autocontenida con un catálogo en la JVM de cliente. Un cliente utiliza su cuadrícula de datos para conectarse al catálogo de hub. Esta conexión hace que el cliente se sincronice con el hub tan pronto como el cliente obtenga una conexión del hub.

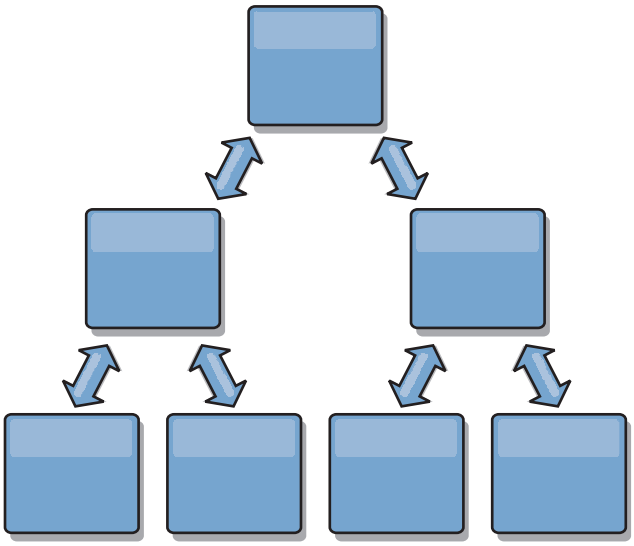
Los cambios realizados por el cliente son locales al cliente y se hace una réplica de ellos asíncrona en el hub. El hub actúa como un dominio de arbitraje, que distribuye los cambios a todos los clientes conectados. La topología de clientes completamente replicados proporciona una memoria caché L2 fiable para una correlacionador relacional de objetos como, por ejemplo, OpenJPA. Los cambios se distribuyen rápidamente entre las JVM de cliente a través del hub. Si el tamaño de memoria caché puede estar contenido en el espacio de almacenamiento intermedio disponible, la topología es una arquitectura fiable para este estilo de L2.

Utilice varias particiones para escalar el dominio del hub en varias JVM, si es necesario. Debido a que todos los datos aún deben caber en una única JVM de cliente, varias plataformas aumentan la capacidad del concentrador para distribuir y arbitrar cambios. Sin embargo, tener varias particiones no cambia la capacidad de un único dominio.

Topologías de árbol

También puede utilizar un árbol dirigido acíclico. Un árbol acíclico no tiene ciclos ni bucles, y una configuración dirigida limita los enlaces a los existentes solo entre padres e hijos. Esta configuración puede ser útil para topologías que tengan muchos dominios de servicio de catálogo y no es práctico tener un hub central que

esté conectado a todos los radios posibles. Este tipo de topología también puede resultar útil cuando debe añadir dominios de servicio de catálogo actualizando el dominio de servicio de catálogo raíz.



Una topología de árbol aún puede tener un punto central de reconciliación en el dominio de servicio de catálogo raíz. El segundo nivel aún puede funcionar como un punto remoto de reconciliación para los cambios que se producen en el dominio de servicio de catálogo por debajo de ellos. El dominio de servicio de catálogo raíz puede arbitrar los cambios entre los dominios de servicio de catálogo en el segundo nivel solamente. También puede utilizar árboles "n-ários", cada uno de los cuales tiene N hijos en cada nivel. Cada dominio de servicio de catálogo se conecta a *n* enlaces.

Clientes totalmente replicados

En esta variación de topología interviene un par de servidores eXtreme Scale que se ejecutan como un hub. Cada cliente crea una cuadrícula de datos de un solo contenedor autocontenida con un catálogo en la JVM de cliente. Un cliente utiliza su cuadrícula de datos para conectarse al catálogo de hub, lo que hace que el cliente se sincronice con el hub tan pronto como el cliente obtiene una conexión del hub.

Los cambios realizados por el cliente son locales al cliente y se hace una réplica de ellos asíncrona en el hub. El hub actúa como un dominio de arbitraje, que distribuye los cambios a todos los clientes conectados. La topología de clientes totalmente replicados proporciona una buena memoria caché L2 para un correlacionador relacional de objetos, como OpenJPA. Los cambios se distribuyen rápidamente entre las JVM de cliente a través del hub. Siempre que el tamaño de la memoria caché se pueda incluir en el espacio de almacenamiento dinámico disponible de los clientes, esta topología es una buena arquitectura para este estilo de L2.

Utilice varias particiones para escalar el dominio del hub en varias JVM, si es necesario. Dado que todos los datos todavía deben caber en una sola JVM cliente, el uso de varias particiones aumenta la capacidad del hub para distribuir y arbitrar los cambios, pero no cambia la capacidad de un dominio único.

Consideraciones sobre la configuración para topologías multimaestro

Considere los puntos siguientes cuando decida si desea utilizar topologías de réplica multimaestro y cómo utilizarlas.

- **Requisitos de conjunto de correlaciones**

  Los conjuntos de correlaciones deben tener las características siguientes para replicar cambios en todos los enlaces del dominio de servicio de catálogo:

  – El nombre de ObjectGrid y el nombre de conjunto de correlaciones de un dominio de servicio de catálogo deben coincidir con el nombre de ObjectGrid y el nombre de conjunto de correlaciones de otros dominios de servicio de catálogo. Por ejemplo, el ObjectGrid "og1" y el conjunto de correlaciones "ms1" deben estar configurados en el dominio de servicio de catálogo A y el dominio de servicio de catálogo B para replicar los datos del conjunto de correlaciones entre los dominios de servicio de catálogo.

  – Es una cuadrícula de datos `FIXED_PARTITION`. Las cuadrículas de datos `PER_CONTAINER` no se pueden replicar.

  – Tiene el mismo número de particiones en cada dominio de servicio de catálogo. El conjunto de correlaciones podría tener o no el mismo número y los mismos tipos de réplicas.

  – Tiene los mismos tipos de datos que se están replicando en cada dominio de servicio de catálogo.

  – Contiene las mismas correlaciones y plantillas de correlación dinámica en cada uno de los dominios de servicio de catálogo.

  – No utiliza el gestor de entidades. Un conjunto de correlaciones que contiene una correlación de entidades no se replica en todos los dominios de servicio de catálogo.

  – No utiliza el soporte de almacenamiento en memoria caché de grabación diferida. Un conjunto de correlaciones que contiene una correlación que está configurada con soporte de grabación diferida no se replica en todos los dominios de servicio de catálogo.

  Los conjuntos de correlaciones con las características anteriores empiezan la réplica una vez que se han iniciado los dominios de servicio de catálogo en la topología.

- **Cargadores de clases con varios dominios de servicio de catálogo**

  Los dominios de servicio de catálogo deben tener acceso a todas las clases utilizadas como claves y valores. Todas las dependencias se deben reflejar en todas las vías de acceso de clases para máquinas virtuales Java (JVM) de contenedor de cuadrícula de datos para todos los dominios. Si un plug-in CollisionArbiter recupera el valor para una entrada de memoria caché, las clases para los valores deben estar presentes para el dominio que inicia el árbitro.

Consideraciones sobre el cargador en una topología multimaestro

Cuando se utilizan cargadores en una topología multimaestro, debe considerar los posibles retos de mantenimiento de la información de revisión y colisión. La cuadrícula de atos mantiene información de revisión sobre los elementos de la cuadrícula de datos de forma que se pueden detectar las colisiones cuando otros fragmentos primarios de la configuración graban entradas en la cuadrícula de datos. Cuando se añaden entradas desde un cargador, esta información de revisión no se incluye y la entrada asume una revisión nueva. Debido a que la revisión de la entrada parece una inserción nueva, se produciría una falta colisión si otro fragmento primario también cambia este estado u obtiene la misma información de un cargador.

Los cambios de la réplica invocan el método get en el cargador con una lista de las claves que no están aún en la cuadrícula de datos pero que se han a cambiar durante la transacción de réplica. Cuando se produce la réplica, estas entradas son entradas de colisión. Cuando se arbitran las colisiones y se aplica la revisión, se llama a una actualización por lotes en el cargador para aplicar los cambios en la base de datos. Todas las correlaciones modificadas en la ventana de revisión se actualizan en la misma transacción.

Interrogante de la precarga

Considere una topología de dos centros de datos con el centro de datos A y el centro de datos B. Ambos centros de datos tienen bases de datos independientes, pero solo el centro de datos A tiene una cuadrícula de datos en ejecución. Al establecer un enlace entre los cnetros de datos para una configuración multimaestro, las cuadrículas de datos del centro de datos A inician el envío de datos a las nuevas cuadrículas de datos del centro de datos B, lo que causa una colisión con cada entrada. Otro problema importante que se produce es con los datos que se encuentra en la base de datos del centro de datos B pero no en la base de atos del centro de datos A. Estas filas no se llenan ni arbitran, lo que genera incoherencias que no se resuelven.

Solución al interrogante de la precarga

Debido a que los datos que se encuentran solo en la base de datos no puede tener revisiones, debe precargar siempre completamente la cuadrícula de datos desde la base de datos local antes de establecer un enlace multimaestro. A continuación, ambas cuadrículas de datos pueden revisar y arbitrar los datos, y finalmente llegar a un estado coherente.

Interrogante de la memoria caché escasa

Con una memoria caché escasa, en primer lugar la aplicación intenta encontrar datos en la cuadrícula de datos. Si los datos no se encuentran en la cuadrícula de datos, se buscan los dato s en la base de atos utilizando el cargador. Se desalojan periodicamente entradas de la cuadrícula de datos para mantener un tamaño pequeño de la memoria caché.

Este tipo de memoria caché puede ser problemático en un escenario de configuración multimaestro porque las entradas de la cuadrícula de datos tienen metadatos de revisión que le ayudarán a detectar qué colisiones se producen y qué lado ha realizado cambios. Cuando los enlaces entre los centros de datos no funcionan, un centro de datos puede actualizar una entrada y a continuación en última instancia actualizar la base de datos e invalidar la entrada en la cuadrícula de datos. Cuando se recupera el enlace, los centros de datos intentan sincronizar las revisiones entre ellos. Sin embargo, debido a que la base de datos se ha actualizado y la entrada de la cuadrícula de datos se ha invalidado, el cambio se pierde desde la perspectiva del centro de datos que se ha interrumpido. Como resultado, los dos lados de la cuadrícula de datos están desincronizados y no son coherentes.

Solución a los interrogantes de memoria caché escasa

Topología y hub y radio:

Puede ejecutar el cargador solo en el hub de una topología de hub y radio, lo que mantiene la coherencia de los datos al mismo tiempo que se escala la cuadrícula de

datos. Sin embargo, si está considerando el despliegue, tenga en cuenta que los cargadores pueden permitir que la cuadrícula de datos se cargue parcialmente, lo que significa que se ha configurado el desalojador. Si los radios de la configuración son memorias caché escasas pero no tienen cargadores, las faltas de coincidencia de memoria caché no tienen ninguna manera de recuperar los datos de la base de datos. Debido a esta restricción, debe utilizar una topología de memoria caché llenada completamente con una configuración de hub y radio.

Invalidaciones y desalojo

La invalidación crea coherencia entre la cuadrícula de datos y la base de datos. Los datos se pueden eliminar de la cuadrícula de datos mediante programación o con desalojo. Al desarrollar la aplicación, debe tener en cuenta que el manejo de revisiones no replica los cambios que se han invalidado, lo que genera incoherencias entre fragmentos primarios.

Los sucesos de invalidación no son cambios de estado de memoria caché y no generan réplica. Los desalojadores configurados se ejecutan independientemente de otros desalojadores de la configuración. Por ejemplo, podría tener un desalojador configurado para un umbral de memoria en un dominio de servicio de catálogo, pero un tipo distinto de desalojador menos agresivo en el servicio de catálogo enlazado. Cuando se eliminan entradas de cuadrícula de datos debido a la política de umbral de memoria, las entradas del otro dominio de servicio de catálogo no resultan afectadas.

Actualizaciones de base de datos e invalidación de cuadrícula de datos

Se producen problemas al actualizar la base de datos directamente en segundo plano al llamar a la invalidación en la cuadrícula de datos para las entradas actualizadas en una configuración multimaestro. Este problema se produce porque la cuadrícula de datos no puede replicar el cambio en otros fragmentos primarios hasta que algún tipo de acceso de memoria caché mueve la entrada a la cuadrícula de datos.

Varios grabadores en una única base de datos lógica

Cuando utiliza una única base de datos con varios fragmentos primarios que se conectan mediante un cargador, se producen conflictos de transacciones. La implementación de cargador debe manejar especialmente estos tipos de escenarios.

Duplicación de datos utilizando réplica multimaestro

Puede configurar bases de datos independientes conectadas a dominios de servicio de catálogo independiente. En esta configuración, el cargador puede enviar cambios de un centro de datos al otro centro de datos.

Consideraciones sobre el diseño para la réplica multimaestro

Al implementar la réplica multimaestro, debe tener en cuenta aspectos del diseño como los siguientes: arbitraje, enlace y rendimiento.

Consideraciones sobre arbitraje en el diseño de topología

Se podrían producir colisiones de cambio si se pueden cambiar en dos lugares a la vez los mismos registros. Configure cada uno de los dominios de servicio de catálogo para que tenga aproximadamente la misma cantidad de recursos de procesador, memoria y red. Podría observar que los dominios de servicio de

catálogo que realicen el manejo de colisiones de cambio (arbitraje) utilicen más recursos que otros dominios de servicio de catálogo. Las colisiones se detectan automáticamente. Se manejan con uno de dos mecanismos:

- **Árbitro de colisión predeterminado**: el protocolo predeterminado utilizará los cambios del dominio de servicio de catálogo con el nombre léxicamente inferior. Por ejemplo, si los dominios de servicio de catálogo A y B generan un conflicto para un registro, el cambio del dominio de servicio de catálogo B se ignorará. El dominio de servicio de catálogo A mantiene su versión y el registro en el dominio de servicio de catálogo B se cambia para que coincida con el registro del dominio de servicio de catálogo A. Este comportamiento se aplica también a las aplicaciones en las que los usuarios o sesiones normalmente se enlazan o tienen una afinidad con una de las cuadrículas siguientes.
- **Árbitro de colisiones personalizado**: las aplicaciones pueden proporcionar un árbitro personalizado. Cuando un dominio de servicio de catálogo detecta una colisión, se inicia un árbitro. Para obtener información sobre cómo desarrollar un árbitro personalizado útil, consulte Desarrollo de árbitros personalizados para la réplica con varios maestros.

Para topologías en las que las colisiones son posibles, considere implementar una topología de hub y radio o una topología de árbol. Estas dos topologías son propicias para evitar colisiones constantes, lo que puede suceder en los escenarios siguientes:

1. Varios dominios de servicio de catálogo sufren una colisión
2. Cada dominio de servicio de catálogo maneja la colisión localmente, lo que genera revisiones
3. Las revisiones colisionan, con lo que se producen revisiones de revisiones

Para evitar colisiones, elija un dominio de servicio de catálogo específico, denominado un *dominio de servicio de catálogo de arbitraje* como el árbitro de colisión para un subconjunto de dominios de servicio de catálogo. Por ejemplo, una topología de hub y radio podría utiliza el hub como el manejador de colisiones. El manejador de colisiones de radio ignora las colisiones detectadas por los dominios de servicio de catálogo de radio. El dominio de servicio de catálogo de hub crea revisiones, lo que evita revisiones de colisiones inesperadas. El dominio de servicio de catálogo que se asigna para manejar colisiones debe enlazar a todos los dominios para los que es responsable para manejar colisiones. En una topología de árbol, los dominios padre internos manejan colisiones para sus hijos inmediatos. Por el contrario, si utiliza una topología en anillo, no puede designar un dominio de servicio de catálogo en el anillo como el árbitro.

En la tabla siguiente se resumen los enfoques de arbitraje que son más compatibles con distintas topologías.

Tabla 8. Enfoques de arbitraje. En esta tabla se indica si el arbitraje de la aplicación es compatible con distintas tecnologías.

Topología	¿Arbitraje de aplicación?	Notas
Una línea de dos dominios de servicio de catálogo	Sí	Elija un dominio de servicio de catálogo como árbitro.

Tabla 8. Enfoques de arbitraje (continuación). En esta tabla se indica si el arbitraje de la aplicación es compatible con distintas tecnologías.

Topología	¿Arbitraje de aplicación?	Notas
Una línea de tres dominios de servicio de catálogo	Sí	El dominio de servicio de catálogo intermedio debe ser el árbitro. Considere el dominio de servicio de catálogo intermedio como hub en una topología de hub y radio simple.
Una línea de más de tres dominios de servicio de catálogo	No	No se admite el arbitraje de aplicaciones.
Un hub con N radios	Sí	El hub con enlaces a todos los radios debe ser el dominio de servicio de catálogo de arbitraje.
Un anillo de N dominios de servicio de catálogo	No	No se admite el arbitraje de aplicaciones.
Un árbol dirigido acíclico (árbol n-ario)	Sí	Todos los nodos raíz deben evaluar solo sus descendientes directos.

Consideraciones sobre enlaces en el diseño de topología

De forma ideal, una topología incluye el número mínimo de enlaces cuando optimiza los compromisos entre las características de latencia de cambios, tolerancia a errores y rendimiento.

- **Latencia de cambios**

  La latencia de cambios la determina el número de dominios de servicio de catálogo intermedio por los que debe pasar un cambio antes de llegar a un dominio de servicio de catálogo específico.

  Una topología tiene la mejor latencia de cambios cuando elimina dominios de servicio de catálogo intermedios enlazando cada dominio de servicio de catálogo a cada uno de los otros dominios de servicio de catálogo. Sin embargo, un dominio de servicio de catálogo debe realizar trabajo de réplica en proporción a su número de enlaces. Para topologías grandes, el gran número de enlaces que se definirán puede causar carga administrativa.

  La velocidad a la que se copia un cambio en otros dominios de servicio de catálogo depende de factores adicionales, como por ejemplo:

  – Procesador y ancho de banda de red en el dominio de servicio de catálogo de origen
  – Número de dominios de servicio de catálogo intermedios y enlaces entre los dominios de servicio de catálogo de origen y de destino
  – Recursos de procesador y de red disponibles a los dominios de servicio de catálogo de origen, de destino e intermedio

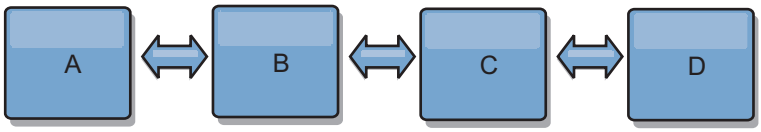
- **Tolerancia al error**

  La tolerancia a errores la determina el número de vías de acceso existentes entre los dos dominios de servicio de catálogo para la réplica de cambios.

  Si solo tiene un enlace entre un par determinado de dominios de servicio de catálogo, una anomalía de enlace no permite la propagación de cambios. De forma similar, los cambios no se propagan entre los dominios de servicio de catálogo si alguno de los dominios intermedios experimenta anomalía de enlace. La topología podría tener un único enlace desde un dominio de servicio de

catálogo a otro de tal forma que el enlace pase por dominios intermedios. Si es así, los cambios no se propagarán si alguno de los dominios de servicio de catálogo intermedios está inactivo.

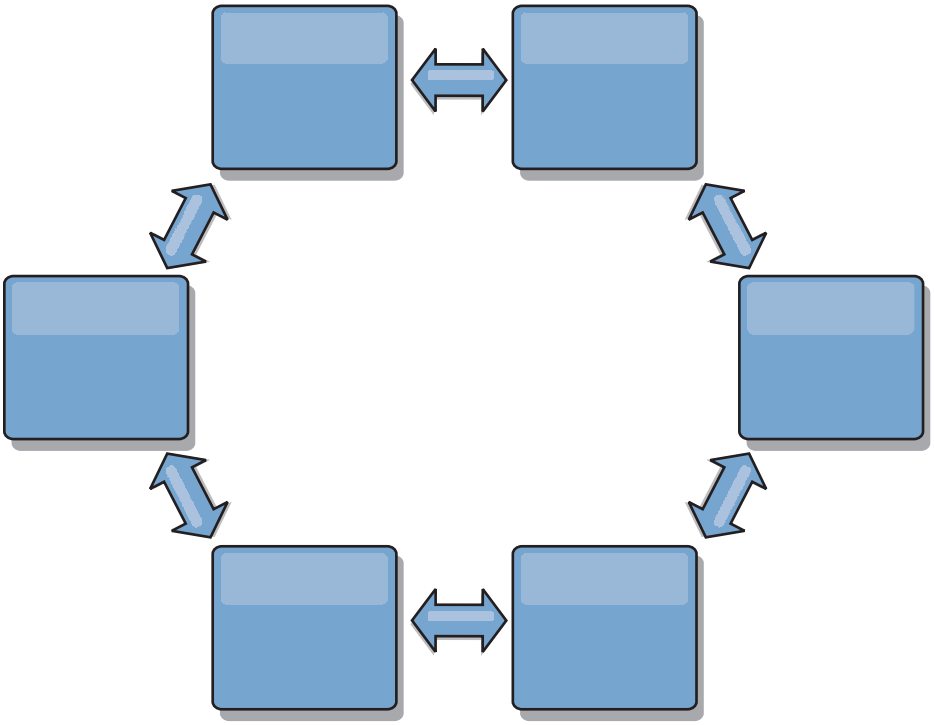
Considere la topología de línea con cuatro dominios de servicio de catálogo A, B, C, y D:



Si se mantiene alguna de estas condiciones, el Dominio D no verá los cambios de A:

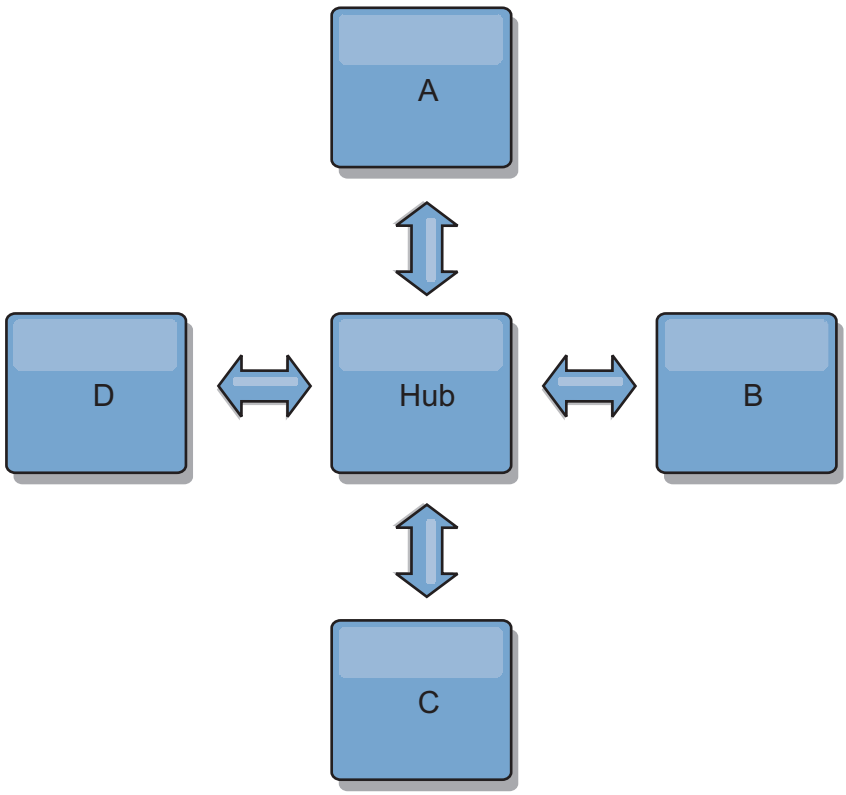
- El dominio A está activo y el dominio B está inactivo
- Los dominios A y B están activos y el dominio C está inactivo
- El enlace entre A y B está inactivo
- El enlace entre B y C está inactivo
- El enlace entre C y D está inactivo

En cambio, con una topología de anillo, cada uno de los dominios de servicio de catálogo puede recibir cambios desde cualquier dirección.



Por ejemplo, si un servicio de catálogo determinado de la topología de anillo está inactivo, los dos dominios adyacentes aún pueden obtener cambios directamente uno del otro.

Todos los cambios se propagan mediante el hub. Por lo tanto, a diferencia de las topologías de línea y de anillo, el diseño de hub y radio puede desglosarse, si el hub falla.



Un único dominio de servicio de topología es resistente a una determinada cantidad de pérdida de servicio. Sin embargo, anomalías mayores como interrupciones de la red amplia o la pérdida de enlaces entre centros de datos físicos puede interrumpir cualquiera de los dominios de servicio de catálogo.

- **Enlace y rendimiento**

  El número de enlaces definidos en un dominio de servicio de catálogo afecta al rendimiento. Más enlaces utilizan más recursos y como resultado el rendimiento de la réplica puede disminuir. La posibilidad de recuperar cambios para un dominio A mediante otros dominios libera de forma efectiva al dominio A de tener que replicar las transacciones en todas partes. La carga de distribución de cambios de un dominio está limitada por el número de enlaces que utiliza, no por cuántos dominios haya en la topología. Esta propiedad de carga proporciona escalabilidad, de forma que los dominios de la topología pueden compartir la carga de la distribución de cambios.

  Un dominio de servicio de catálogo puede recuperar los cambios indirectamente mediante otros dominios de servicio de catálogo. Considere una topología de línea con cinco dominios de servicio de catálogo.

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

  – A extrae los cambios de B, C, D y E a B
  – B extrae los cambios de A y C directamente y los cambios de D y E a C
  – C realiza los cambios de B y D directamente y los cambios de A a B y de E a D
  – D extrae los cambios de C y E directamente y los cambios de A y B a C
  – E extrae los cambios de D directamente, y los cambios de A, B y C a D

  La carga de distribución de los dominios de servicio de catálogo A y E es la menor, ya que cada uno de ellos tiene un enlace a un único dominio de servicio de catálogo. Cada uno de los dominios B, C y D tiene un enlace a dos dominios. Por lo tanto, la carga de distribución de los dominios B, C y D es el doble de la

carga de los dominios A y E. La carga de trabajo depende del número de enlaces de cada dominio, no del número global de dominios de la topología. Por lo tanto, la distribución de cargas descrita permanecería constante, incluso si la línea contuviera 1000 dominios.

Consideraciones sobre el rendimiento de réplica multimaestros

Tenga en cuenta las limitaciones siguientes cuando utilice topologías de réplica multimaestro:

- **Cambiar ajuste de distribución**, se trata en la sección anterior.
- **Rendimiento de enlace de réplica** WebSphere eXtreme Scale crea un único socket TCP/IP entre cualquier par de JVM. Todos el tráfico entre las JVM se produce entre el único socket, incluido tráfico de la réplica multimaestro. Los dominios de servicio de catálogo se alojan en como mínimo *n* JVM de contenedor, lo que proporciona como mínimo *n* enlaces TCP a dominios de servicio de catálogo de igual. Por lo tanto, los dominios de servicio de catálogo con una gran cantidad de contenedores tienen niveles más altos de rendimiento de la réplica. Más contenedores requieren más recursos de procesador y red.
- **Ajuste de la ventana deslizante TCP y RFC 1323** El soporte de RFC 1323 en ambos extremos de un enlace proporciona más datos para un viaje de ida y vuelta. Este soporte produce un mejor rendimiento, ampliando la capacidad de la ventana en un factor de aproximadamente 16.000.

  Recuerde que los sockets TCP utilizan un mecanismo de ventana deslizando para controlar el flujo de datos masivo. Este mecanismo normalmente limita el socket a 64 KB para un intervalo de viaje de ida y vuelta. Si el intervalo de viaje de ida y vuelta es 100 ms, el ancho de banda se limita a 640 KB/segundo sin ajuste adicional. El uso de todo el ancho de banda disponible en un enlace podría requerir un ajuste específico de un sistema operativo. La mayoría de sistemas operativos incluyen parámetros de ajuste, incluidas las opciones de RFC 1323, para ampliar el rendimiento sobre los enlaces de latencia alta.

  Varios factores pueden afectar al rendimiento de la réplica:

  – La velocidad a la que eXtreme Scale recupera cambios.
  – La velocidad a la que eXtreme Scale puede dar servicio a solicitudes de recuperación de réplica.
  – La capacidad de la ventana deslizante.
  – Con el ajuste de almacenamiento intermedio de red en ambos lados de un enlace, eXtreme Scale recupera cambios sobre el socket de forma eficiente.
- **Serialización de objetos** Todos los datos deben ser serializables. Si un dominio de servicio de catálogo no utiliza `COPY_TO_BYTES`, el dominio de servicio de catálogo debe utilizar Java o ObjectTransformers para optimizar el rendimiento de serialización.
- **Compresión** WebSphere eXtreme Scale comprime todos los datos enviados entre dominios de servicio de catálogo de forma predeterminada. La inhabilitación de la compresión no está disponible actualmente.
- **Ajuste de la memoria** El uso de memoria para una topología de réplica multimaestro es considerablemente independiente del número de dominios de servicio de catálogo de la topología.

  La réplica multimaestro añade una cantidad fija de proceso por entrada Map para manejar el mantenimiento de versiones. Cada contenedor también realiza un seguimiento de una cantidad fija de datos para cada dominio de servicio de catálogo de la topología. Una topología con dos dominios de servicio de catálogo utiliza aproximadamente la misma memoria que una topología con 50 dominios de servicio de catálogo. WebSphere eXtreme Scale no utiliza registros

de reproducción o colas similares en su implementación. Por lo tanto, no hay ninguna estructura de recuperación lista en el caso de que un enlace de réplica no esté disponible durante el periodo de tiempo considerable y se reinicie posteriormente.

Interoperatividad con otros productos WebSphere

Puede integrar WebSphere eXtreme Scale con otros productos de servidor como, por ejemplo, WebSphere Application Server y WebSphere Application Server Community Edition.

WebSphere Application Server

Puede integrar WebSphere Application Server con diversos aspectos de la configuración de WebSphere eXtreme Scale. Puede desplegar aplicaciones de cuadrícula de datos y utilizar WebSphere Application Server para alojar los servidores de contenedor y catálogo. También puede utilizar la seguridad de WebSphere Application Server en el entorno de WebSphere eXtreme Scale.

WebSphere Portal

Puede persistir sesiones HTTP de WebSphere Portal en una cuadrícula de datos en WebSphere eXtreme Scale.

WebSphere Application Server Community Edition

WebSphere Application Server Community Edition puede compartir el estado de sesión, pero no de una forma eficaz y escalable. WebSphere eXtreme Scale proporciona un alto rendimiento, una capa de persistencia distribuida que puede utilizarse para replicar el estado, pero que no se integra fácilmente con otro servidor de aplicaciones fuera de WebSphere Application Server. Puede integrar estos dos productos para proporcionar una solución de gestión de sesiones escalable.

WebSphere Real Time

Con el soporte de WebSphere Real Time, la oferta Java de tiempo real líder del sector, WebSphere eXtreme Scale, permite a las aplicaciones Extreme Transaction Processing (XTP) tener tiempos de respuesta coherentes y predecibles.

Capítulo 3. Escenarios

Un escenario presenta ejemplos para ayudar al usuario a comprender un concepto o realizar una tarea. El escenario utiliza información real para crear una imagen completa.

Utilización de un entorno OSGi para desarrollar y ejecutar plug-ins de eXtreme Scale

Utilice estos escenarios para completar tareas comunes en un entorno OSGi. Por ejemplo, la infraestructura OSGi es ideal para iniciar servidores y clientes en un contenedor OSGi, lo que le permite añadir y actualizar dinámicamente plug-ins de WebSphere eXtreme Scale en el entorno de ejecución.

Los siguientes escenarios son sobre la creación y ejecución de plug-ins dinámicos, lo que le permite instalar, iniciar, detener, modificar y desinstalar plug-ins. También puede completar otro escenario probable, lo que le permite utilizar la infraestructura OSGi sin posibilidades dinámicas. Puede seguir empaquetando las aplicaciones como paquetes, que se definen y comunican mediante servicios. Estos paquetes basados en servicios ofrecen muchas ventajas, que incluyen posibilidades de desarrollo y despliegue más eficaces.

Objetivos del escenario

Después de completar las lecciones de este módulo, sabrá cómo completar las tareas siguientes:

- Crear plug-ins dinámicos de eXtreme Scale para utilizar en un entorno OSGi.
- Ejecutar contenedores de eXtreme Scale en un entorno OSGi sin prestaciones dinámicas.

Requisitos previos

Lea el tema “Visión general de la infraestructura OSGi” en la página 24 para obtener más información sobre el soporte de OSGi y las ventajas que puede ofrecer.

Visión general de la infraestructura OSGi

OSGi define un sistema de módulo dinámico para Java. La plataforma de servicio OSGi tiene una arquitectura por capas, y está diseñada para ejecutarse en diversos perfiles Java estándar. Puede iniciar servidores y clientes de WebSphere eXtreme Scale en un contenedor OSGi.

Ventajas de la ejecución de aplicaciones en el contenedor OSGi

El soporte OSGi de WebSphere eXtreme Scale le permite desplegar el producto en la infraestructura OSGi de Eclipse Equinox. Anteriormente, si deseaba actualizar los plug-ins utilizados por eXtreme Scale, tenía que reiniciar la máquina virtual Java (JVM) para aplicar las nuevas versiones de los plug-ins. Con la prestación de actualización dinámica que proporciona la infraestructura OSGi, ahora puede actualizar las clases de plug-in sin reiniciar la JVM. Estos plug-ins los exportan los

paquetes de usuario como servicios. WebSphere eXtreme Scale accede al servicio o a los servicios buscándolos en el registro OSGi.

Los contenedores de eXtreme Scale se pueden configurar para que se inicien de forma más fácil y dinámica utilizando el servicio de administración de configuración OSGi o con OSGi Blueprint. Si desea desplegar una cuadrícula de datos nueva con la estrategia de colocación, puede hacerlo creando una configuración OSGi o desplegando un paquete con archivos XML de descriptor de eXtreme Scale. Con el soporte de OSGi, los paquetes de aplicaciones que contienen eXtreme Scale se pueden instalar, iniciar, detener, actualizar y desinstalar sin reiniciar todo el sistema. Con esta posibilidad, puede actualizar la aplicación sin interrumpir la cuadrícula de datos.

Se pueden configurar beans y servicios de plug-in con ámbitos de fragmento personalizados, lo que permite opciones de integración sofisticadas opciones con otros servicios que se ejecutan en la cuadrícula de datos. Cada plug-in puede utilizar clasificaciones OSGi Blueprint para verificar que cada instancia del plug-in está activada en la versión correcta. Se proporcionan un bean gestionado por OSGi (MBean) y el programa de utilidad **xscmd**, que permiten consultar los servicios OSGi de plug-in de eXtreme Scale y sus clasificaciones.

Esta prestación permite a los administradores reconocer rápidamente los errores potenciales de configuración y administración y actualizar las clasificaciones de servicio de plug-in utilizadas por eXtreme Scale.

Paquetes OSGi

Para interactuar con los plug-ins y desplegarlos en la infraestructura OSGi, debe utilizar *paquetes*. En la plataforma de servicio OSGi, un paquete es un archivo de archivado Java (JAR) que contiene código Java, recursos y un manifiesto que describe el paquete y sus dependencias. El paquete es la unidad de despliegue de una aplicación. El producto eXtreme Scale da soporte a los siguientes tipos de paquete:

Paquete de servidor
El paquete de servidor es el archivo `objectgrid.jar`, se instala con la instalación de servidor autónomo de eXtreme Scale, es necesario para ejecutar servidores eXtreme Scale y también se puede utilizar para ejecutar clientes de eXtreme Scale o cachés locales en memoria. El ID de paquete para el archivo `objectgrid.jar` es com.ibm.websphere.xs.server_`<versión>`, donde la versión tiene el formato: `<Versión>.<Release>.<Modificación>`. Por ejemplo, el paquete de servidor para eXtreme Scale versión 7.1.1 es com.ibm.websphere.xs.server_7.1.1.

Paquete de cliente
El paquete de cliente es el archivo `ogclient.jar`, se instala con las instalaciones autónomas y de cliente de eXtreme Scale y se utiliza para ejecutar clientes de eXtreme Scale o cachés locales en memoria. El ID de paquete para el archivo `ogclient.jar` es com.ibm.websphere.xs.client_`versión`, donde la versión tiene el formato: `<Versión>.<Release>.<Modificación>`. Por ejemplo, el paquete de cliente para eXtreme Scale versión 7.1.1 es com.ibm.websphere.xs.client_7.1.1.

Limitaciones

No puede reiniciar el paquete de eXtreme Scale porque no puede reiniciar el Intermediario para solicitudes de objetos (ORB). Para reiniciar el servidor eXtreme

Scale, debe reiniciar la infraestructura OSGi.

Instalación de la infraestructura OSGi de Eclipse Equinox con Eclipse Gemini para clientes y servidores

Si desea desplegar WebSphere eXtreme Scale en una infraestructura OSGi, debe configurar el entorno de Eclipse Equinox.

Acerca de esta tarea

La tarea requiere que descargue e instale la infraestructura Blueprint, lo que le permite configurar posteriormente JavaBeans y exponerlos como servicios. El uso de los servicios es importante porque puede exponer plug-ins como servicios OSGi de forma que los pueda utilizar el entorno de ejecución de eXtreme Scale. El producto da soporte a dos contenedores blueprint en la infraestructura OSGi principal de Eclipse Equinox: Eclipse Gemini y Apache Aries. Utilice este procedimiento para configurar el contenedor Eclipse Gemini.

Procedimiento

1. Descargue Eclipse Equinox SDK Versión 3.6.1 o posterior del sitio web de Eclipse. Cree un directorio para la infraestructura Equinox, por ejemplo: `/opt/equinox`. Estas instrucciones hacen referencia a este directorio como `raíz_equinox`. Extraiga el archivo comprimido en el directorio `raíz_equinox`.
2. Descargue el archivo comprimido de gemini-blueprint incubation 1.0.0 del sitio web de Eclipse. Extraiga el contenido del archivo en un directorio temporal y copie los siguientes archivos extraídos en el directorio `raíz_equinox/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. Descargue la infraestructura Spring versión 3.0.5 de la siguiente página web de SpringSource: http://www.springsource.com/download/community. Extráigala en un directorio temporal y copie los siguientes archivos extraídos en el directorio `raíz_equinox/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. Descargue el archivo de archivado Java archive (JAR) de AOP Alliance de la página web de SpringSource. Copie el archivo com.springsource.org.aopalliance-1.0.0.jar en el directorio `raíz_equinox/plugins`.
5. Descargue el archivo JAR de Apache Commons Logging 1.1.1 de la página web de SpringSource. Copie el archivo `com.springsource.org.apache.commons.logging-1.1.1.jar` en el directorio `raíz_equinox/plugins`.
6. Descargue el cliente de línea de mandatos de Luminis OSGi Configuration Admin. Utilice este paquete para gestionar las configuraciones administrativas de OSGi. Puede descargar el archivo JAR de la siguiente página web: https://opensource.luminis.net/wiki/display/SITE/OSGi+Configuration+Admin+command+line+client. Copie el archivo `net.luminis.cmc-0.2.5.jar` en el directorio `raíz_equinox/plugins`.

7. Descargue el paquete de instalación de archivos de la Versión 3.0.2 de Apache Felix de la siguiente página web: http://felix.apache.org/site/index.html. Copie el archivo `org.apache.felix.fileinstall-3.0.2.jar` en el directorio `raíz_equinox/plugins`.
8. Cree un directorio de configuración en el directorio `equinox_root/plugins`, por ejemplo:

   ```
   mkdir equinox_root/plugins/configuration
   ```

9. Cree el archivo `config.ini` siguiente en el directorio `equinox_root/plugins/configuration`, sustituyendo *equinox_root* por la vía de acceso absoluta al directorio `equinox_root` y eliminando todos los espacios de cola después de la barra inclinada invertida de cada línea. Debe incluir una línea en blanco al final del archivo; por ejemplo:

   ```
   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
   ```

   Si ya ha configurado el entorno, puede limpiar el repositorio de plug-ins de Equinox eliminando el directorio siguiente: `raíz_equinox\plugins\configuration\org.eclipse.osgi`.
10. Ejecute los mandatos siguientes para iniciar la consola de equinox.

    Si está ejecutando una versión distinta de Equinox, el nombre de archivo JAR será distinto al del ejemplo siguiente:

    ```
    java -jar plugins\org.eclipse.osgi_3.6.1.R36x_v20100806.jar -console
    ```

Instalación de paquetes de eXtreme Scale

WebSphere eXtreme Scale incluye paquetes que se pueden instalar en una infraestructura OSGi de Eclipse Equinox. Estos paquetes son necesarios para iniciar los servidores eXtreme Scale o utilizar clientes de eXtreme Scale en OSGi.

Antes de empezar

En esta tarea se supone que se han instalado los productos siguientes:

- Infraestructura OSGi de Eclipse Equinox
- Cliente o servidor autónomo de eXtreme Scale

Acerca de esta tarea

eXtreme Scale incluye dos paquetes. Sólo se necesita uno de los paquetes siguientes en una infraestructura OSGi:

objectgrid.jar

El paquete de servidor es el archivo `objectgrid.jar` que se instala con la instalación de servidor autónomo de eXtreme Scale, es necesario para ejecutar servidores eXtreme Scale y también se puede utilizar para ejecutar clientes eXtreme Scale o cachés locales en memoria. El ID de paquete para el archivo `objectgrid.jar` es com.ibm.websphere.xs.server_<versión>,

donde la versión tiene el formato: <Versión>.<Release>.<Modificación>. Por ejemplo, el paquete de servidor para eXtreme Scale versión 7.1.1 es com.ibm.websphere.xs.server_7.1.1.

ogclient.jar
El paquete ogclient.jar se instala con las instalaciones autónomas y de cliente de eXtreme Scale y se utiliza para ejecutar clientes de eXtreme Scale o cachés locales en memoria. El ID de paquete para el archivo ogclient.jar es com.ibm.websphere.xs.client_<versión>, donde la versión está en el formato: <Versión>_<Release>_<Modificación. Por ejemplo, el paquete de cliente para eXtreme Scale Versión 7.1.1 es com.ibm.websphere.xs.client_7.1.1.

Para obtener más información sobre el desarrollo de plug-ins de eXtreme Scale, consulte el tema Plug-ins y API del sistema.

Procedimiento

Para instalar el paquete de cliente o servidor de eXtreme Scale en la infraestructura OSGi de Eclipse Equinox utilizando la consola de OSGi:

1. Inicie la infraestructura de Eclipse Equinox con la consola habilitada; por ejemplo:
   ```
   inicio_java/bin/java -jar <raíz_equinox>/plugins/
   org.eclipse.osgi_3.6.1.R36x_v20100806.jar -console
   ```
2. Instale el paquete de cliente o servidor de eXtreme Scale en la consola de Equinox:
   ```
   osgi> install file:///<vía_acceso_archivo>
   ```
3. Equinox visualiza el ID de paquete para el paquete recién instalado:
   ```
   El ID de paquete es 25
   ```
4. Inicie el paquete en la consola de Equinox, donde <id> es el ID de paquete asignado al instalar el paquete:
   ```
   osgi>  start <id>
   ```
5. Recupere el estado de servicio en la consola de Equinox para verificar que el paquete se ha iniciado; por ejemplo:
   ```
   osgi> ss
   ```
   Cuando el paquete se ha iniciado satisfactoriamente, visualiza el estado ACTIVO; por ejemplo:
   ```
   25      ACTIVE      com.ibm.websphere.xs.server_7.1.1
   ```

Instale el paquete de cliente o servidor de eXtreme Scale en la infraestructura OSGi de Eclipse Equinox utilizando el archivo config.ini:

6. Copie el paquete de cliente o servidor de eXtreme Scale (objectgrid.jar o ogclient.jar) del directorio <raíz_instalación_wxs>/ObjectGrid/lib en el directorio de plug-ins de Eclipse Equinox; por ejemplo: <raíz_equinox>/plugins
7. Edite el archivo de configuración config.ini de Eclipse Equinox y añada el paquete a la propiedad osgi.bundles; por ejemplo:
   ```
   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 que haya una línea en blanco después del último nombre de paquete. Cada paquete está separado por una coma.

8. Inicie la infraestructura de Eclipse Equinox con la consola habilitada; por ejemplo:

   ```
   inicio_java/bin/java -jar <raíz_equinox>/plugins/
   org.eclipse.osgi_3.6.1.R36x_v20100806.jar -console
   ```

9. Recupere el estado de servicio en la consola de Equinox para verificar que el paquete se ha iniciado:

   ```
   osgi> ss
   ```

   Cuando el paquete se haya iniciado satisfactoriamente, visualizará el estado ACTIVO; por ejemplo:

   ```
   25      ACTIVE      com.ibm.websphere.xs.server_7.1.1
   ```

Resultados

El paquete de servidor o cliente de eXtreme Scale se ha instalado e iniciado en la infraestructura OSGi de Eclipse Equinox.

Creación y ejecución de plug-ins dinámicos de eXtreme Scale para su uso en un entorno OSGi

Todos los plug-ins de eXtreme Scale se pueden configurar para un entorno OSGi. La principal ventaja de los plug-ins dinámicos es que le permiten actualizarlos sin concluir la cuadrícula. Esto le permite desarrollar una aplicación sin reiniciar los procesos del contenedor de cuadrícula.

Acerca de esta tarea

El soporte OSGi de WebSphere eXtreme Scale le permite desplegar el producto en una infraestructura OSGi como por ejemplo Eclipse Equinox. Anteriormente, si deseaba actualizar los plug-ins utilizados por eXtreme Scale, tenía que reiniciar la máquina virtual Java (JVM) para aplicar las nuevas versiones de los plug-ins. Con el soporte de plug-ins dinámicos proporcionado por eXtreme Scale y la capacidad de actualizar paquetes que proporciona la infraestructura OSGi, ahora puede actualizar las clases de plug-in sin reiniciar la JVM. Estos plug-ins los exportan los *paquetes* como servicios. WebSphere eXtreme Scale accede al servicio consultando el registro OSGi. En la plataforma de servicio OSGi, un paquete es un archivo de archivado Java (JAR) que contiene código Java, recursos y un manifiesto que describe el paquete y sus dependencias. El paquete es la unidad de despliegue de una aplicación.

Procedimiento

1. Crear plug-ins dinámicos de eXtreme Scale.
2. Configurar plug-ins de eXtreme Scale con OSGi Blueprint.
3. Instalar e iniciar plug-ins habilitados para OSGi.

Creación de plug-ins dinámicos de eXtreme Scale

WebSphere eXtreme Scale incluye los plug-ins ObjectGrid y BackingMap. Estos plug-ins se implementan en Java y se configuran utilizando el archivo XML de descriptor de ObjectGrid. Para crear un plug-in dinámico que se pueda actualizar dinámicamente, es necesario estar al corriente de los sucesos de ciclo de vida de ObjectGrid y BackingMap porque es posible que sea necesario completar algunas acciones durante la actualización. La ampliación de un paquete de plug-in con

métodos de devolución de llamada de ciclo de vida, escuchas de sucesos, o ambos, permite al plug-in completar estas acciones en los momentos adecuados.

Antes de empezar

En este tema se supone que ha creado el plug-in apropiado. Para obtener más información sobre el desarrollo de plug-ins de eXtreme Scale, consulte el tema Plug-ins y API del sistema.

Acerca de esta tarea

Todos los plug-ins de eXtreme Scale se aplican a una instancia BackingMap u ObjectGrid. Muchos plug-ins también interactúan con otros plug-ins. Por ejemplo, un cargador y un plug-in TransactionCallback trabajan juntos para interactuar correctamente con una transacción de base de datos y las diversas llamadas JDBC de base de datos. Es posible que algunos plug-ins requieran también que se almacenen en la memoria caché datos de configuración de otros plug-ins a fin de mejorar el rendimiento.

Los plug-ins BackingMapLifecycleListener y ObjectGridLifecycleListener proporcionan operaciones de ciclo de vida para las instancias BackingMap y ObjectGrid respectivas. Este proceso permite notificar a los plug-ins cuando es posible que se cambien la BackingMap o la ObjectGrid padre y sus respectivos plug-ins. Los plug-ins BackingMap implementan la interfaz BackingMapLifecyleListener y los plug-ins ObjectGrid implementan la interfaz ObjectGridLifecycleListener. Estos plug-ins se invocan automáticamente cuando cambia el ciclo de vida de la BackingMap o ObjectGrid padre. Para obtener más información sobre los plug-ins de ciclo de vida, consulte el tema Gestión de ciclos de vida de plug-ins.

Puede esperar ampliar los paquetes utilizando los métodos de ciclo de vida o escuchas de suceso en las siguientes tareas comunes:

- Inicio y detención de recursos, como por ejemplo hebras o suscriptores de mensajería.
- Si se especifica que se produzca una notificación cuando los plug-ins de igual se actualicen, lo que permite acceso directo al plug-in y la detección de los cambios.

Siempre que acceda a otro plug-in directamente, acceda a ese plug-in mediante el contenedor OSGi para asegurarse de que todas las partes del sistema hagan referencia al plug-in correcto. Si, por ejemplo, algún componente de la aplicación almacena en la memoria caché o hace referencia directamente a una instancia de un plug-in, mantendrá su referencia a esa versión del plug-in, incluso después de que el plug-in se haya actualizado dinámicamente. Este comportamiento puede causar problemas relacionados con la aplicación así como fugas de memoria. Por consiguiente, escriba código que dependa de plug-ins dinámicos que obtienen la referencia utilizando la semántica OSGi, getService(). Si la aplicación debe almacenar en memoria caché uno o varios plug-ins, escucha los sucesos de ciclo de vida utilizando las interfaces ObjectGridLifecycleListener y BackingMapLifecycleListener. La aplicación debe poder renovar también su memoria caché cuando sea necesario, en modalidad de seguridad de hebra.

Todos los plug-ins de eXtreme Scale utilizados con OSGi también deben implementar las interfaces BackingMapPlugin u ObjectGridPlugin respectivas. Los plug-ins nuevos, como la interfaz MapSerializerPlugin, imponen esta práctica.

Estas interfaces proporcionan al entorno de ejecución de eXtreme Scale y a OSGi una interfaz coherente para inyectar el estado en el plug-in y controlar su ciclo de vida.

Utilice esta tarea para especificar que se produzca una notificación cuando se actualicen plug-ins de igual. Puede crear una fábrica de escuchas que genere una instancia de escucha.

Procedimiento

- Actualice la clase de plug-in ObjectGrid para implementar la interfaz ObjectGridPlugin. Esta interfaz incluye métodos que permiten a eXtreme Scale inicializar, establecer la instancia de ObjectGrid y destruir el plug-in. Consulte el siguiente código de ejemplo:

```
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() {
        // Manejar la inicialización de plug-in aquí. Lo llama
        // eXtreme Scale y no el gestor de beans OSGi.
        state = State.INITIALIZED;
    }
    boolean isInitialized() {
        return state == State.INITIALIZED;
    }

    public void destroy() {
        // Destruir el plug-in y liberar los recursos. A éste
        // lo puede llamar el gestor de beans OSGi o eXtreme Scale.
        state = State.DESTROYED;
    }

    public boolean isDestroyed() {
        return state == State.DESTROYED;
    }
}
```

- Actualice la clase de plug-in ObjectGrid para implementar la interfaz ObjectGridLifecycleListener. Consulte el siguiente código de ejemplo:

```
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:
                  // Buscar un cargador o MapSerializerPlugin utilizando
                  // OSGi o directamente desde la instancia de ObjectGrid.
                  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;
            }
        }
        ...
    }
```

- Actualice un plug-in BackingMap. Actualice la clase de plug-in BackingMap para implementar la interfaz de plug-in BackingMap. Esta interfaz incluye métodos que permiten a eXtreme Scale inicializar, establecer la instancia de BackingMap y destruir el plug-in. Consulte el siguiente código de ejemplo:

```
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() {
        // Manejar la inicialización de plug-in aquí. Lo llama
        // eXtreme Scale y no el gestor de beans OSGi.
        state = State.INITIALIZED;
    }
    boolean isInitialized() {
        return state == State.INITIALIZED;
    }

    public void destroy() {
        // Destruir el plug-in y liberar los recursos. A éste
        // lo puede llamar el gestor de beans OSGi o eXtreme Scale.
        state = State.DESTROYED;
    }

    public boolean isDestroyed() {
        return state == State.DESTROYED;
    }
}
```

- Actualice la clase de plug-in BackingMap para implementar la interfaz BackingMapLifecycleListener. Consulte el siguiente código de ejemplo:

```
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:
                  // Buscar un MapSerializerPlugin utilizando
                  // OSGi o directamente desde la instancia de ObjectGrid.
                  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

Al implementar la interfaz ObjectGridPlugin o BackingMapPlugin, eXtreme Scale puede controlar el ciclo de vida del plug-in en los momentos correctos.

Al implementar la interfaz ObjectGridLifecycleListener o BackingMapLifecycleListener, el plug-in se registra automáticamente como escucha de los sucesos de ciclo de vida ObjectGrid o BackingMap asociados. El suceso INITIALIZING se utiliza para señalar que todos los plug-ins ObjectGrid y BackingMap se han inicializado y están disponibles para buscarse y utilizarse. El suceso ONLINE se utiliza para señalar que el ObjectGrid está en línea y listo para iniciar el proceso de sucesos.

Configuración de plug-ins de eXtreme Scale con OSGi Blueprint

Todos los plug-ins de eXtreme Scale ObjectGrid y BackingMap se pueden definir como servicios y beans OSGi utilizando el servicio OSGi Blueprint disponible con Eclipse Gemini o Apache Aries.

Antes de empezar

Antes de configurar los plug-ins como servicios OSGi, primero debe empaquetar los plug-ins en un paquete OSGi y conocer los principios fundamentales de los plug-ins necesarios. El paquete debe importar los paquetes de servidor o cliente de WebSphere eXtreme Scale y otros paquetes dependientes necesarios para los plug-ins o crear una dependencia de paquete en los paquetes de servidor o cliente de eXtreme Scale. Este tema describe cómo configurar el XML de Blueprint para crear beans de plug-ins y exponerlos como servicios OSGi para que eXtreme Scale los utilice.

Acerca de esta tarea

Los beans y servicios están definidos en un archivo XML Blueprint y el contenedor Blueprint descubre, crea y conecta los beans entre ellos y los expone como servicios. El proceso deja los beans disponibles para otros paquetes OSGi, incluidos los paquetes de servidor y cliente de eXtreme Scale.

Al crear servicios de plug-in personalizados para utilizarlos con eXtreme Scale, el paquete que va a alojar los plug-ins, debe estar configurado para utilizar Blueprint. Además, se debe crear y almacenar un archivo XML Blueprint dentro del paquete. Para obtener una visión general de la especificación, lea la información sobre la creación de aplicaciones OSGi con la especificación de contenedor Blueprint.

Procedimiento

1. Cree un archivo XML Blueprint. Puede utilizar el nombre que desee para el archivo. No obstante, debe incluir el espacio de nombre blueprint:

```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">
...
</blueprint>
```

2. Cree definiciones de bean en el archivo XML Blueprint para cada plug-in de eXtreme Scale.

   Los beans se definen utilizando el elemento `<bean>`, se pueden conectar a otras referencias de bean y pueden incluir parámetros de inicialización.

   Importante: Al definir un bean, debe utilizar el ámbito correcto. Blueprint soporta los ámbitos de singleton y prototipo. eXtreme Scale también soporta un ámbito de fragmento personalizado.

   Defina la mayoría de los plug-ins de eXtreme Scale como beans de ámbito de fragmento o prototipo, ya que todos los beans deben ser exclusivos para cada fragmento ObjectGrid o instancia de BackingMap con los que estén asociados. Los beans de ámbito de fragmento pueden ser útiles cuando se utilizan los beans en otros contextos para permitir recuperar la instancia correcta.

   Para definir un bean de ámbito de prototipo, utilice el atributo `scope="prototype"` en el bean:

```
<bean id="myPluginBean" class="com.mycompany.MyBean" scope="prototype">
...
</bean>
```

   Para definir un bean de ámbito de fragmento, debe añadir el espacio de nombres `objectgrid` al esquema XML y utilizar el atributo `scope="objectgrid:shard"` en el 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. Cree definiciones de bean PluginServiceFactory para cada bean de plug-in. Todos los beans de eXtreme Scale deben tener un bean PluginServiceFactory definido para que se pueda aplicar el ámbito de bean correcto. eXtreme Scale incluye un BlueprintServiceFactory que se puede utilizar. Incluye dos propiedades que se deben establecer. Debe establecer la propiedad blueprintContainer en la referencia `blueprintContainer` y la propiedad beanId se debe establecer en el nombre de identificador de bean. Cuando eXtreme Scale busca el servicio para instanciar los beans adecuados, el servidor busca la instancia de componente de bean utilizando el contenedor Blueprint.

```
bean id="myPluginBeanFactory"
    class="com.ibm.websphere.objectgrid.plugins.osgi.BluePrintServiceFactory">
      <property name="blueprintContainer" ref="blueprintContainer"/>
<property name="beanId" value="myPluginBean" />
</bean>
```

4. Crear un administrador de servicios para cada bean PluginServiceFactory. Cada administrador de servicios expone el bean PluginServiceFactory, utilizando el elemento `<service>`. El elemento de servicio identifica el nombre a exponer en OSGi, la referencia al bean PluginServiceFactory, la interfaz a exponer y la

clasificación del servicio. eXtreme Scale utiliza la clasificación de administrador de servicios para realizar actualizaciones de servicio cuando la cuadrícula de eXtreme Scale está activa. Si no se especifica la clasificación, la infraestructura OSGi supone una clasificación de 0. Lea la información sobre la actualización de clasificaciones de servicio para obtener más información.

Blueprint incluye varias opciones para configurar administradores de servicios. Para definir un administrador de servicios simple para un bean PluginServiceFactory, cree un elemento `<service>` para cada bean PluginServiceFactory:

```
<service ref="myPluginBeanFactory"
    interface="com.ibm.websphere.objectgrid.plugins.osgi.PluginServiceFactory"
    ranking="1">
</service>
```

5. Almacene el archivo XML Blueprint en el paquete de plug-ins. El archivo XML Blueprint debe almacenarse en el directorio `OSGI-INF/blueprint` para que se descubra el contenedor Blueprint.

   Para almacenar el archivo XML Blueprint en un directorio diferente, debe especificar la siguiente cabecera de manifiesto Bundle-Blueprint:

   ```
   Bundle-Blueprint: OSGI-INF/blueprint.xml
   ```

Resultados

Los plug-ins de eXtreme Scale están ahora configurados para exponerse en un contenedor OSGi Blueprint. Además, el archivo XML de descriptor ObjectGrid está configurado para hacer referencia a los plug-ins utilizando el servicio OSGi Blueprint.

Instalación e inicio de plug-ins habilitados para OSGi

En esta tarea, instalará el paquete de plug-in dinámico en la infraestructura OSGi. A continuación, iniciará el plug-in.

Antes de empezar

En este tema se supone que se han completado las tareas siguientes:

- Se ha instalado el paquete de servidor o cliente de eXtreme Scale en la infraestructura OSGi de Eclipse Equinox. Consulte “Instalación de paquetes de eXtreme Scale” en la página 184.
- Se han implementado uno o varios plug-ins dinámicos de BackingMap u ObjectGrid. Consulte “Creación de plug-ins dinámicos de eXtreme Scale” en la página 186.
- Los plug-ins dinámicos se han empaquetado como servicios OSGi en paquetes OSGi.

Acerca de esta tarea

Esta tarea describe cómo instalar el paquete utilizando la consola Eclipse Equinox. El paquete se puede instalar utilizando varios métodos diferentes, incluida la modificación del archivo de configuración `config.ini`. Los productos que incorporan Eclipse Equinox incluyen métodos alternativos para gestionar paquetes. Para obtener más información sobre cómo añadir paquetes en el archivo `config.ini` de Eclipse Equinox, consulte los opciones de ejecución de Eclipse.

OSGi permite que se inicien paquetes que tienen servicios duplicados. WebSphere eXtreme Scale utiliza la clasificación de servicios más reciente. Al iniciar varias infraestructuras OSGi en una cuadrícula de datos de eXtreme Scale, debe

asegurarse de que se inician las clasificaciones de servicio correctas en cada servidor. Si no es así, la cuadrícula se inicia con una mezcla de versiones diferentes.

Para ver qué versiones están siendo utilizadas por la cuadrícula de datos, utilice el programa de utilidad xscmd para comprobar las clasificaciones actuales y disponibles. Para obtener más información sobre las clasificaciones de servicio disponibles, consulte Actualización de servicios OSGi para plug-ins de eXtreme Scale con **xscmd**.

Procedimiento

Instalar el paquete de plug-in en la infraestructura OSGi de Eclipse Equinox utilizando la consola OSGi.

1. Inicie la infraestructura de Eclipse Equinox con la consola habilitada; por ejemplo:

   ```
   <inicio_java>/bin/java -jar <raíz_equinox>/plugins/org.eclipse.osgi_3.6.1.R36x_v20100806.jar -console
   ```

2. Instale el paquete de plug-in en la consola de Equinox.

   ```
   osgi> install file:///<vía_acceso_archivo>
   ```

   Equinox visualiza el ID de paquete para el paquete recién instalado:

   ```
   Bundle id is 17
   ```

3. Entre la línea siguiente para iniciar el paquete en la consola de Equinox, donde `<id>` es el ID de paquete asignado al instalar el paquete:

   ```
   osgi>  install <id>
   ```

4. Recupere el estado de servicio en la consola de Equinox para verificar que el paquete se ha iniciado:

   ```
   osgi> ss
   ```

   Cuando el paquete se ha iniciado satisfactoriamente, visualiza el estado ACTIVO; por ejemplo:

   ```
   17      ACTIVE      com.mycompany.plugin.bundle_VRM
   ```

Instalar el paquete de plug-in en la infraestructura OSGi de Eclipse Equonix utilizando el archivo config.ini.

5. Copie el paquete de plug-in en el directorio de plug-ins de Eclipse Equinox; por ejemplo:

   ```
   <raíz_equinox>/plugins
   ```

6. Edite el archivo de configuración `config.ini` de Eclipse Equinox y añada el paquete a la propiedad osgi.bundles; por ejemplo:

   ```
   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 que haya una línea en blanco después del último nombre de paquete. Cada paquete está separado por una coma.

7. Inicie la infraestructura de Eclipse Equinox con la consola habilitada; por ejemplo:

   ```
   <inicio_java>/bin/java -jar <raíz_equinox>/plugins/org.eclipse.osgi_3.6.1.R36x_v20100806.jar -console
   ```

8. Recupere el estado de servicio en la consola de Equinox para verificar que el paquete se ha iniciado; por ejemplo:

```
osgi> ss
```

Cuando el paquete se ha iniciado satisfactoriamente, visualiza el estado ACTIVO; por ejemplo:

```
17      ACTIVE      com.mycompany.plugin.bundle_VRM
```

Resultados

El paquete de plug-in ya está instalado e iniciado. Ahora ya se puede iniciar el contenedor o cliente de eXtreme Scale. Para obtener más información sobre el desarrollo de plug-ins de eXtreme Scale, consulte el tema Plug-ins y API del sistema.

Ejecución de contenedores de eXtreme Scale con plug-ins dinámicos en un entorno OSGi

Si la aplicación se aloja en la infraestructura OSGi de Eclipse Equinox con Eclipse Gemini o Apache Aries, puede utilizar esta tarea para ayudar a instalar y configurar la aplicación WebSphere eXtreme Scale en OSGi.

Antes de empezar

Antes de iniciar esta tarea, asegúrese de completar las tareas siguientes:

- Instalar la infraestructura OSGi de Eclipse Equinox con Eclipse Gemini
- Crear y ejecutar plug-ins dinámicos de eXtreme Scale para utilizarlos en un entorno OSGi

Acerca de esta tarea

Con los plug-ins dinámicos, puede actualizar dinámicamente el plug-in mientras la cuadrícula sigue activa. Esto le permite actualizar una aplicación sin reiniciar los procesos del contenedor de cuadrícula. Para obtener más información sobre el desarrollo de plug-ins de eXtreme Scale, consulte Plug-ins y API del sistema.

Procedimiento

1. Configure los plug-ins habilitados para OSGi utilizando el archivo XML de descriptor ObjectGrid.
2. Inicie los servidores de contenedor eXtreme Scale utilizando la infraestructura OSGi de Eclipse Equinox.
3. Administre los servicios OSGi para los plug-ins eXtreme Scale con el programa de utilidad xscmd.
4. Configure servdires con OSGi Blueprint.

Configuración de plug-ins habilitados para OSGi mediante el archivo XML de descriptor de ObjectGrid

En esta tarea, se añaden servicios OSGi existentes a un archivo XML de descriptor para que el contenedor de WebSphere eXtreme Scale pueda reconocer y cargar correctamente los plug-ins habilitados para OSGi.

Antes de empezar

Para configurar los plug-ins, asegúrese de:

- Crear el paquete y habilitar plug-ins dinámicos para despliegue OSGi.

- Tener los nombres de los servicios OSGi que representan los plug-ins disponibles.

Acerca de esta tarea

Ha creado un servicio OSGi para recortar los plug-in. Ahora, estos servicios deben definirse en el archivo `objectgrid.xml` para que los contenedores de eXtreme Scale pueden cargar y configurar el plug-in o los plug-ins correctamente.

Procedimiento

1. Cualquier plug-in específico de cuadrícula, por ejemplo TransactionCallback, debe especificarse en el elemento objectGrid. Consulte el ejemplo siguiente del archivo `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: El valor de atributo `osgiService` debe coincidir con el valor de atributo `ref` que se especifica en el archivo XML `blueprint`, donde se ha definido el servicio para myTranCallback PluginServiceFactory.

2. Cualquier plug-in específico de correlación, por ejemplo los cargadores o serializadores, se debe especificar en el elemento backingMapPluginCollections y se debe hacer referencia a él desde el elemento backingMap. Consulte el ejemplo siguiente del archivo `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

El archivo `objectgrid.xml` de este ejemplo indica a eXtreme Scale que cree una cuadrícula denominada `MyGrid` con dos correlaciones, `MyMap1` y `MyMap2`. La correlación `MyMap1` utiliza el serializador recortado por el servicio OSGi, mySerializerFactory. La correlación `MyMap2` utiliza un serializador del servicio OSGi, myOtherSerializerFactory, y un cargador del servicio OSGi, myLoader.

Inicio de servidores eXtreme Scale utilizando la infraestructura OSGi de Eclipse Equinox

Los servidores de contenedor de WebSphere eXtreme Scale se pueden iniciar en una infraestructura OSGi de Eclipse Equinox utilizando varios métodos.

Antes de empezar

Para poder iniciar un contenedor eXtreme Scale, debe haber completado las siguientes tareas:

1. El paquete de servidor de WebSphere eXtreme Scale debe estar instalado en Eclipse Equinox.
2. La aplicación debe estar empaquetado como un paquete OSGi.
3. Los plug-ins de WebSphere eXtreme Scale (si existen) deben estar empaquetados como un paquete OSGi. Pueden estar empaquetados en el mismo paquete que la aplicación o como paquetes independientes.

Acerca de esta tarea

Esta tarea describe cómo iniciar un servidor de contenedor eXtreme Scale en una infraestructura OSGi de Eclipse Equinox. Puede utilizar cualquiera de los métodos siguientes para iniciar los servidores de contenedor utilizando la implementación de Eclipse Equinox:

- Servicio OSGi Blueprint

  Puede incluir toda la configuración y los metadatos en un paquete OSGi. Consulte la imagen siguiente para comprender el proceso de Eclipse Equinox para este método:

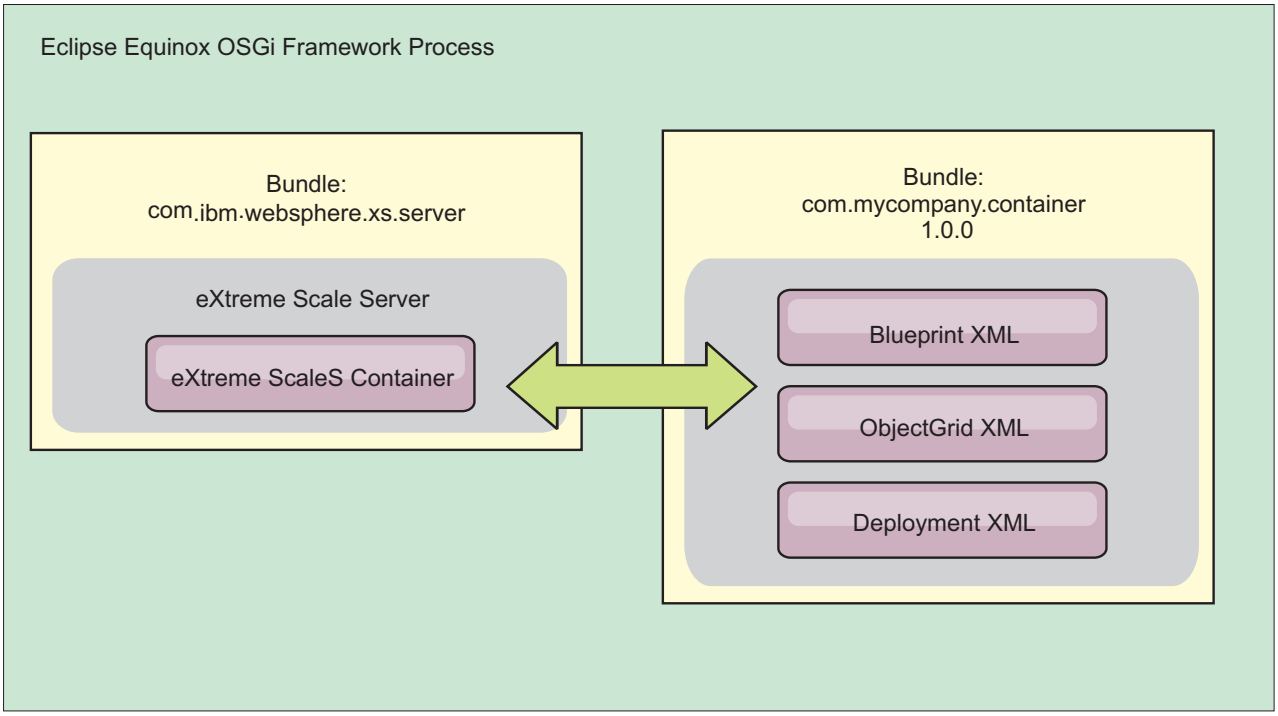


Figura 56. Proceso de Eclipse Equinox para incluir toda la configuración y los metadatos en un paquete OSGi

- Servicio de administración de configuración OSGi

  Puede especificar la configuración y los metadatos fuera de un paquete OSGi. Consulte la imagen siguiente para comprender el proceso de Eclipse Equinox para este método:

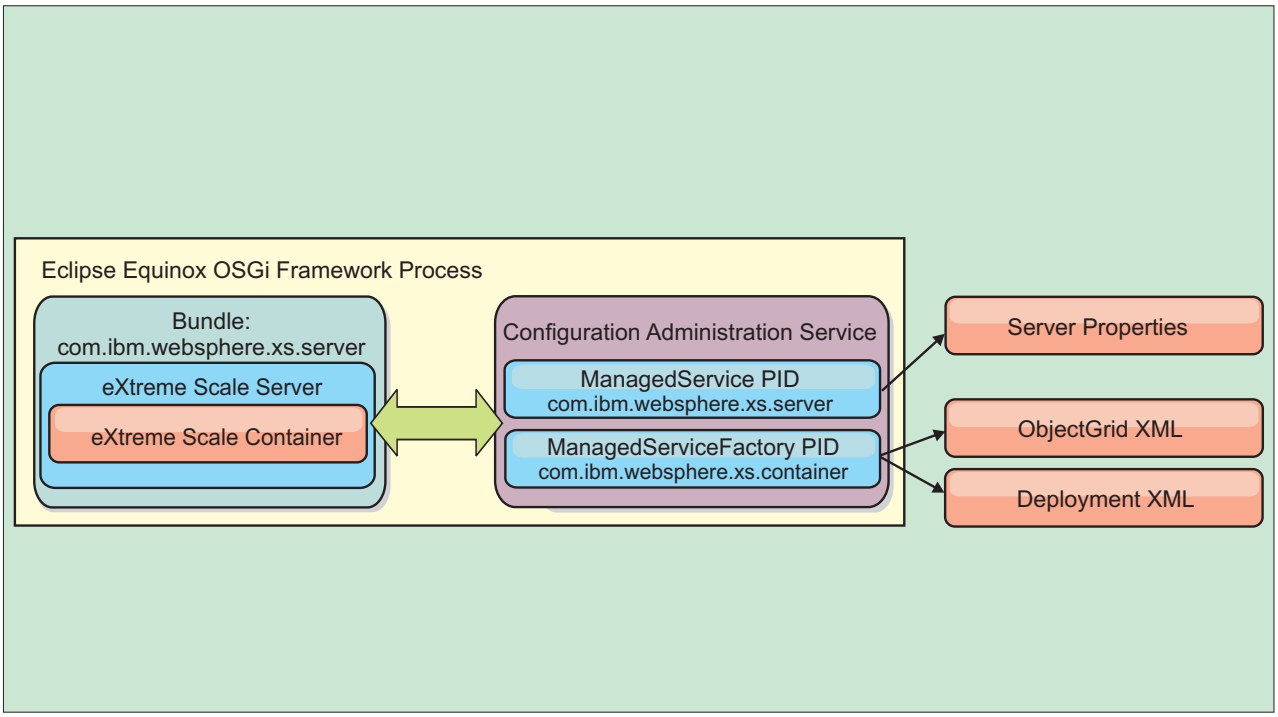


Figura 57. Proceso de Eclipse Equinox para especificar la configuración y los metadatos fuera de un paquete OSGi

- A través de programas
  Soporta soluciones de configuración personalizadas.

En cada caso, se configura un singleton de servidor eXtreme Scale y se configuran uno o varios contenedores.

El paquete de servidor eXtreme Scale, `objectgrid.jar`, incluye todas las bibliotecas necesarias para iniciar y ejecutar un contenedor de cuadrícula de eXtreme Scale en una infraestructura OSGi. El entorno de ejecución de servidor se comunica con los objetos de datos y los plug-ins proporcionados por el usuario utilizando el administrador de servicios OSGi.

Importante: Después de que un paquete de servidor eXtreme Scale se haya iniciado y el servidor eXtreme Scale se haya inicializado, no se puede reiniciar. Se debe reiniciar el proceso de Eclipse Equinox para reiniciar un servidor eXtreme Scale.

Puede utilizar el soporte de eXtreme Scale para el espacio de nombres Spring para configurar los servidores de contenedor de eXtreme Scale en un archivo XML Blueprint. Cuando se añaden los elementos XML de servidor y contenedor al archivo XML Blueprint, el manejador de espacio de nombres de eXtreme Scale inicia automáticamente un servidor de contenedor utilizando los parámetros definidos en el archivo XML Blueprint cuando se inicia el paquete. El manejador detiene el contenedor cuando se detiene el paquete.

Para configurar servidores de contenedor de eXtreme Scale con XML Blueprint, realice los pasos siguientes:

Procedimiento

- Inicie un servidor de contenedor de eXtreme Scale utilizando OSGi Blueprint.
  1. Cree un paquete de contenedor.
  2. Instale el paquete de contenedor en la infraestructura OSGi de Eclipse Equinox. Consulte “Instalación e inicio de plug-ins habilitados para OSGi” en la página 192.
  3. Inicie el paquete de contenedor.
- Inicie un servidor de contenedor de eXtreme Scale utilizando la administración de configuración de OSGi.
  1. Configure el servidor y el contenedor utilizando la administración de configuración.
  2. Cuando el paquete de servidor de eXtreme Scale se ha iniciado o los identificadores persistentes se crean con la administración de configuración, el servidor y el contenedor se inician automáticamente.
- Inicie un servidor de contenedor de eXtreme Scale utilizando la API ServerFactory. Consulte la documentación de API de servidor.
  1. Cree una clase de activador de paquete OSGi y utilice la API ServerFactory de eXtreme Scale para iniciar un servidor.

Administración de servicios habilitado para OSGi utilizando el programa de utilidad `xscmd`

Puede utilizar el programa de utilidad **`xscmd`** para completar las tareas de administrador, por ejemplo ver los servicios y sus clasificaciones que están siendo utilizados por cada contenedor y actualizar el entorno de ejecución para utilizar las nuevas versiones de los paquetes.

Acerca de esta tarea

Con la infraestructura OSGi de Eclipse Equinox, puede instalar varias versiones del mismo paquete y puede actualizar esos paquetes durante la ejecución. WebSphere eXtreme Scale es un entorno distribuido que ejecuta los servidores de contenedor en muchas instancias de infraestructura OSGi.

Los administradores son responsables de copiar, instalar e iniciar manualmente paquetes en la infraestructura OSGi. eXtreme Scale incluye un ServiceTrackerCustomizer OSGi para realizar un seguimiento de los servicios que se han identificado como plug-ins de eXtreme Scale en el archivo XML de descriptor de ObjectGrid. Utilice el programa de utilidad **xscmd** para validar qué versión del plug-in se utiliza, qué versiones están disponibles para utilizarse y para realizar actualizaciones de paquete.

eXtreme Scale utiliza el número de clasificación de servicio para identificar la versión de cada servicio. Cuando se cargan dos o más servicios con la misma referencia, eXtreme Scale utiliza automáticamente el servicio con la clasificación más alta.

Procedimiento

- Ejecute el mandato **osgiCurrent** y verifique que cada servidor de eXtreme Scale utiliza la clasificación de servicio de plug-in correcta.

  Dado que eXtreme Scale elige automáticamente la referencia de servicio con la clasificación más alta, es posible que la cuadrícula de datos empiece con varias clasificaciones de un servicio de plug-in.

  Si el mandato detecta una discrepancia de clasificaciones o si no puede encontrar un servicio, se establece un nivel de error distinto de cero. Si el mandato se ha completado satisfactoriamente, el nivel de error se establece en 0.

  El siguiente ejemplo muestra la salida del mandato **osgiCurrent** cuando dos plug-ins se instalan en la misma cuadrícula en cuatro servidores. El plug-in loaderPlugin utiliza la clasificación 1 y txCallbackPlugin utiliza la clasificación 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
```

  El siguiente ejemplo muestra la salida del mandato **osgiCurrent** cuando server2 se ha iniciado con una clasificación más reciente de 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
```

- Ejecute el mandato **osgiAll** para verificar que los servicios de plug-in se han iniciado correctamente en cada servidor de contenedor de eXtreme Scale.

Al iniciar paquetes que contienen servicios a los que una configuración ObjectGrid hace referencia, el entorno de ejecución de eXtreme Scale realiza automáticamente un seguimiento del plug-in, pero no lo utiliza inmediatamente. El mandato **osgiAll** muestra qué plug-ins están disponibles para cada servidor.

Cuando se ejecuta sin parámetros, se muestran todos los servicios para todas las cuadrículas y servidores. Se pueden especificar filtros adicionales, incluido el filtro **-serviceName** <nombre_servicio>, para limitar la salida a un solo servicio o a un subconjunto de la cuadrícula de datos.

El ejemplo siguiente muestra la salida del mandato **osgiAll** cuando se inician dos plug-ins en dos servidores. loaderPlugin tiene las dos clasificaciones 1 y 2 iniciadas y txCallbackPlugin tiene la clasificación 1 iniciada. El mensaje de resumen al final de la salida confirma que ambos servidores ven las mismas clasificaciones de servicio:

```
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.
```

El ejemplo siguiente muestra la salida del mandato **osgiAll** cuando el paquete que incluye loaderPlugin con la clasificación 1 se detiene en server1. El mensaje de resumen en la parte inferior de la salida confirma que en server1 falta ahora loaderPlugin con la clasificación 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
```

El siguiente ejemplo muestra la salida si el nombre de servicio se especifica con el argumento **-sn**, pero el servicio no 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.
```

- Ejecute el mandato **osgiCheck** para comprobar conjuntos de servicios de plug-in y clasificaciones para ver si están disponibles.

  El mandato **osgiCheck** acepta uno o más conjuntos de clasificaciones de servicio en el formato: `-serviceRankings <nombre de servicio>;<clasificación>[,<nombreServicio>;<clasificación>]`

  Cuando las clasificaciones están todas disponibles, el método vuelve con un nivel de error de 0. Si una o más clasificaciones no están disponibles, se establece un nivel de error distinto de cero y una tabla de todos los servidores que no incluyen las clasificaciones de servicio especificadas. Se pueden utilizar filtros adicionales para limitar la comprobación de servicio a un subconjunto de los servidores disponibles en el dominio de eXtreme Scale.

  Por ejemplo, si la clasificación o el servicio especificados están ausentes, se visualiza el siguiente mensaje:

  ```
  Server  OSGi Service Unavailable Rankings
  ------  ------------ --------------------
  server1 loaderPlugin 3
  server2 loaderPlugin 3
  ```

- Ejecute el mandato **osgiUpdate** para actualizar la clasificación de uno o más plug-ins para todos los servidores de una sola ObjectGrid y MapSet en una sola operación.

  El mandato acepta uno o más conjuntos de clasificaciones de servicio con el formato: `-serviceRankings <nombre de servicio>;<clasificación>[,<nombreServicio>;<clasificación>] -g <nombre de cuadrícula> -ms <nombre de conjunto de correlaciones>`

  Con este mandato, puede completar las siguientes operaciones:

  - Verifique que los servicios especificados están disponibles para actualizarse en cada uno de los servidores.
  - Cambie el estado de la cuadrícula a fuera de línea utilizando la interfaz StateManager. Si desea más información, consulte Gestión de la disponibilidad del ObjectGrid. Este proceso inmoviliza la cuadrícula, espera a que se hayan completado las transacciones en ejecución e impide que se inicien transacciones nuevas. Este proceso también señala a los plug-ins ObjectGridLifecycleListener y BackingMapLifecycleListener que interrumpan cualquier actividad transaccional. Consulte Plug-ins para proporcionar escuchas de sucesos para obtener información sobre los plug-ins de escucha de sucesos.
  - Actualice cada contenedor de eXtreme Scale que se ejecuta en una infraestructura OSGi para utilizar las nuevas versiones de servicio.
  - Cambie el estado de la cuadrícula para que esté en línea, lo que permite que continúen las transacciones.

  El proceso de actualización es idempotent, de modo que si un cliente no puede completar ninguna tarea, la operación se retrotrae. Si un cliente no puede realizar la retrotracción o se interrumpe durante el proceso de actualización, se puede emitir el mismo mandato de nuevo y continúa en el paso adecuado.

  Si el cliente no puede continuar y el proceso se reinicia desde otro cliente, utilice la opción `-force` para permitir que el cliente realice la actualización. El mandato **osgiUpdate** impide que varios clientes actualicen el mismo conjunto de correlaciones simultáneamente. Para obtener más detalles sobre el mandato **osgiUpdate**, consulte Actualización de servicios OSGi para plug-ins de eXtreme Scale con **xscmd**.

Configuración de servidores con OSGi Blueprint

Puede configurar servidores de contenedor de WebSphere eXtreme Scale utilizando un archivo XML de OSGi Blueprint, lo que permite simplificar el empaquetado y el desarrollo de paquetes de servidor autocontenidos.

Antes de empezar

En este tema se supone que se han completado las tareas siguientes:

- Se ha instalado e iniciado la infraestructura OSGi de Eclipse Equinox con el contenedor Eclipse Gemini o Apache Aries Blueprint.
- Se ha instalado e iniciado el paquete de servidor de eXtreme Scale.
- Se ha creado el paquete de plug-ins dinámicos de eXtreme Scale.
- Se han creado el archivo XML de política de despliegue y el archivo XML de descriptor de ObjectGrid de eXtreme Scale.

Acerca de esta tarea

Esta tarea describe cómo configurar un servidor de eXtreme Scale con un contenedor utilizando un archivo XML Blueprint. El resultado del procedimiento es paquete de contenedor. Cuando se inicie el paquete de contenedor, el paquete de el servidor eXtreme Scale realizará un seguimiento del paquete, analizará el XML de servidor e iniciará un servidor y contenedor.

Un paquete de contenedor se puede combinar de manera opcional con la aplicación y los plug-ins de eXtreme Scale cuando no son necesarias actualizaciones de plug-in dinámicas o los plug-ins no soportan la actualización dinámica.

Procedimiento

1. Cree un archivo XML Blueprint con el espacio de nombres `objectgrid` incluido. Puede utilizar el nombre que desee para el archivo. No obstante, debe incluir el espacio de nombres blueprint:

```
<?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. Añada la definición XML para el servidor de eXtreme Scale con las propiedades de servidor adecuadas. Consulte el archivo XML de descriptor Spring para obtener detalles sobre todas las propiedades de configuración disponibles. Consulte el ejemplo siguiente de la definición 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. Añadir la definición XML para el contenedor de eXtreme Scale con la referencia a la definición de servidor y a los archivos XML de descriptor ObjectGrid y XML de despliegue ObjectGrid incorporados en el paquete; por ejemplo:

```
<objectgrid:container id="container"
    objectgridxml="/META-INF/objectGrid.xml"
    deploymentxml="/META-INF/objectGridDeployment.xml"
    server="xsServer" />
```

4. Almacene el archivo XML Blueprint en el paquete de contenedor. El XML Blueprint se debe almacenar en el directorio `OSGI-INF/blueprint` para que se encuentre el contenedor Blueprint.

   Para almacenar el archivo XML Blueprint en un directorio diferente, debe especificar la cabecera de manifiesto Bundle-Blueprint; por ejemplo:

   ```
   Bundle-Blueprint: OSGI-INF/blueprint.xml
   ```

5. Empaquete los archivos en un solo archivo JAR de paquete. Consulte el ejemplo siguiente de una jerarquía de directorios de paquete:

   ```
   MyBundle.jar
       /META-INF/manifest.mf
       /META-INF/objectGrid.xml
       /META-INF/objectGridDeployment.xml
       /OSGI-INF/blueprint/blueprint.xml
   ```

Resultados

Ya se ha creado un paquete de contenedor de eXtreme Scale y ahora se puede instalar en Eclipse Equinox. Cuando se inicia el paquete de contenedor, el entorno de ejecución de servidor de eXtreme Scale del paquete de servidor de eXtreme Scale inicia automáticamente el servidor de eXtreme Scale de singleton utilizando los parámetros definidos en el paquete e inicia un servidor de contenedor. El paquete se puede detener e iniciar, lo que hace que el contenedor se detenga y se inicie. El servidor es un singleton y no se detiene cuando el paquete se inicia por primera vez.

Capítulo 4. Ejemplos

Hay disponibles varias guías de aprendizaje, ejemplos y muestras de WebSphere eXtreme Scale.

Ejemplos

Los temas siguientes muestran las principales características de WebSphere eXtreme Scale.

- Ejemplo de la API de DataGrid
- Configuración de despliegues locales

Ejemplos de comunidad

Los ejemplos siguientes muestran cómo utilizar WebSphere eXtreme Scale en distintos entornos para mostrar distintas características del producto.

- **Infraestructura de servicio asíncrono**: la infraestructura de servicio asíncrono proporciona un tejido de proceso escalable y con tolerancia a errores para el proceso asíncrono de mensajes. Para obtener más información, incluido cómo descargar el ejemplo, consulte la Galería de ejemplos: Ejemplo de la infraestructura de servicio asíncrono.
- **Seguridad de la autenticación de cliente**: este ejemplo describe cómo configurar la autenticación que requiere que el cliente proporcione credenciales válidas para el servidor pueda proporcionar acceso a la cuadrícula. Para obtener más información, incluido cómo descargar el ejemplo, consulte la Galería de ejemplos: Seguridad de la autenticación de cliente
- **Creación de correlaciones dinámicas**: este ejemplo muestra cómo crear correlaciones una vez que se ha inicializado ya la cuadrícula. Para eXtreme Scale 7.0 y superior, puede utilizar las plantillas para recuperar correlaciones. Para obtener más información, incluido cómo descargar el ejemplo, consulte la Galería de ejemplos: Creación de correlaciones dinámicas después de la inicialización de la cuadrícula.
- **Réplica multimaestro**: el ejemplo de iniciación a la réplica multimaestro se proporciona para presentar una rápida introducción a la réplica multimaestro (AP). Para obtener más información, incluido cómo descargar el ejemplo, consulte la Galería de ejemplos: Ejemplo de réplica multimaestro.
- **Consultas con la API del gestor de entidades**: este ejemplo muestra cómo utilizar consultas en una correlación particionada distribuida con la API EntityManager. Para obtener más información, incluido cómo descargar el ejemplo, consulte la Galería de ejemplos: Ejecución de consultas en una cuadrícula particionada utilizando la API del gestor de entidades.
- **Consultas paralelas con una implementación ReduceGridAgent**: muestra cómo utilizar la API de Data Grid para ejecutar una consulta en cada partición de la cuadrícula. Para obtener más información, incluido cómo descargar el ejemplo, consulte la Galería de ejemplos: Ejecución de consultas en paralelo utilizando un ReduceGridAgent .

Artículos con guías de aprendizaje y ejemplos

Tabla 9. Artículos disponibles por característica

Artículo	Características
Building grid-ready applications	API ObjectMap, API EntityManager, Consulta, Agentes, Java SE y EE, Estadísticas, Particionamiento, Administración/Operaciones, Eclipse
Scalable grid-style computing and data processing	API EntityManager, Agentes
Building a scalable, resilient, high-performance database alternative	API ObjectMap, Réplica, Particionamiento, Administración/Operaciones, Eclipse
Enhancing xsadmin for WebSphere eXtreme Scale	Administración
Redbook: User's Guide	Todos los temas

Prueba gratuita

Para empezar a utilizar WebSphere eXtreme Scale, descargue una versión de prueba gratuita. Puede desarrollar aplicaciones innovadores y de alto rendimiento ampliando el concepto de almacenamiento en memoria caché de datos utilizando características avanzadas.

Descarga de versión de prueba

Puede descargar una versión de prueba gratuita de WebSphere eXtreme Scale, desde Descargar prueba de eXtreme Scale.

Después de descargar y descomprimir la versión de prueba de eXtreme Scale, vaya al directorio `gettingstarted` y lea el archivo `GETTINGSTARTED_README.txt`. Esta guía de aprendizaje le ayuda a empezar a utilizar eXtreme Scale, crear una cuadrícula en varios servidores y ejecutar algunas aplicaciones sencillas para almacenar y recuperar los datos de una cuadrícula. Antes de desplegar eXtreme Scale en un entorno de producción, hay varias opciones que se deben tener en cuenta, incluidos el número de servidores para utilizar, la cantidad de almacenamiento en cada servidor y la duplicación síncrona o asíncrona.

Archivos de propiedades de ejemplo

Los archivos de propiedades del servidor contienen valores para ejecutar los servidores de catálogo y los servidores de contenedor. Puede especificar un archivo de propiedades de servidor para una configuración autónoma o de WebSphere Application Server. Los archivos de propiedades de cliente contienen valores para el cliente.

Puede utilizar los siguientes archivos de propiedades de ejemplo que están en el directorio *raíz_intal_wxs*`\properties` para crear el archivo de propiedades:

- `sampleServer.properties`
- `sampleClient.properties`

Ejemplo: Programa de utilidad xsadmin

Con el programa de utilidad **xsadmin**, puede formatear y visualizar información textual sobre la topología de WebSphere eXtreme Scale. El programa de utilidad de ejemplo proporciona un método para analizar y descubrir los datos de despliegue actuales, y se puede utilizar como base para crear programas de utilidad personalizados.

Antes de empezar

- 7.1.1+ El programa de utilidad **xsadmin** se proporciona como un ejemplo de cómo puede crear propiedades personalizadas para el despliegue. El programa de utilidad **xscmd** se proporciona como un programa de utilidad soportado para supervisar y administrar el entorno. Para obtener más información, consulte Administración con el programa de utilidad **xscmd**.
- Para que el programa de utilidad **xsadmin** visualice resultados, debe haber creado la topología de la cuadrícula de datos. Los servidores de catálogo y los servidores de contenedor deben estar iniciados. Si desea más información, consulte Inicio y detención de los servidores autónomos.
- Verifique que la variable de entorno *JAVA_HOME* esté establecida para utilizar el entorno de ejecución que se ha instalado con el producto. Si está utilizando la versión de prueba del producto, debe establecer la variable de entorno *JAVA_HOME*.

Acerca de esta tarea

El programa de utilidad de ejemplo **xsadmin** utiliza una implementación de beans gestionados (MBeans). Esta aplicación de supervisión de ejemplo habilita prestaciones de supervisión integradas rápidamente que puede ampliar utilizando las interfaces del paquete com.ibm.websphere.objectgrid.management. Puede mirar el código fuente de la aplicación de ejemplo **xsadmin** en el archivo *inicio_wxs*/samples/xsadmin.jar de una instalación autónoma o en el archivo *inicio_wxs*/xsadmin.jar de una instalación de WebSphere Application Server.

Puede utilizar el programa de utilidad de ejemplo **xsadmin** para visualizar el diseño actual y estado específico de la cuadrícula de datos como, por ejemplo, contenido de la cuadrícula. En este ejemplo, el diseño de la cuadrícula de datos de esta tarea consta de una sola cuadrícula de datos *ObjectGridA* con una correlación *MapA* que pertenece al conjunto de correlaciones *MapSetA*. Este ejemplo muestra cómo puede visualizar todos los contenedores activos en una cuadrícula de datos e imprimir métricas filtradas relacionadas con el tamaño de correlación de la correlación *MapA*. Para ver todas las posibles opciones del mandato, ejecute el programa de utilidad **xsadmin** sin los argumentos o con la opción **-help**.

Procedimiento

1. Vaya al directorio bin.

   ```
   cd inicio_wxs/bin
   ```

2. Ejecute el programa de utilidad **xsadmin**.
   - Para visualizar la ayuda en línea, ejecute el mandato siguiente:

     UNIX

     ```
     xsadmin.sh
     ```

     Windows

     ```
     xsadmin.bat
     ```

Debe pasar sólo una de las opciones listadas para que el programa de utilidad funcione. Si no se especifica ninguna opción **-g** o **-m**, el programa de utilidad **xsadmin** muestra la información para cada una de las cuadrículas de la topología.

- Para habilitar estadísticas para todos los servidores, ejecute el mandato siguiente:

  UNIX

  ```
  xsadmin.sh -g ObjectGridA -setstatsspec ALL=enabled
  ```

  Windows

  ```
  xsadmin.bat -g ObjectGridA -setstatsspec ALL=enabled
  ```

- Para visualizar todos los contenedores en línea para una cuadrícula, ejecute el mandato siguiente:

  UNIX

  ```
  xsadmin.sh -g ObjectGridA -m MapSetA -containers
  ```

  Windows

  ```
  xsadmin.bat -g ObjectGridA -m MapSetA -containers
  ```

  Se visualiza toda la información de contenedores. A continuación se muestra un ejemplo de la salida:

  ```
  Conexión al servicio de catálogo en localhost:1099


  *** Mostrar todos los contenedores en línea para la cuadrícula - 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
  ```

  Atención: Para obtener esta información cuando Transport Layer Security/Secure Sockets Layer (TLS/SSL) está habilitado, debe iniciar los servidores de catálogo y contenedor con el puerto de servicio JMX establecido. Para establecer el puerto de servicio JMX, puede utilizar la opción **-JMXServicePort** en el script **startOgServer** o puede llamar al método setJMXServicePort en la interfaz ServerProperties.

- Para conectarse al servicio de catálogo y visualizar información sobre MapA, ejecute el mandato siguiente:

  UNIX

  ```
  xsadmin.sh -g ObjectGridA -m MapSetA -mapsizes -fm MapA
  ```

  Windows

  ```
  xsadmin.bat -g ObjectGridA -m MapSetA -mapsizes -fm MapA
  ```

  Se muestra el tamaño de la correlación especificada. A continuación se muestra un ejemplo de la salida:

  ```
  Conexión al servicio de catálogo en localhost:1099

  ****Mostrando resultados para Grid - ObjectGridA, MapSet - MapSetA*****
  ```

```
*** Listado de Maps para server1 ***
 Map Name Partition Map Size Used Bytes (B) Shard Type
 MapA     0         0        0              Primary
```

- Para conectarse al servicio de catálogo utilizando un puerto JMX específico y visualizar información sobre la correlación MapA, ejecute el mandato siguiente: 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
```

El programa de utilidad de ejemplo **xsadmin** se conecta al servidor MBean que se ejecuta en un servidor de catálogo. Un servidor de catálogo puede ejecutarse como proceso autónomo, WebSphere Application Server o incorporado en un proceso de aplicación personalizado. Utilice la opción **-ch** para especificar el nombre de host del servicio de catálogo, y la opción **-p** para especificar el puerto de denominación del servicio de catálogo.

Se muestra el tamaño de la correlación especificada. A continuación se muestra un ejemplo de la salida:

```
Conexión al servicio de catálogo en CatalogMachine:6645

*****Mostrando resultados para Grid - ObjectGridA, MapSet - MapSetA*****


*** Listado de Maps para server1 ***
Map Name: MapA  Partition #: 0  Map Size: 0  Shard Type: Primary
Server Total: 0
```

- Para conectarse a un servicio de catálogo alojado en un proceso de WebSphere Application Server, realice los pasos siguientes:

  La opción **-dmgr** es necesaria al conectarse con un servicio de catálogo que alberga cualquier proceso o clúster de procesos de WebSphere Application Server. Utilice la opción **-ch** para especificar el nombre de host si no es `localhost`, y la opción **-p** para alterar temporalmente el puerto del programa de arranque del servicio de catálogo, que utilice el proceso BOOTSTRAP_ADDRESS. La opción **-p** solo es necesaria si BOOTSTRAP_ADDRESS no está establecido en el valor predeterminado de `9809`.

  Nota: La versión autónoma de WebSphere eXtreme Scale no se puede utilizar para conectarse a un servicio de catálogo alojado por un proceso de WebSphere Application Server. Utilice el programa de utilidad **xsadmin** que es el script incluido en el directorio *raíz_was*`/bin`, disponible al instalar WebSphere eXtreme Scale en WebSphere Application Server o WebSphere Application Server Network Deployment.

  a. Desplácese al directorio bin de WebSphere Application Server:

     ```
     cd raíz_was/bin
     ```

  b. Inicie el programa de utilidad **xsadmin** mediante el siguiente mandato:

     UNIX

     ```
     xsadmin.sh -g ObjectGridA -m MapSetA -mapsizes -fm MapA -dmgr
     ```

     Windows

     ```
     xsadmin.bat -g ObjectGridA -m MapSetA -mapsizes -fm MapA -dmgr
     ```

Se muestra el tamaño de la correlación especificada.

```
Conexión al servicio de catálogo en localhost:9809

****Mostrando resultados para Grid - ObjectGridA, MapSet - MapSetA*****


*** Listado de Maps para server1 ***
Map Name: MapA  Partition #: 0  Map Size: 0  Shard Type: Primary
Server Total: 0
```

- Para visualizar la colocación configurada y de tiempo de ejecución de la configuración, ejecute uno de los mandatos siguientes:

```
xsadmin -placementStatus
xsadmin -placementStatus -g myOG -m myMapSet
xsadmin -placementStatus -m myMapSet
xsadmin -placementStatus -g myOG
```

Puede hacer que el mandato visualice información de colocación para toda la configuración, una sola cuadrícula de datos, un único conjunto de correlaciones o una combinación de una cuadrícula de datos y un conjunto de correlaciones. A continuación se muestra un ejemplo de la salida:

```
***********Mostrar estado de colocación para 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>
```

Creación de un perfil de configuración para el programa de utilidad `xsadmin`

Puede guardar los parámetros especificados con frecuencia para el programa de utilidad **xsadmin** en el archivo de propiedades. Como resultado, las llamadas al programa de utilidad **xsadmin** son más breves.

Antes de empezar

Cree un despliegue básico de WebSphere eXtreme Scale que incluya como mínimo un servidor de catálogo y como mínimo un servidor de contenedor. Para obtener más información, consulte Script **startOgServer**.

Acerca de esta tarea

Consulte "Referencia del programa de utilidad **xsadmin**" en la página 211 para ver una lista de las propiedades que puede colocar en un perfil de configuración para el programa de utilidad **xsadmin**. Si especifica un archivo de propiedades y un parámetro correspondiente como argumento de línea de mandatos, el argumento de línea de mandatos sustituye el valor del archivo de propiedades.

Procedimiento

1. Cree un archivo de propiedades del perfil de configuración. Este archivo de propiedades debe contener las propiedades globales que desee utilizar en todas las invocaciones del mandato **xsadmin**.

Guarde el archivo de propiedades con cualquier nombre que elija. Por ejemplo, podría colocar el archivo en la vía de acceso siguiente: `/opt/ibm/WebSphere/wxs71/ObjectGrid/security/<my.properties>`.

Sustituya `<my.properties>` con el nombre de su archivo. Por ejemplo, podría establecer las propiedades siguientes en el archivo:

- `XSADMIN_TRUST_TYPE=jks`
- `XSADMIN_TRUST_PATH=/opt/ibm/WebSphere/wxs71/ObjectGrid/bin/security/key.jks`
- `XSADMIN_USERNAME=ogadmin`

2. Ejecute el programa de utilidad xsadmin con el archivo de propiedades que ha creado. Utilice el parámetro **-profile** para indicar la ubicación del archivo de propiedades. También puede utilizar el parámetro **-v** para visualizar la salida detallada.

```
./xsadmin.sh -l -v -password xsadmin -ssl -trustPass ogpass -profile
/opt/ibm/WebSphere/wxs71/ObjectGrid/security/<my.properties>
```

Referencia del programa de utilidad `xsadmin`

Puede pasar argumentos al programa de utilidad **xsadmin** con dos métodos distintos: con un argumento de línea de mandatos o con un archivo de propiedades.

Argumentos de `xsadmin`

Puede definir un archivo de propiedades para el programa de utilidad **xsadmin** con versión 7.1 Fix 1 o posterior. Al crear un archivo de propiedades puede ahorrarse algunos de los argumentos utilizados con frecuencia, como por ejemplo el nombre de usuario. En la tabla siguiente se indican las propiedades que puede añadir a un archivo de propiedades. Si especifica tanto una propiedad en un archivo de propiedades como el argumento de línea de mandatos equivalente, este último sobrescribe el valor del archivo de propiedades.

Para obtener más información sobre cómo definir un archivo de propiedades para el programa de utilidad **xsadmin**, consulte "Creación de un perfil de configuración para el programa de utilidad **xsadmin**" en la página 210.

Tabla 10. Argumentos del programa de utilidad **xsadmin**

Argumento de línea de mandatos	Nombre de la propiedad equivalente en el archivo de propiedades	Descripción y valores válidos
-bp	n/d	Indica el puerto de escucha. **Valor predeterminado:** 2809
-ch	n/d	Indica el nombre de host JMX para el servidor de catálogo. **Valor predeterminado:** localhost
-clear	n/d	Borra la correlación especificada. **Permite los filtros siguientes:** -fm
-containers	n/d	Para cada cuadrícula de datos y conjunto de correlaciones, muestra una lista de servidores de contenedores. **Permite los filtros siguientes:** -fnp
-continuous	n/d	Especifique este distintivo si desea resultados de tamaño de correlación continuos para supervisar la cuadrícula de datos. Al ejecutar este mandato con el argumento **-mapsizes**, el tamaño de la correlación se visualiza cada 20 segundos.
-coregroups	n/d	Muestra todos los grupos principales del servidor de catálogo. Este argumento se utiliza para diagnósticos avanzados.

Tabla 10. Argumentos del programa de utilidad **xsadmin** *(continuación)*

Argumento de línea de mandatos	Nombre de la propiedad equivalente en el archivo de propiedades	Descripción y valores válidos
-dismissLink *<dominio_servicio_catálogo>*	n/d	Elimina un enlace entre dos dominios de servicio de catálogo. Indique el nombre del dominio foráneo de servicio de catálogo al que se ha conectado previamente con el argumento **-establishLink**.
-dmgr	n/d	Indica si está estableciendo conexión con un servicio de catálogo alojado en WebSphere Application Server. **Valor predeterminado:**`false`
-empties	n/d	Especifique este distintivo si desea que se muestren los contenedores vacíos en la salida.
-establishLink *<nombre_dominio_foráneo> <host1:puerto1,host2:puerto2...>*	n/d	Conecta el dominio de servicio de catálogo con un dominio de servicio de catálogo foráneo. Utilice el formato siguiente: `-establishLink <nombre_dominio_foráneo> <host1:puerto1,host2:puerto2...>`. *nombre_dominio_foráneo* es el nombre del dominio de servicio de catálogo foráneo y *host1:puerto1,host2:puerto2...* es una lista separada por comas de nombres de host de servidores de catálogos y puertos ORB (Object Request) que están en ejecución en este dominio de servicio de catálogo.
-fc	n/d	Filtra únicamente para este contenedor. Si está filtrando servidores de contenedor en un entorno de WebSphere Application Server Network Deployment, utilice el siguiente formato: *<nombre_célula>/<nombre_nodo>/<nombreServidor_sufijoContenedor>* **Utilícelo con los argumentos siguientes: -mapsizes, -teardown,-revisions,-getTraceSpec,-setTraceSpec,-getStatsSpec,-setStatsSpec**
-fh	n/d	Filtra únicamente para este host. **Utilícelo con los argumentos siguientes: -mapsizes, -teardown,-revisions,-getTraceSpec,-setTraceSpec,-getStatsSpec,-setStatsSpec,-routetable**
-fm	n/d	Filtra solo para esta correlación. **Utilícelo con los argumentos siguientes: -clear, -mapsizes**
-fnp	n/d	Filtra los servidores que no tienen fragmentos primarios. **Utilícelo con los argumentos siguientes: -containers**
-fp	n/d	Filtra únicamente para esta partición. **Utilícelo con los argumentos siguientes: -mapsizes, -teardown,-revisions,-getTraceSpec,-setTraceSpec,-getStatsSpec,-setStatsSpec,-routetable**
-fs	n/d	Filtra únicamente para este servidor. Si está filtrando servidores de aplicaciones en un entorno de WebSphere Application Server Network Deployment, utilice el formato siguiente: *<nombre_célula>/<nobre_nodo>/<nombre_servidor>* **Utilícelo con los argumentos siguientes: -mapsizes, -teardown,-revisions,-getTraceSpec,-setTraceSpec,-getStatsSpec,-setStatsSpec**
-fst	n/d	Filtra únicamente para este tipo de fragmento. Especifique P únicamente para fragmentos primarios, A únicamente para fragmentos de réplicas asíncronas, y S únicamente para fragmentos de réplicas síncronas. **Utilícelo con los argumentos siguientes: -mapsizes, -teardown,-revisions,-getTraceSpec,-setTraceSpec,-getStatsSpec,-setStatsSpec**
-fz	n/d	Filtra únicamente para esta zona. **Utilícelo con los argumentos siguientes: -mapsizes, -teardown,-revisions,-getTraceSpec,-setTraceSpec,-getStatsSpec,-setStatsSpec,-routetable**
-force	n/d	Fuerza la acción que indica el mandato, inhabilitando las posibles solicitudes preventivas. Este argumento es útil para la ejecución de mandatos por lotes.
-g	n/d	Especifica el nombre de ObjectGrid.
-getstatsspec	n/d	Visualiza la información de estadísticas actual. Puede establecer la especificación de estadísticas con el argumento **-setstatsspec**. **Permite los filtros siguientes:** `-fst -fc -fz -fs -fh -fp`

Tabla 10. Argumentos del programa de utilidad **xsadmin** *(continuación)*

Argumento de línea de mandatos	Nombre de la propiedad equivalente en el archivo de propiedades	Descripción y valores válidos
-getTraceSpec	n/d	Visualiza la especificación de rastreo actual. Puede establecer la especificación de rastreo con el argumento **-settracespec**. **Permite los filtros siguientes:** `-fst -fc -fz -fs -fh -fp`
-h	n/d	Visualiza la ayuda del programa de utilidad **xsadmin**, que incluye una lista de argumentos.
-hosts	n/d	Muestra todos los hosts de la configuración.
-jmxUrl	XSADMIN_JMX_URL	Especifica la direccionar de un servidor conector JMX API en el formato siguiente: `service:jmx:`*`protocolo`*`:`*`sap`*. A continuación se indican las definiciones de las variables *protocolo sap*: *protocolo* Especifica el protocolo de transporte a utilizar para establecer conexión con el servidor conector. *sap* Especifica la dirección en la que se encuentre el servidor conector. Para obtener más información sobre el formato del URL del servicio JMX, consulte Class JMXServiceURL (Java 2 Platform SE 5.0).
-l	n/d	Muestra todas las cuadrículas de datos y conjuntos de correlaciones conocidos.
-m	n/d	Especifica el nombre del conjunto de correlaciones.
-mapsizes	n/d	Muestra el tamaño de cada correlación del servidor de catálogo para verificar que la distribución de claves es uniforme en los distintos fragmentos. **Permite los filtros siguientes:** `-fm -fst -fc -fz -fs -fh -fp`
-mbeanservers	n/d	Muestra una lista de todos los puntos finales de servidor MBean.
-overridequorum	n/d	Sobrescribe el valor de quórum de forma que los sucesos del servidor de contenedor no se ignoran en caso de error del centro de datos.
-password	XSADMIN_PASSWORD	Especifica la contraseña utilizada para iniciar sesión en el programa de utilidad **xsadmin**. No especifique la contraseña en el archivo de propiedades si desea que la contraseña se mantenga segura.
-p	n/d	Indica el puerto JMX del host del servidor de catálogo. **Valor predeterminado:** 1099 ó 9809 para un host de WebSphere Application Server, 1099 para configuraciones autónomas.
-placementStatus	n/d	Muestra la colocación configurada y la colocación en tiempo de ejecución de la configuración. Puede definir el ámbito de la salida en una combinación de cuadrículas de datos y conjuntos de correlaciones o para la configuración completa: • Configuración completa `-placementStatus` • Para una cuadrícula de datos: específica: `-placementStatus -g` *`mi_cuadrícula`* • Para un conjunto de correlaciones específico: `-placementStatus -m` *`mi_conjunto_correlaciones`* • Para una cuadrícula de datos y un conjunto de correlaciones específicos: `-placementStatus -g` *`mi_cuadrícula`* `-m` *`mi_conjunto_correlaciones`*
-primaries	n/d	Muestra una lista de los fragmentos primarios.
-profile	n/d	Especifica una vía de acceso completa al archivo de propiedades del programa de utilidad **xsadmin**.
-quorumstatus	n/d	Muestra el estado de quórum del servicio de catálogo.
-releaseShard *<nombre_servidor_contenedor> <nombre_cuadrícula_objetos> <nombre_conjunto_correlaciones> <nombre_partición>*	n/d	Se utiliza junto con el argumento **-reserveShard**. Se debe invocar el argumento **-releaseShard** después de que se haya reservado y colocado un fragmento. El argumento **-releaseShard** invoca el método ContainerMBean.release().
-reserved	n/d	Se utiliza con el argumento **-containers** para visualizar solo fragmentos que se han reservado con el argumento **-reserveShard**.

Tabla 10. Argumentos del programa de utilidad **xsadmin** *(continuación)*

Argumento de línea de mandatos	Nombre de la propiedad equivalente en el archivo de propiedades	Descripción y valores válidos
-reserveShard <*nombre_servidor_contenedor*> <*nombre_cuadrícula_objetos*> <*nombre_conjunto_correlaciones*> <*nombre_partición*>	n/d	Mueve un fragmento primario al servidor de contenedor especificado. Este argumento invoca el método ContainerMBean.reserve().
-resumeBalancing <*nombre_objectgrid*> <*nombre_conjunto_correlaciones*>	n/d	Intenta equilibrar las solicitudes y permitir futuros intentos de reequilibrado en el ObjectGrid y conjunto de correlaciones especificados.
-revisions	n/d	Visualiza identificadores de revisión de un dominio de servicio de catálogo, que incluyen: cada cuadrícula de datos, número de partición, tipo de partición (primaria o de réplica), dominio de servicio de catálogo, ID de tiempo de vida y número de revisiones de datos para cada fragmento específico. Puede utilizar este argumento para determinar si se detecta una réplica asíncrona o un dominio enlazado. Este argumento invoca el método ObjectGridMBean.getKnownRevisions(). **Permite los filtros siguientes:** `-fst -fc -fz -fs -fh -fp`
-routetable	n/d	Muestra el estado actual de la cuadrícula de datos desde una perspectiva de cliente-servidor. La tabla de direccionamiento es la información que un cliente-servidor de ObjectGrid utiliza para la comunicación con la cuadrícula de datos. Utilice la tabla de direccionamiento como una ayuda para el diagnóstico cuando intente identificar problemas de conexiones o excepciones `TargetNotAvailable`. **Argumentos necesarios:** en un entorno autónomo, debe especificar los parámetros **-bp** y **-p** con este argumento si no está utilizando los valores predeterminados para el puerto de escucha del programa de arranque y el puerto JMX para el host de servidor de catálogo. **Permite los filtros siguientes:** `-fz -fh -fp`
-settracespec <*serie_rastreo*>	n/d	Habilita el rastreo en los servidores en tiempo de ejecución. Consulte el siguiente ejemplo: `-setTraceSpec "ObjectGridReplication=all=enabled"` Consulte los apartados Recopilación de rastreo y Opciones de rastreo para obtener más información sobre las series de rastreo que puede especificar. **Permite los filtros siguientes:** `-fst -fc -fz -fs -fh -fp`
-swapShardWithPrimary <*nombre_servidor_contenedor*> <*nombre_objectgrid*> <*nombre_conjunto_correlaciones*> <*nombre_partición*>	n/d	Intercambia el fragmento de réplica especificado del servidor de contenedor especificado con el fragmento primario. Mediante la ejecución de este mandato puede equilibrar manualmente los fragmentos primarios cuando sea necesario.
-setstatsspec <*espec_estad*>	n/d	Habilita la recopilación de estadísticas. Este argumento invoca los métodos DynamicServerMBean.setStatsSpec y DynamicServerMBean.getStatsSpec. Consulte Clase StatsSpec para obtener más información sobre los módulos de estadísticas que puede supervisar. **Permite los filtros siguientes:** `-fm -fst -fc -fz -fs -fh -fp`
-suspendBalancing <*nombre_objectgrid*> <*nombre_conjunto_correlaciones*>	n/d	Evita intentos futuros de equilibrar el ObjectGrid y el conjunto de correlaciones especificados.
-ssl	n/d	Indica que la Capa de sockets seguros (SSL- Secure Sockets Layer) está habilitada.
-teardown	n/d	Detiene una lista o un grupo de servidores de catálogo y de contenedor. **Permite los filtros siguientes:** `-fst -fc -fz -fs -fh -fp` **Formato para proporcionar una lista de servidores:** *nombre_servidor_1*,*nombre_servidor_2* ... **Para detener todos los servidores de una zona, incluya el argumento -fz:** –fz <*nombre_zona*> **Para detener todos los servidores de un host, incluya el argumento -fh:** –fh <*nombre_host*>

Tabla 10. Argumentos del programa de utilidad **xsadmin** *(continuación)*

Argumento de línea de mandatos	Nombre de la propiedad equivalente en el archivo de propiedades	Descripción y valores válidos
-triggerPlacement	n/d	Fuerza la ejecución de la colocación de fragmentos, ignorando el valor **numInitialContainers** configurado en el archivo XML de despliegue. Puede utilizar este argumento cuando realice mantenimiento en los servidores para permitir que la colocación de fragmentos se continúe ejecutando, aunque el valor **numInitialContainers** sea inferior al valor configurado.
-trustPass	XSADMIN_TRUST_PASS	Especifica la contraseña del almacén de confianza especificado.
-trustPath	XSADMIN_TRUST_PATH	Especifica una vía de acceso al archivo de almacén de confianza. Ejemplo: `etc/test/security/server.public`
-trustType	XSADMIN_TRUST_TYPE	Especifica el tipo de almacén de confianza. Valores válidos: JKS, JCEK, PKCS12, etc.
-unassigned	n/d	Muestra una lista de fragmentos que no se pueden colocar en la cuadrícula de datos. Los fragmentos no se pueden colocar cuando el servicio de colocación tiene una restricción que evita la colocación.
-username	XSADMIN_USERNAME	Especifica el nombre de usuario para iniciar sesión en el programa de utilidad **xsadmin**.
-v	n/d	Habilita la acción de línea de mandatos detallada. Utilice este distintivo si utiliza variables de entorno, un archivo de propiedades o ambas cosas para especificar ciertos argumentos de línea de mandatos y desea ver sus valores. Consulte el apartado "Opción verbose del programa de utilidad **xsadmin**" para obtener más información.
-xml	n/d	Visualiza la salida no filtrada del método PlacementServiceMBean.listObjectGridPlacement(). Los otros argumentos **xsadmin** filtran la salida de este método y organizan los datos en un formato más consumible.

Opción verbose del programa de utilidad `xsadmin`

Puede utilizar la opción verbose de **xsadmin** para solucionar problemas. Ejecute el mandato `xsadmin -v` para que se listen todos los parámetros configurados. La opción verbose muestra todos los valores de todos los ámbitos, incluidos argumentos de línea de mandatos, argumentos de archivo de propiedades y argumentos especificados por el entorno. La sección `Argumentos efectivos` incluye los valores que se utilizan en el entorno si ha especificado la misma propiedad utilizando varios ámbitos.

Ejemplo de la opción verbose

Argumentos del mandato `xsadmin`:

El texto siguiente es un ejemplo de salida cuando se utiliza la opción verbose desde la línea de mandatos después de haber ejecutado el mandato siguiente con un valor de propiedades especificado:

```
./xsadmin -l -v -username xsadmin -password xsadmin -ssl -trustPass ogpass
-profile /opt/ibm/WebSphere/wxs71/ObjectGrid/security/my.properties
```

Argumentos del archivo de propiedades:

A continuación se muestra el contenido del archivo `/opt/ibm/WebSphere/wxs71/ObjectGrid/security/my.properties properties`:

```
XSADMIN_TRUST_PASS=ogpass
XSADMIN_TRUST_TYPE=jks
XSADMIN_TRUST_PATH=/opt/ibm/WebSphere/wxs71/ObjectGrid/bin/security/key.jks
XSADMIN_USERNAME=ogadmin
XSADMIN_PASSWORD=ogpass
```

Resultados del mandato:

En la siguiente salida del mandato **xsadmin** anterior, el texto que está en ***negrita cursiva*** indica propiedades y valores especificados tanto en la línea de mandatos como en el archivo de propiedades. En la sección Argumentos efectivos de la línea de mandatos, puede ver que los argumentos especificados en la línea de mandatos sustituyen a los valores del archivo de propiedades.

```
Argumentos especificados en la línea de mandatos
*********************************
XSADMIN_USERNAME=xsadmin
XSADMIN_PASSWORD=xsadmin
XSADMIN_TRUST_PATH=<no especificado>
XSADMIN_TRUST_TYPE=<no especificado>
XSADMIN_TRUST_PASS=ogpass
XSADMIN_PROFILE=/opt/ibm/WebSphere/wxs71/ObjectGrid/security/my.properties
XSADMIN_JMX_URL=<no especificado>
*********************************
Argumentos especificados del archivo de propiedades
*********************************
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=<no especificado>
*********************************
Argumentos especificados por el entorno
*********************************
XSADMIN_USERNAME=<no especificado>
XSADMIN_PASSWORD=<no especificado>
XSADMIN_TRUST_PATH=<no especificado>
XSADMIN_TRUST_TYPE=<no especificado>
XSADMIN_TRUST_PASS=<no especificado>
XSADMIN_JMX_URL=<no especificado>
*********************************
Argumentos efectivos
*********************************
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=<no especificado>
SSL authentication enabled: true
*********************************
Conexión al servicio de catálogo en localhost:1099
*** Mostrar todos los nombres 'objectGrid:mapset'
Grid Name  MapSet Name
accounting defaultMapSet
```

Atención: La propiedad XSADMIN_PROFILE, aunque se visualiza en la modalidad detallada, no es una clave válida que pueda especificar en un archivo de propiedades. El valor de esta propiedad en la salida detallada indica el valor de la propiedad que se utiliza, tal como se indica en el argumento de línea de mandatos **-profile**.

Salida sin la opción detallada

A continuación se muestra un ejemplo de la misma salida de mandato sin la opción verbose habilitada:

```
./xsadmin -l -username xsadmin -password xsadmin -ssl -trustPass ogpass
-profile /opt/ibm/WebSphere/wxs71/ObjectGrid/security/my.properties
```

```
Conexión al servicio de catálogo en localhost:1099
*** Mostrar todos los nombres 'objectGrid:mapset'
Grid Name  MapSet Name
accounting defaultMapSet
```

Avisos

Las referencias en esta publicación a productos, programas o servicios de IBM no implica que IBM tenga previsto ponerlos a la venta en todos los países en los que IBM opera. Cualquier referencia a un producto, programa o servicio de IBM no pretende indicar ni implica que sólo se pueda utilizar este producto, programa o servicio de IBM. En su lugar, se puede utilizar cualquier producto, programa o servicio funcionalmente equivalente que no vulnere ningún derecho de propiedad intelectual de IBM. La evaluación y la verificación del funcionamiento con otros productos, excepto aquellos expresamente designados por IBM, es responsabilidad del usuario.

IBM puede tener patentes o solicitudes de patentes pendientes que conciernan al tema de este documento. La posesión de este documento no le da ninguna licencia sobre estas patentes. Puede enviar preguntas acerca de licencias por escrito a:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, New York 10594 Estados Unidos

Los propietarios de licencias de este programa que deseen obtener información sobre el mismo con el fin de habilitar: (i) el intercambio de información entre programas creados de forma independiente y otros programas (incluido este) y (ii) el uso mutuo de la información intercambiada, se deben poner en contacto con:

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

Esta información puede estar disponible, bajo las condiciones y los términos adecuados, incluyendo en algunos casos, el pago de una cuota.

Marcas registradas

Los siguientes términos son marcas registradas de IBM Corporation en Estados Unidos yen otros países.

- AIX
- CICS
- Cloudscape
- DB2
- Domino
- IBM
- Lotus
- RACF
- Redbooks
- Tivoli
- WebSphere
- z/OS

Java y todas las marcas registradas basadas en Java son marcas registradas de Sun Microsystems, Inc. en Estados Unidos y/o en otros países.

LINUX es una marca registrada de Linus Torvalds en Estados Unidos y/o en otros países.

Microsoft, Windows, Windows NT y el logotipo de Windows son marcas registradas de Microsoft Corporation en Estados Unidos y/o en otros países.

UNIX es una marca registrada de The Open Group en Estados Unidos y en otros países.

Otros nombres de compañías, productos y servicios pueden ser marcas registradas o de servicio de terceros.

Índice

A

AP 167
API
  DataSerializer 75
arquitectura
  clientes 16
  correlaciones 15
  fragmentos 13
  particiones 13
  servidores de contenedor 13
  topologías 141
  visión general 11

B

base de datos
  memoria caché complementaria 50, 150
  memoria caché de grabación a través 51, 151
  memoria caché de grabación diferida 53, 154
  memoria caché de lectura a través 51, 151
  memoria caché escasa y completa 49, 150
  precarga de datos 59, 160
  preparación de datos 59, 160
  sincronización 61, 162
  técnicas de sincronización de base de datos 61, 162
bloqueo
  estrategias de 124
  optimista 124
  pesimista 124

C

cargadores
  base de datos 58, 159
  Visión general de JPA (Java Persistence API) 66
código de ejemplo 205
colocación
  estrategias 80
  visión general 76
contenedor OSGi
  configuración de Apache Aries Blueprint 190
convenios de directorio 8
correlaciones de respaldo
  estrategia de bloqueo 123
cuadrículas de datos 76

D

desalojadores
  visión general 22
disponibilidad
  anomalía 92
  conectividad 92
  réplica 95
  réplica de conjunto de correlaciones 120
  visión general 92
disponibilidad de partición (AP) 167
distribuir cambios
  usar Java Message Service 127
dominios de servicio de catálogo 101

E

Eclipse Equinox
  configuración del entorno 183
ejemplos 205
equilibrio de carga
  conjuntos de correlaciones 120
  réplica 95
  réplicas 114
escalabilidad
  con unidades o datos 91
  visión general 76
escenarios 181

F

fragmentos
  anomalía 114
  asignación 111
  ciclo de vida 114
  colocación 76
  primario 111
  recuperación 114
  réplica 111

G

gestor de sesiones HTTP
  visión general 33
grabación diferida
  integración de la base de datos 53, 154

I

índices
  calidad de los datos 64, 165
  rendimiento 64, 165
integración con otros servidores 180
integración de la memoria caché
  visión general 26
interoperatividad del gestor de sesiones
  con productos WebSphere 180

J

Java Persistence API (JPA)
  plug-in de memoria caché
    introducción 26
  topología de memoria caché
    con partición incorporada 26
    incorporada 26
    remota 26

M

memoria caché 206
  distribuido 146
  embedded 145
  local 142
  visión general 11
  visión general técnica 10
memoria caché coherente 48, 149
memoria caché complementaria
  integración de la base de datos 50, 150
memoria caché completa 49, 150
memoria caché dinámica
  visión general 35
memoria caché distribuida 146
memoria caché en línea 50, 150
memoria caché escasa 49, 150
memoria caché incorporada 145
memoria caché local
  réplica por igual 143

N

notas del release 6
nuevas características 4

O

ordenación
  visión general 68
OSGi
  entorno de Eclipse Equinox 183
  visión general 24, 181

P

particiones
  colocación fija 80
  con entidades 78
  introducción 78
  transacciones 84, 128
  visión general 76
planificar 141
plug-ins
  DataSerializer 75
  ObjectTransformer 71
precarga de correlaciones
  conjuntos de correlaciones 120
  equilibrio de carga 114

precarga de correlaciones *(continuación)*
réplica 95
proceso de transacción Extreme 1
prueba gratuita 206
programa de utilidad xsadmin
mandatos 211
opción verbose 215
perfil de configuración 210
supervisar 207
propiedades
ejemplos 206
proveedor de memoria caché dinámica
introducción 35
prueba gratuita 206

Q

quórums
visión general 104

R

rendimiento
equilibrio de carga 114
réplica 95
réplica de conjunto de correlaciones 120
réplica
cargadores 109
coste de memoria 109
tipos de fragmento 109
réplica de cuadrícula de datos multimaestro
planificar 167
réplica multimaestro
planificación de la configuración 172
planificación del diseño 174
planificar 167
planificar para cargadores 173
topologías 167
réplicas
lectura de datos 113
requisitos
hardware 7
software 7
resolución de problemas
notas del release 6

S

seguridad
autenticación 135
autorización 135
transporte seguro 135
serialización 68
Java 70
visión general 75
servicio de catálogo
visión general 12
servicio de datos REST
planificar 137
visión general 137
servidores de contenedor
alta disponibilidad 101
colocación por contenedor 80
visión general 13
sesiones 33
soporte 6

T

topologías
clientes 16
correlaciones 15
plan 141
servidores de contenedor 13
visión general 11
transacciones
copyMode 122
cuadrícula cruzada 84, 128
partición única 84, 128
visión general 121
visión general del proceso 120

V

validación basada en sucesos 63, 164
ventajas
almacenar en memoria caché de grabación diferida 53, 154
visión general
visión general del producto 1
visión general técnica 10
visión general de eXtreme Scale 1
Visión general de eXtreme Scale
prueba gratuita 206

Z

zonas
visión general 17

Impreso en España