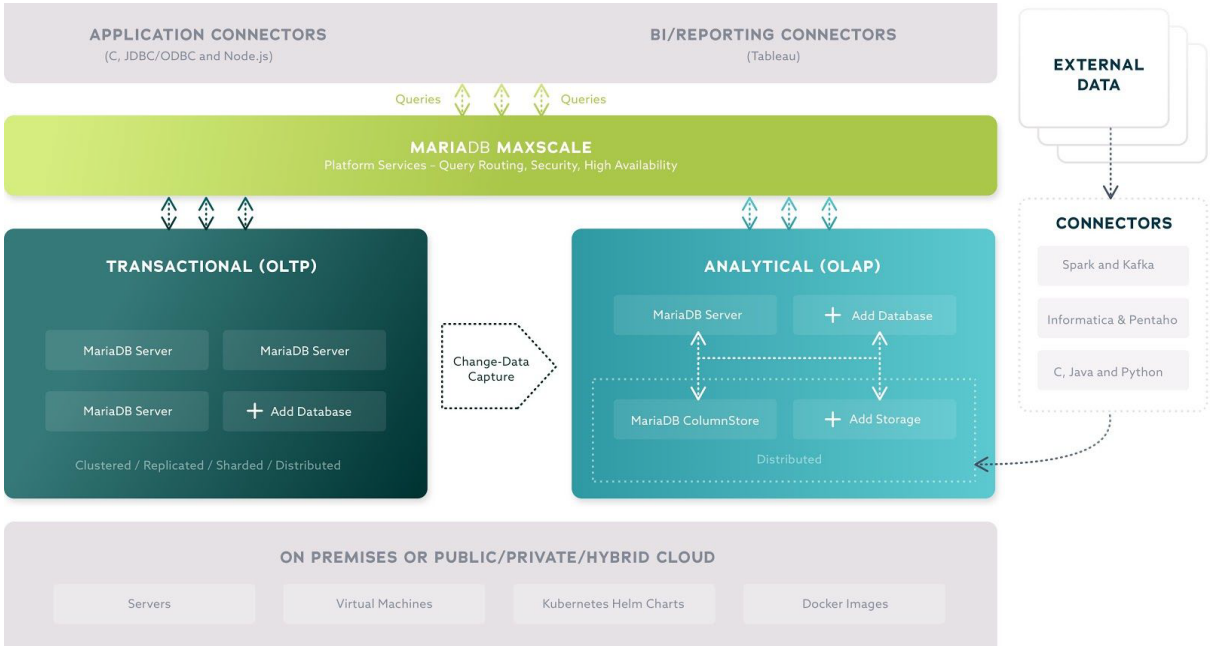
MariaDB

MARIADB ENTERPRISE: MANUAL DE ALTA DISPONIBILIDAD

MARIADB ENTERPRISE

Transacciones y análisis, UNITED

MariaDB Enterprise es una base de datos de código abierto para el procesamiento transaccional, analítico o híbrido transaccional/analítico a escala. Al preservar los datos históricos y optimizar los análisis en tiempo real sin dejar de procesar las transacciones, MariaDB Enterprise proporciona a las empresas los medios para crear ventajas competitivas y rentabilizar los datos, es decir, proporcionar a los clientes impulsados por los datos una visión procesable y además, dotarles de capacidades de análisis propias.



MariaDB Server

MariaDB Server es la base de MariaDB Enterprise. Es la única base de datos de código abierto con las mismas características empresariales de las bases de datos propietarias, incluidas la compatibilidad de la base de datos Oracle (por ej., PL/SQL), las tablas temporales, el *sharding* (escalado horizontal de la base de datos), la reversión de la base de datos a un punto específico en el tiempo y el cifrado transparente de datos.

MariaDB ColumnStore

MariaDB ColumnStore amplía MariaDB Server con el almacenamiento distribuido en columnas y el procesamiento en paralelo a gran escala para el análisis *ad hoc* e interactivo de cientos de miles de millones de filas a través de SQL estándar, sin necesidad de crear y mantener índices, y con un 10 % del espacio en disco mediante alta compresión.

MariaDB MaxScale

MariaDB MaxScale proporciona a MariaDB Enterprise un conjunto de servicios para aplicaciones modernas, incluyendo el enrutamiento transparente de consultas y la captura de datos de cambio (CDC) para cargas de trabajo híbridas, transaccionales y analíticas, alta disponibilidad (por ejemplo, failover automático) y seguridad avanzada (por ejemplo, enmascaramiento de datos).

TABLA DE CONTENIDOS

INTRODUCCIÓN

En la actualidad, las empresas enfrentan una transformación digital: las operaciones fuera de línea se convierten en operaciones en línea, las aplicaciones empresariales se convierten en aplicaciones orientadas al cliente y el compromiso se genera en cualquier lugar y en todas partes a través de aplicaciones ***web***, móviles y de ***Internet of Things*** (IoT). Cuando se trata de la experiencia del cliente, la disponibilidad no es una preferencia, es un requisito.

Todo se basa en los datos. Debe estar disponible las 24 horas del día, los 7 días de la semana, los 365 días del año. Sin embargo, dado que la infraestructura puede fallar y fallará, la plataforma de la base de datos debe mantener o restablecer automáticamente la disponibilidad en caso de falla del servidor, de la base de datos o de la red.

MariaDB Enterprise utiliza almacenamiento local y replicación (con o sin clúster) para proporcionar alta disponibilidad a través de varios servidores de base de datos. No existe ningún punto de falla. De hecho, al configurar MariaDB Enterprise para alta disponibilidad, el tiempo de inactividad debido a una falla imprevista de la infraestructura queda prácticamente eliminado.

Sin embargo, cuando se trata de alta disponibilidad, se deben tomar en cuenta las compensaciones entre el desempeño, la durabilidad y la integridad. En ocasiones, la durabilidad y la integridad son más importantes que el desempeño. En ocasiones, el desempeño es más importante. Para dar con un equilibrio, hay que sopesar las necesidades del negocio, el caso de uso y los requisitos técnicos.

Esta guía trata sobre el modo en que la replicación y el agrupamiento funcionan en MariaDB Enterprise y el modo en que se pueden configurar las variables y los parámetros para mejorar el desempeño, la durabilidad o la integridad. Asimismo, menciona aspectos importantes de la selección entre la replicación asincrónica, semisincrónica y sincrónica y explica acerca del proceso de recuperación de fallas para diferentes topologías de replicación y clúster.

Esta guía detallará:

- Cómo funcionan la replicación maestro/réplica y los clúster multimaestros.
- Las ventajas y desventajas de la replicación asincrónica, semisincrónica y sincrónica.
- El impacto de la alta disponibilidad en el desempeño, la durabilidad y la integridad.
- Cómo mejorar la disponibilidad a partir de la detección de la tipología con *failover* automático.

Conceptos

El tiempo medio estimado entre fallas (MTBF) es una medida de fiabilidad. Es el promedio de tiempo transcurrido entre fallas (por ej., el tiempo estimado entre fallas de la base de datos). Cuanto más prolongado, mejor.

El tiempo medio estimado de recuperación (MTTR) es una medida de mantenimiento. Es el promedio de tiempo transcurrido entre fallas y recuperación (por ej., la duración de la recuperación de fallas de la base de datos). Cuanto menos prolongado, mejor.

Cálculo de disponibilidad
Si el MTBF es de 12 meses (525 601 minutos) y el MTTR es 5 minutos, la base de datos se encuentra disponible el 99,999 % del tiempo.
Disponibilidad = MTBF / (MTBF + MTTR)
525 601 / (525 601 + 5) = 99,999% disponibilidad

Terminología

Switchover ocurre cuando la base de datos activa pasa a base de datos en espera y la base de datos en espera pasa a base de datos activa, habitualmente en el contexto de un mantenimiento planificado.

Failover ocurre cuando una base de datos en espera pasa a base de datos activa debido a que la base de datos activa falló, no responde o es inaccesible.

Failback ocurre cuando la base de datos activa que falló pasa de nuevo a estar activa, y la base de datos activa temporal vuelve a ser una base de datos en espera.

La **arquitectura** *shared-nothing* hace referencia a aquella en la que no existen fuentes compartidas (por ej., CPU, disco o memoria) entre los servidores de la bases de datos: no existe un único punto de falla.

La **arquitectura** *share-everything* hace referencia a aquella que comparte el disco y la memoria entre servidores de bases de datos (por ej., Oracle RAC).

ALTA DISPONIBILIDAD

MariaDB Enterprise proporciona alta disponibilidad con clústeres multimaestros (sincrónico) o replicación maestro/réplica (asincrónico o semisincrónico) con *failover* automático.

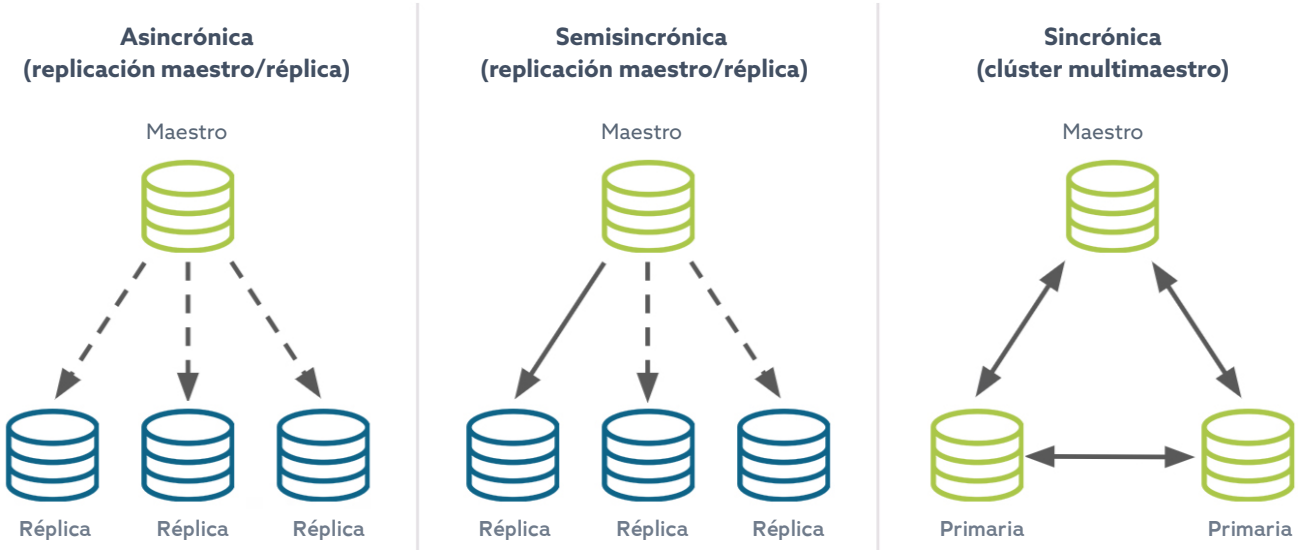


Diagrama 1: Diferentes tipos de replicación

Replicación maestro/réplica

El maestro asigna a las transacciones un ID de transacción global (GTID) y las escribe en su registro binario. La réplica solicita la transacción siguiente al primario y envía el GTID actual, la escribe en el registro de retransmisión y la ejecuta.

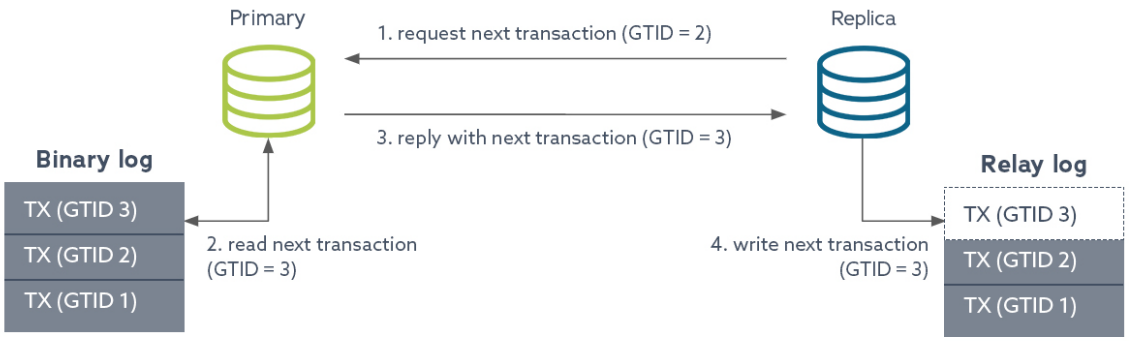


Diagrama 2: El proceso de replicación maestro/réplica

Automatic failover

El *proxy* de la base de datos, MariaDB MaxScale, realizará un *failover* automático si el maestro falla y promueve la réplica más actualizada (es decir, la que tiene el GTID más alto) a primaria y reconfigura el nodo restante para replicar desde aquella. Además, si se activa la reincorporación automática, el nodo maestro que falló se reconfigurará como réplica si se recupera.

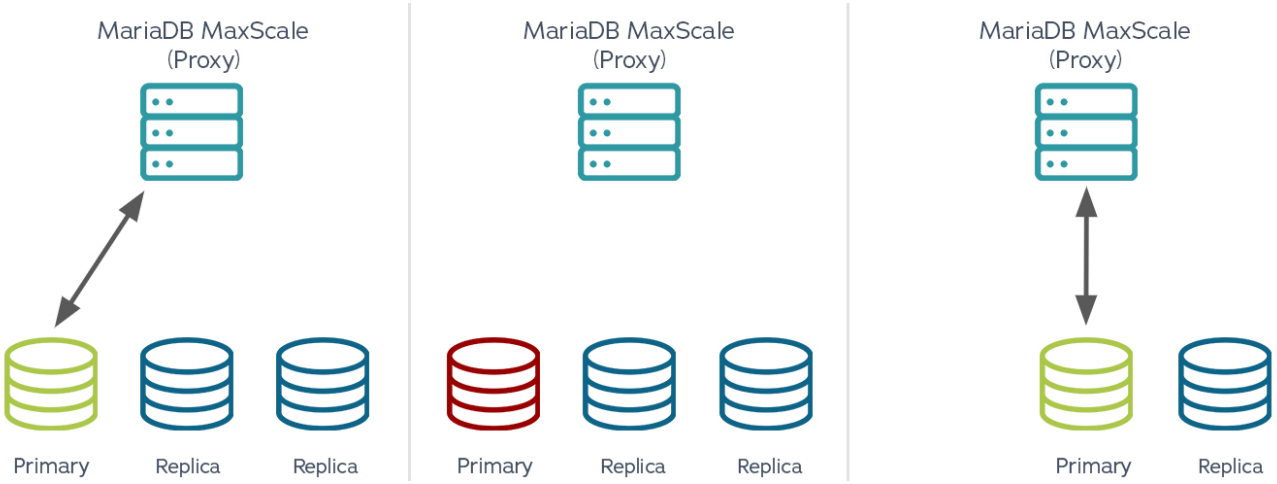


Diagrama 3: Failover automático con replicación maestro/réplic

Replicación asincrónica

Con la replicación asincrónica, las transacciones se reproducen luego de confirmarse. El maestro no espera ninguna de las reproducciones para reconocer la transacción antes de confirmarla. No afecta el rendimiento de la escritura. Sin embargo, si el maestro falla y el *failover* automático se encuentra activado, habrá pérdida de datos si una o más transacciones no se reprodujeron a una réplica.

En el siguiente ejemplo, el *proxy* de la base de datos promovería la réplica 1 a primaria porque es la réplica con el GTID más alto. Sin embargo, la transacción más reciente (GTID = 3) no se reprodujo antes de que el maestro fallara. Se perderían datos.

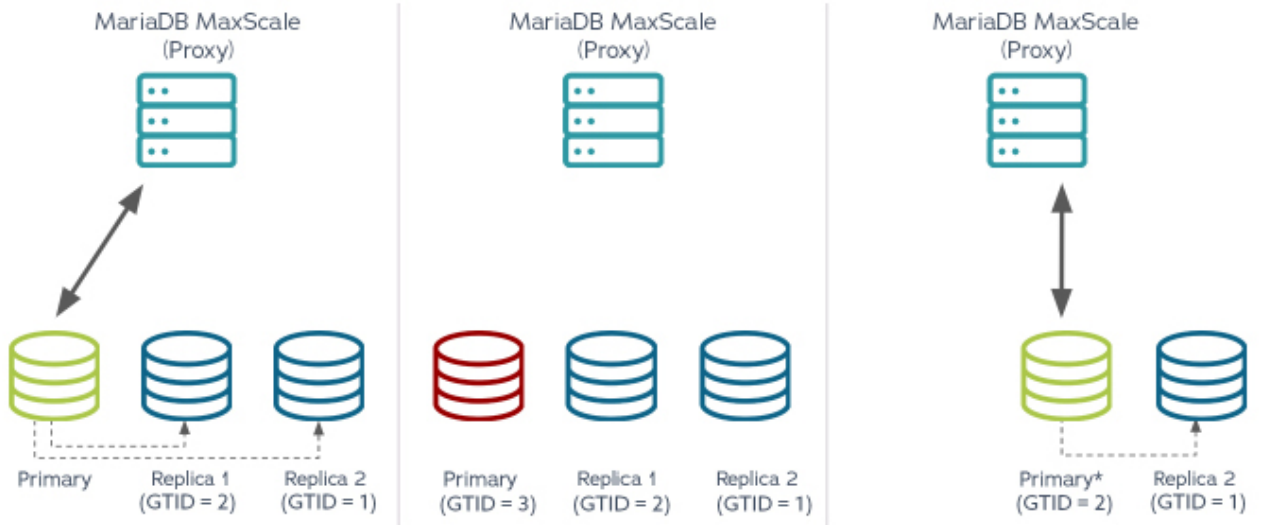


Diagrama 4: La función del GTID en el failover automático con replicación asincrónica

La replicación asíncrona se recomienda para cargas de trabajo de lectura intensiva o cargas de trabajo mixtas/de escritura intensiva donde se necesita el mayor rendimiento de escritura

Ejemplos:

- **Cargas de trabajo de lectura intensiva**: catálogos de productos, informes.
- **Cargas de trabajo mixtas**: carros de la compra, evaluaciones de clientes.
- **Cargas de trabajo de escritura intensiva**: datos de visitas, datos de sensores**.**

Replicación semisincrónica

Con la replicación semisincrónica, una transacción no se compromete hasta que se reproduce a una réplica. Afecta al rendimiento de la escritura, pero el efecto se minimiza al esperar que las transacciones se reproduzcan en una réplica en lugar de en todas las réplicas. Sin embargo, si el maestro falla y el *failover* atuomático se encuentra activado, no habrá pérdida de datos debido a que cada transacción se reprodujo a una réplica.

En el ejemplo siguiente, el *proxy* de la base de datos promovería la réplica 2 a primaria porque es la réplica con el GTID más alto. Con la replicación semisincrónica, no habría pérdida de datos dado que al menos una de las réplicas tendrá cada transacción escrita en su registro de relevo.

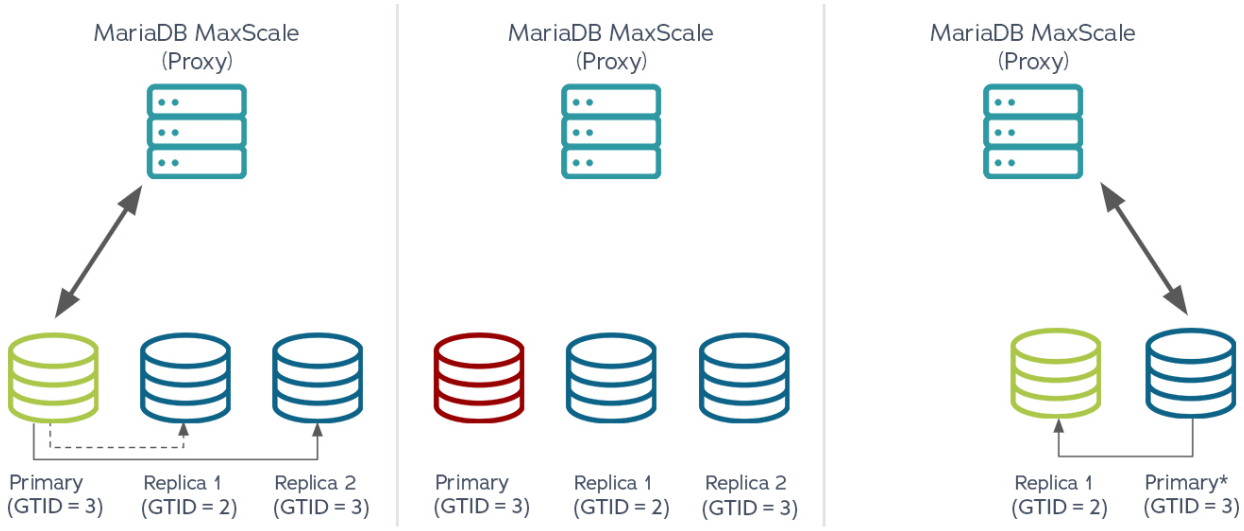


Diagrama 5: La función del GTID en el failover automático con replicación semisincrónica

Nota

El maestro esperará hasta 10 segundos (por defecto) para que una transacción se reproduzca a una réplica antes de volver a la replicación asíncrona. Si lo hace, y una de las réplicas se actualiza, el maestro restaurará la replicación semisincrónica. Si todas las réplicas son lentas, el tiempo de espera se puede reducir para mantener el rendimiento de escritura (aunque con menos durabilidad), o aumentar para mantener la durabilidad (aunque con menos rendimiento de escritura).

Variables del sistema

Variable	Valores	Predeterminado
rpl_semi_sync_master_enabled	0 (activado) | 1 (desactivado)	0
rpl_semi_sync_master_timeout	0 to n (ms, max: 18446744073709551615)	10000

La replicación semisincrónica se recomienda para cargas de trabajo mixtas/de escritura intensiva en las que se requiere un alto rendimiento de escritura y una fuerte durabilidad.

Ejemplos:

- **Cargas de trabajo mixtas: inventario**

Agrupamiento o clúster multimaestro

MariaDB Enterprise admite agrupamientos multimaestros a través de MariaDB Clúster (es decir, Galera Clúster). El nodo de origen asigna a una transacción un GTID y, durante la fase de confirmación, envía todas las filas modificadas por ella (es decir, las escrituras) a todos los nodos del agrupamiento, incluido él mismo. Si todos los nodos del agrupamiento aceptan las escrituras, el nodo de origen utiliza las escrituras y confirma la transacción. Los otros nodos utilizarán las escrituras y confirmarán la transacción de modo asincrónico.

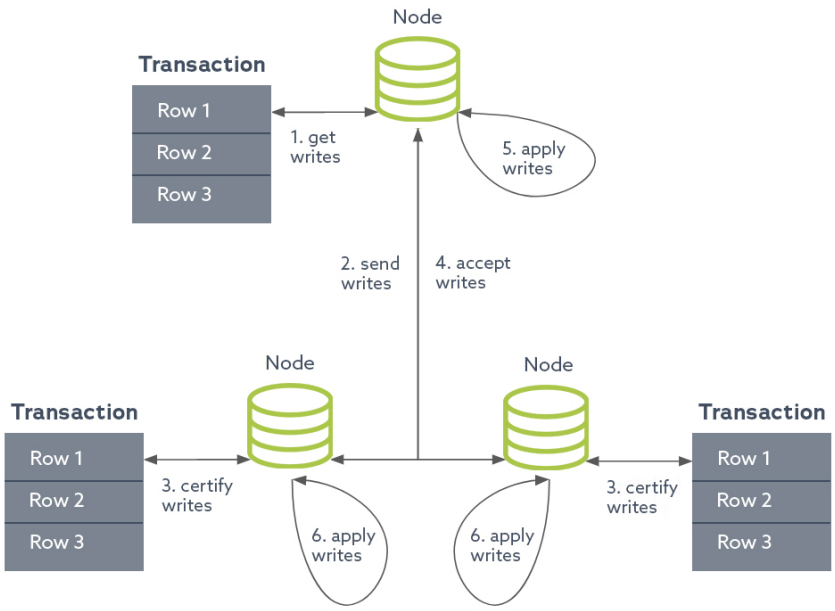


Diagrama 6: El proceso de agrupamiento en clúster multimaestros (replicación sincrónica)

Automatic failover

Si ocurre una falla en un nodo, el agrupamiento lo eliminará automáticamente y el *proxy* de la base de datos, MariaDB MaxScale, dejará de dirigir las consultas hacia él. Si el *proxy* de la base de datos dirigía lecturas y escrituras al nodo que había fallado, y dado que todos los nodos pueden aceptar lecturas y escrituras, el proxy de la base de datos seleccionará un nodo diferente y comenzará a dirigir lecturas y escrituras hacia él.

Replicación replicación

Con la replicación sincrónica, una transacción no se confirma hasta que sus cambios (es decir, las filas modificadas) se hayan reproducido en todos los nodos del agrupamiento. El rendimiento de escritura se encuentra limitado por el nodo más lento del agrupamiento. Sin embargo, si el nodo al que se dirigió una escritura falla, no habrá pérdida de datos dado que los cambios de cada transacción se replicaron en todos los nodos del agrupamiento.

En el ejemplo de abajo, el *proxy* de la base de datos estaría dirigiendo las lecturas y escrituras al Nodo 2 puesto que posee el valor de prioridad más bajo de los nodos restantes. No habría pérdida de datos dado que, con la replicación sincrónica, cada nodo posee los cambios de cada transacción.

Nota

El *proxy* de la base de datos puede configurarse para que el *failover* automático sea determinista (por ejemplo, basado en el valor de la prioridad) al establecer el parámetro use_priority como true en la configuración del monitor de Galera Cluster y el parámetro priority en las configuraciones del servidor de la base de datos.

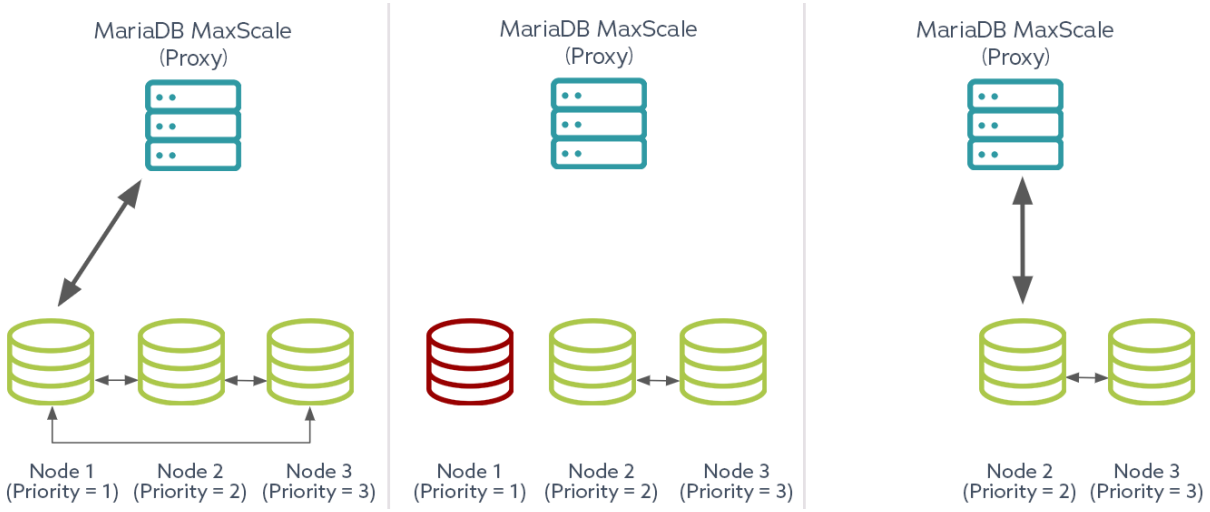


Diagram 7: Automatic failover with multi-primary clustering

La replicación sincrónica se recomienda para cargas de trabajo mixtas/de escritura intensiva en las que se requiere la mayor durabilidad.

Ejemplos:

- **Cargas de trabajo mixtas:** perfiles de clientes
- **Cargas de trabajo intensivas en escritura:** pagos

Recuperación transparente

El *failover* automático garantiza que la base de datos permanezca siempre disponible, pero no es transparente para las aplicaciones puesto que tienen que crear nuevas conexiones y reintentar las consultas/transacciones si el maestro falla. Sin embargo, el *proxy* de la base de datos, MariaDB MaxScale, incluye ahora opciones avanzadas para evitar que las aplicaciones se interrumpan por un servidor de base de datos que falle, con o sin *failover* automático.

Migración de la conexión

Cuando un servidor de base de datos falla o se vuelve inalcanzable/no disponible, el *proxy* de la base de datos cierra las conexiones de los clientes con él, y las aplicaciones deben crear otras nuevas. Sin embargo, el *proxy* de la base de datos puede ahora configurarse para *migrar* las conexiones al primario debido a la *failover* automático, de modo que las aplicaciones ya no deben crear una nueva conexión cuando falla un servidor de base de datos.

Migración de la sesión

El *proxy* de la base de datos almacena en la memoria caché los comandos de sesión, lo que le permite migrar las sesiones y recrearlas en un servidor de base de datos diferente. Por ejemplo, si se crea una sesión en una réplica y la réplica falla, el *proxy* de la base de datos reproducirá los comandos de la sesión en una réplica diferente para que no se interrumpa la aplicación. Al combinarse con la migración de conexiones, las conexiones y sesiones se migran automáticamente al fallar un servidor de base de datos, y es transparente para las aplicaciones.

Reintento retrasado

Si una aplicación envía una consulta de escritura a la base de datos despues de que falle el maestro, y antes de que el *failover* automático se complete, el *proxy* de la base de datos devolverá un error. Sin embargo, el *proxy* de la base de datos puede ahora configurarse para esperar y reintentar automáticamente la consulta. En efecto, espera a que se complete el *failover* automático y luego reintenta la consulta en nombre de la aplicación, por lo que ya no tiene que reintentar las consultas una vez falla el servidor de base de datos.

Nota

El *proxy* de la base de datos puede configurarse para dirigir las consultas de lectura a las réplicas cuando no hay un maestro, de modo que si el este falla y el *failover* automático no se completa, las consultas de lectura pueden aún ejecutarse.

Repetición de la transacción

Además, si se activa el *failover* automático después de que comience una transacción pero antes de que haya finalizado (es decir, una transacción en ejecución), el *proxy* de la base de datos puede configurarse para reproducir la transacción desde el principio en el nuevo maestro. Por ejemplo, si hubiera cinco inserciones en una transacción y el maestro fallara tras las dos primeras, el *proxy* de la base de datos esperaría a que se completara el proceso de *failover* automático y luego reproduciría las dos primeras inserciones en el nuevo maestro, y así permitiría que la transacción continúe.

Parámetros (MariaDB MaxScale)

Parámetros	Valores	Predeterminado
master_reconnection	True (activado) | False (desactivado)	False
master_failure_mode	fail_instantly (cerrar la conexión) fail_on_write (cerrar la conexión si recibe una consulta de escritura) error_on_write (devolver un error si se recibe una consulta de escritura)	fail_instantly
delayed_retry	True (activado) | False (desactivado)	False
delayed_retry_timeout	N (El tiempo que se debe esperar antes de devolver un error)	10
transaction_replay	True (activado) | False (desactivado)	False

LECTURA DE ADAPTABILIDAD

Replicación maestro/réplica

Si se utiliza la replicación maestro/réplica tanto para alta disponibilidad como para escalabilidad de lecturas, el *proxy* de la base de datos, MariaDB MaxScale, puede dirigir las escrituras al maestro y equilibrar la carga de las lecturas a través de las réplicas al utilizar la división de lectura-escritura.

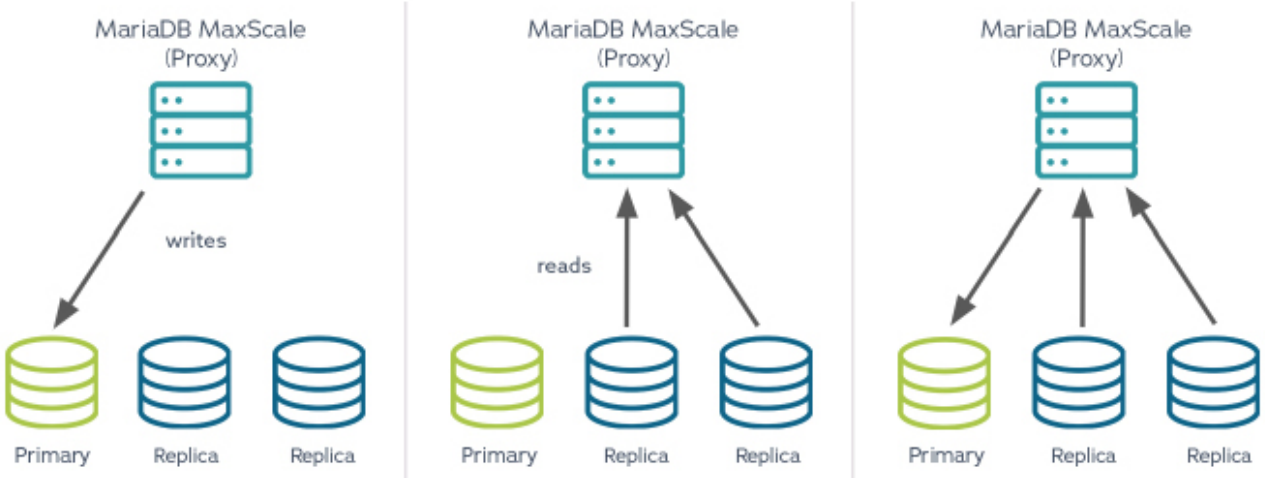


Diagrama 8: División de lectura-escritura con replicación maestro/replica

Sin embargo, se deben considerar los requisitos de integridad. Con replicación maestro/replica (asincrónica o semisincrónica), la consulta de las réplicas puede devolver lecturas obsoletas si existen transacciones pendientes de replicar, y luego de que las transacciones se hayan reproducido y persistido en el registro de retransmisión en una o más réplicas, todavía tienen que ejecutarse.

Si se requieren lecturas consistentes, debe activarse el filtro de lecturas críticas consistentes (CCR) en el *proxy* de la base de datos. Si existe un escrito, el filtro CCR se activará, y de ese modo, el *proxy* de la base de datos dirige todas las lecturas posteriores al maestro durante 60 segundos (por defecto). Si el retraso en la replicación es superior a 60 segundos (es decir, se tarda más de 60 segundos en reproducir las escrituras a las réplicas), se puede aumentar el tiempo.

Consejo

El filtro CCR puede configurarse para que se active por escrituras específicas y no por todas las escrituras.

Parámetros

Variable	Valores	Predeterminado
time	El tiempo para dirigir las lecturas al maestro (segundos)	60
count	El número de lecturas que deben dirigirse al maestro	-
match	Una expresión regular para determinar si una escritura debe activar el enrutamiento	-
ignore	Una expresión regular para determinar si una escritura no debe activar el enrutamiento	-

Como alternativa, se pueden habilitar las lecturas causales en el *proxy* de la base de datos. Al activarse las lecturas causales, el *proxy* de la base de datos modificará la consulta, y obligará a la base de datos a esperar hasta que haya alcanzado al cliente antes de ejecutar la consulta. Aprovecha los identificadores de transacciones globales (GTID). Por ejemplo, si el GTID de la escritura más reciente del cliente fuera 2000, la base de datos esperaría hasta que su GTID alcanzara o superara los 2000 antes de ejecutar la consulta.

Nota

El contexto de las lecturas causales son los clientes, no las aplicaciones.

Parámetros (MariaDB MaxScale)

Parámetros	Valores	Predeterminado
causal_reads	True (activado) | False (desactivado)	False
causal_reads_timeout	N (El número de segundos a esperar antes de dirigir al master)	10

Los clúster

Si se utiliza un clúster multimaestro para una alta disponibilidad y escalabilidad de lectura, el *proxy* de la base de datos puede asignar el funcionamiento de maestro a un único nodo (y dirigir las escrituras hacia él) mientras asigna el rol de réplica a los nodos restantes (y equilibra la carga de lecturas entre ellos). En los clúster multimaestros el enrutamiento de todas las escrituras al mismo nodo evita los bloqueos y los conflictos de escritura.

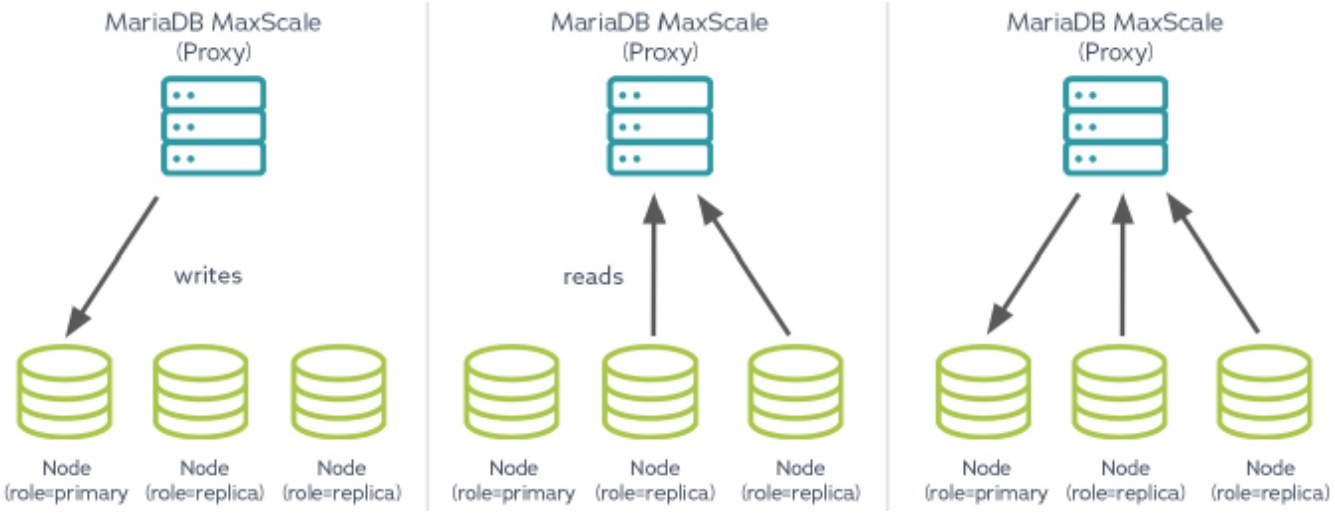


Diagrama 9: División de lectura-escritura en un clúster multimaestro

Es posible que los nodos con el rol de réplica devuelvan lecturas antiguas porque mientras los conjuntos de escritura (es decir, las transacciones) se replican sincrónicamente, los primeros se aplican asincrónicamente. Existe un pequeño retraso entre la recepción de un conjunto de de escritura y su aplicación.

Sin embargo, se pueden evitar las lecturas antiguas al establecer la variable del sistema wsrep_sinc_espera a 1, y forzar a los nodos a retrasar una lectura hasta que se hayan aplicado todos los conjuntos de escritura. El impacto en la latencia de lectura es mínimo, y asegura que cualquier dato escrito en un nodo con el rol primario se puede leer inmediatamente desde cualquier nodo con el rol de réplica.

Nota

La variable del sistema wsrep_sinc_espera se puede establecer para especificar los tipos de consultas que deben retrasarse (por ejemplo, READ) hasta que se hayan aplicado todos los conjuntos de escritura.

Variables del sistema

Variable	Valores	Predeterminado
wsrep_sync_wait	0 (DESACTIVADO) 1 (LECTURA) 2 (ACTUALIZAR y ELIMINAR) 3 (LECTURA, ACTUALIZAR y ELIMINAR) 4 (INSERTAR y SUSTITUIR) 5 (LECTURA, INSERTAR y SUSTITUR) 6 (ACTUALIZAR, ELIMINAR, INSERTAR y SUSTITUIR) 7 (LECTURA, ACTUALIZAR, ELIMINAR, INSERTAR y SUSTITUIR) 8 (MOSTRAR), 9-15 (1-7 + MOSTRAR)	0

TOPOLOGÍAS AVANZADAS

Centros de datos múltiples

Replicación maestro/replica

En el ejemplo siguiente, se puede utilizar la replicación circular (por ejemplo, la replicación bidireccional) para sincronizar los datos entre un centro de datos activo (DC1) y un centro de datos pasivo (DC2). El maestro de DC1 se encuentra configurado como una réplica del maestro en DC2. El maestro de DC2 se encuentra configurado como una réplica del primario de DC1. Si. DC1 falla, las aplicaciones pueden conectarse al *proxy* de la base de datos, MariaDB MaxScale, en DC2.

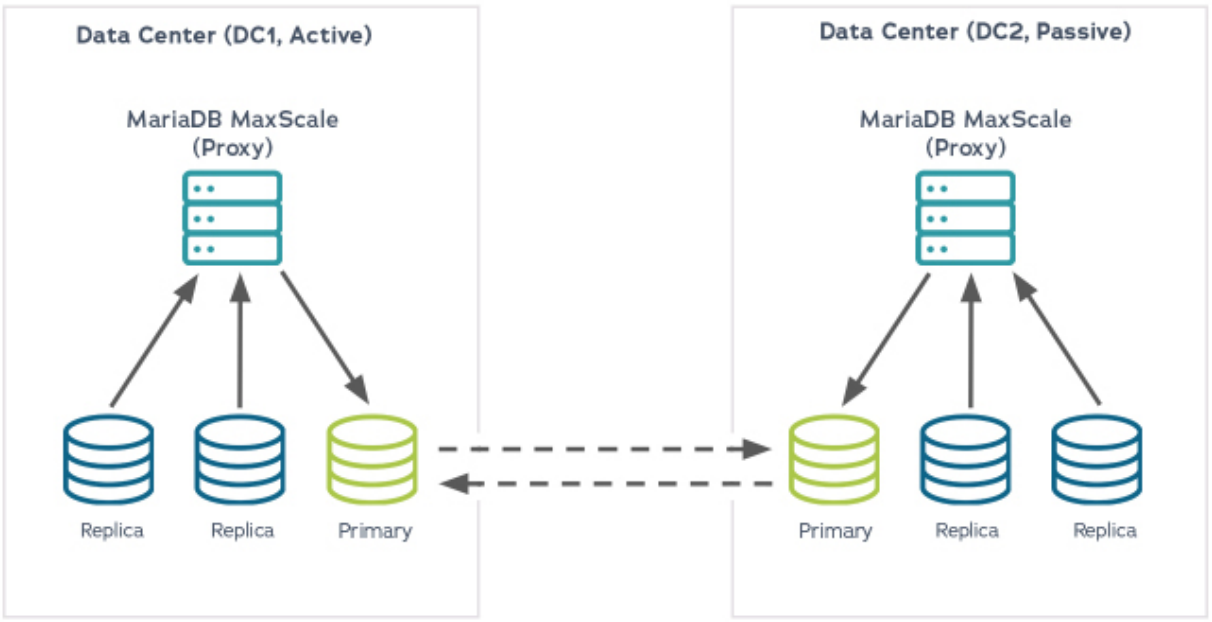


Diagrama 10: Replicación bidireccional y asíncrona entre centros de datos

Los clúster

En el ejemplo siguiente, se despliega un clúster de tres nodos en dos centros de datos con dos nodos en el centro de datos activo (DC1) y un nodo en el centro de datos pasivo (DC2). La configuración para el *proxy* de la base de datos en DC1, proxy 1, asigna un valor de prioridad de 1 al nodo 1, 2 al nodo 2 y 3 al nodo 3. Asigna el papel de primario al nodo 1 pues posee el valor de prioridad más bajo. El proxy 1 utiliza un *router* básico para dirigir todas las lecturas y escrituras a cualquier nodo al que se le haya asignado el rol de primario, el nodo 1 por defecto. Si el nodo 1 falla, el proxy 1 asignará el papel de primario al nodo 2, en el mismo centro de datos, DC1.

Si DC1 falla, las aplicaciones pueden conectarse al proxy de la base de datos en DC2, proxy 2. La configuración para el proxy 2 asigna un valor de prioridad de 1 al nodo 3, 2 al nodo 2 y 3 al nodo 1. Asigna el papel de primario al nodo 3 porque tiene el valor de prioridad más bajo El proxy 2 utiliza un *router* básico para dirigir todas las lecturas y escrituras a cualquier nodo al que se le haya asignado el rol primario, el Nodo 3 por defecto.

Nota

Un clúster puede desplegarse en varios centros de datos si a) hay suficiente ancho de banda de red entre ellos para minimizar la latencia y b) los conjuntos de escritura son pequeños (es decir, las transacciones para no cambiar muchas filas).

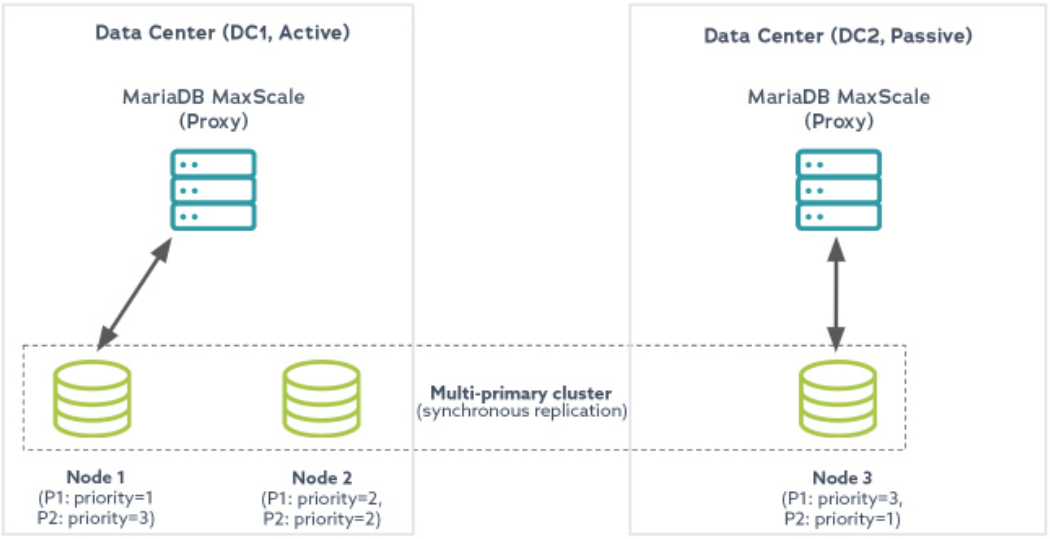


Diagrama 11: Clúster multimaestros entre varios centros de datos

Lectura de adaptabilidad

Replicación maestro/replica

En el ejemplo siguiente, se configura un segundo *proxy* de base de datos y se utiliza como servidor binario para transmitir las transacciones desde el maestro a muchas réplicas para adaptar la lectura. El servidor binario reduce la sobrecarga de replicación en el maestro: en vez de que muchas réplicas reproduzcan desde el maestro, un único servidor binario reproduce desde él.

El maestro se configura para la replicación semisincrónica, para alta disponibilidad y durabilidad, con la réplica M1 y la réplica M2. Para la replicación asincrónica, para adaptabilidad de lectura, se configura con el servidor binario. El *proxy* de la base de datos se configura con dos *routers*, uno para cada agrupamiento, y cada *router* tiene un puerto diferente. El primero dirigirá todos los escritos al maestro en el clúster 1. El segundo dirigirá todas las lecturas a las réplicas del clúster 2 (réplica R2 a réplica R100).

Nota

La aplicación tiene que utilizar diferentes conexiones para la división de lectura y escritura que se basa en el puerto.

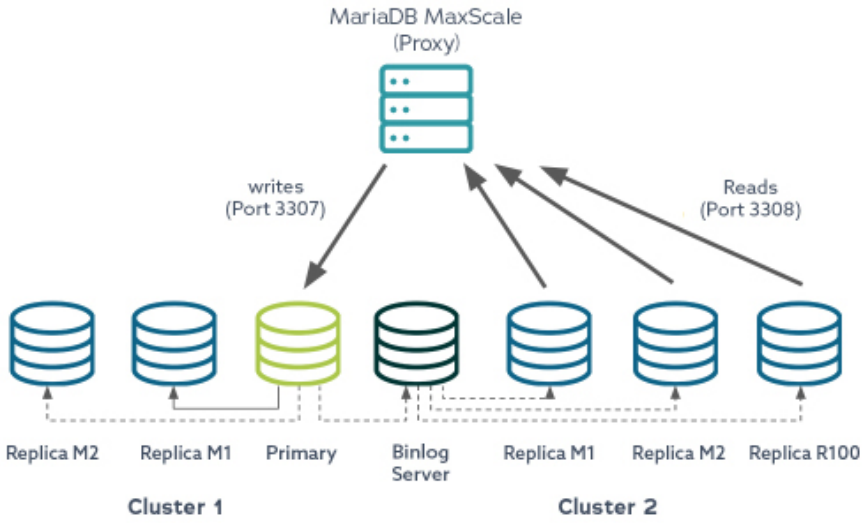


Diagram 12: Extreme read scaling with a binlog server

Copia de seguridad dedicada

Replicación maestro/replica

En el ejemplo siguiente, una de las réplicas se encuentra dedicada a las copias de seguridad; el *proxy* de la base de datos no dirige las lecturas hacia esta. Sin embargo, los administradores de base de datos ejecutan MariaDB Backup en aquella para crear copias de seguridad. Al dedicar una de las réplicas a las copias de seguridad y solo a las copias de seguridad, la carga de trabajo en el maestro y en las réplicas restantes no se interrumpirá al crear una copia de seguridad.

El primario se configura para la replicación semisincrónica con la réplica 2 y la réplica 3 para una alta disponibilidad y durabilidad, y la replicación asincrónica con la réplica 1 para las copias de seguridad.

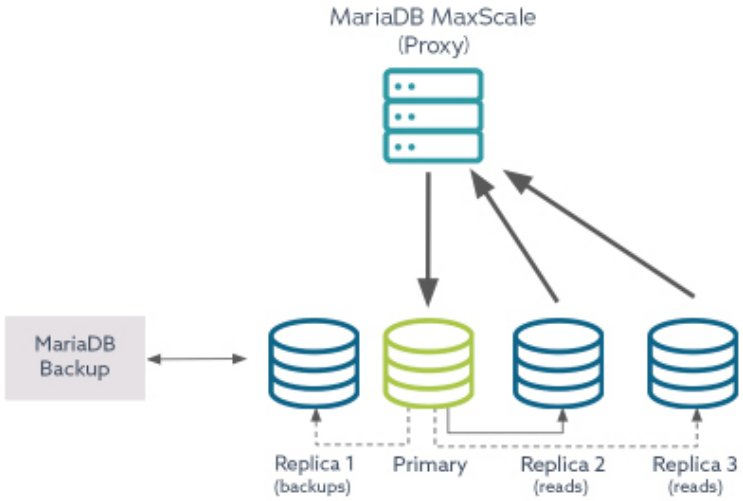


Diagrama 13: Dedicar una réplica para las copias de seguridad

CUESTIONES INTERNAS

Replicación maestro/réplica

Componentes

Registro binario (*binlog*)

El primario registra los cambios que realizan las sentencias DML y DDL, y su tiempo de ejecución, como eventos en su registro binario. El registro binario se compone de archivos de registro y un archivo de índice. Mientras que los eventos del registro binario se almacenan en formato binario, el registro binario mysql puede mostrarlos como texto.

Confirmación de grupo

Por defecto, el primario llama a fsync() para vaciar las escrituras del registro binario al disco durante la fase de confirmación de una transacción. Sin embargo, cuando aparecen transacciones paralelas, se puede utilizar una confirmación de grupo para vaciar las escrituras del registro binario de múltiples

Variables del sistema

Variable	Values	Default
sync_binlog	0 (posponer a OS) n (fsync cada n transacciones)	0

Formato

El registro binario soporta tres formatos de registro: basado en sentencias, basado en filas y mixto (predeterminado). En el formato basado en sentencias, las sentencias se almacenan. En el formato basado en filas, se almacenan las filas que se modificaron por una sentencia. En el formato mixto, las sentencias se almacenan por defecto, pero si una sentencia no es segura para la replicación (por ejemplo, no es determinista), las filas que se modificaron se almacenan en su lugar y de ese modo, se logra que sea seguro utilizar funciones no deterministas con la replicación.

Consejo

El formato puede cambiarse a basado en filas para mejorar el rendimiento de la replicación si muchas sentencias dan lugar a cambios en un pequeño número de filas o tardan mucho en ejecutarse.

Variables del sistema

Variable	Valores	Predeterminado
binlog_format	STATEMENT | ROW | MIXED	MIXED

Cifrado y compresión

Los eventos del registro binario pueden ser cifrados para proteger los datos sensibles o comprimidos mediante zlib para reducir el IO de disco y de red. Cuando la compresión del registro binario se encuentra activada, los eventos individuales del registro binario se comprimen antes de ser escritos en él y permanecerán comprimidos al reproducirse. El hilo de réplica IO descomprimirá los eventos del registro binario antes de que se escriban en su registro de transmisión.

Nota

Si la compresión del registro binario se encuentra activada, no se comprimirán todos los eventos del registro binario. Depende de la longitud del evento (sentencia o fila) y de la longitud mínima que se configura para la compresión.

Variables del sistema

Variable	Valores	Predeterminado
encrypt-binlog	0 (DESACTIVADO) 1 (ACTIVADO)	0 (DESACTIVADO)
log_bin_compress	0 (DESACTIVADO) 1 (ACTIVADO)	0 (DESACTIVADO)
log_bin_compress_min_len	10 - 1024	256

Identificadores de transacciones globales (GTIDs)

El maestro agrupa, ordena y reproduce los eventos del registro binario al utilizar GTIDs. Los GTID se componen de tres números separados por un guión: el ID del dominio, un entero sin signo de 32 bits; el ID del servidor, un entero sin signo de 32 bits; y la secuencia, un entero sin signo de 64 bits. El identificador resultante garantiza que las transacciones sean únicas en varias instancias de la base de datos, lo que permite la replicación de múltiples fuentes y la replicación circular (es decir, la replicación maestro/replica réplica).

Los GTID permiten a las réplicas continuar leyendo los eventos del registro binario al cambiar el primario. Por ejemplo, si una réplica se promueve a primaria, el resto de las réplicas pueden ser reconfiguradas para continuar leyendo los eventos del registro binario desde aquella y desde donde lo dejaron, su GTID actual.

Asimismo, los GTID permiten que las réplicas mantengan un estado de replicación duradero y consistente. La tabla del sistema mysql.gtid_replica_pos es donde las réplicas almacenan su GTID actual. Si esta tabla se almacena en un motor de almacenamiento transaccional (por ej., InnoDB) se utilizará una única transacción para actualizar su GTID actual y ejecutar los eventos del registro binario asociados a esta.

Visión lógica del registro binario

ID de confirmación	GTID	ID del servidor	Tipo de evento	Posición	Posición final
100	0-1-200	1	Consulta	0	150
100	0-1-201	1	Consulta	151	500
100	0-1-202	1	Consulta	501	600
101	0-2-203	1	Consulta	601	800

Proceso

1. El hilo de réplica IO solicita eventos del registro binario, incluye su GTID actual.
2. El maestro devuelve los eventos del registro binario para los siguientes GTID.
3. El hilo de réplica IO escribe los eventos del registro binario en su registro de retransmisión.
4. El hilo SQL de la réplica lee los eventos del registro binario de su registro de retransmisión.
5. El hilo de réplica SQL ejecuta los eventos del registro binario y actualiza su GTID actual.

Clúster multimaestro

Componentes

MariaDB Clúster

MariaDB Cluster, basado en Galera Cluster, utiliza la comunicación en grupo, el ordenamiento global de las transacciones, los conjuntos de escritura y la certificación para la replicación sincrónica.

Comunicación de grupo

El clúster se basa en la comunicación de grupo. Permite que los nodos se unan automáticamente al clúster y que este elimine automáticamente los nodos fallidos. En el contexto de la replicación, la comunicación de grupo garantiza el ordenamiento total de los mensajes enviados desde múltiples nodos.

Conjuntos de escritura

Un conjunto de escritura contiene todas las filas modificadas por una transacción (y sus claves primarias), y se crea durante la fase de confirmación. Se reproduce a todos los nodos, incluido el nodo de origen, a través de la comunicación de grupo.

Ordenación global de las transacciones

Cuando se reproduce un conjunto de escritura, se le asigna un GTID, lo que garantiza que los conjuntos de escritura (y,

por tanto, las transacciones) se ejecutan en el mismo orden en cada nodo. Se compone de un UUID y un número de secuencia (entero con signo de 64 bits) separados por dos puntos.

Certificación

El conjunto de escritura se certificará en cada nodo al utilizar su GTID y las claves primarias de las filas modificadas por la transacción. Si el conjunto de escritura supera la prueba de certificación, se aplica y se confirma la transacción. Si no lo hace, el conjunto de escritura se descarta y la transacción se revierte.

Proceso

1. Sincrónico
    a. Nodo de origen: crea un conjunto de escritura.
    b. Nodo de origen: asigna un ID de transacción global al conjunto de escritura y lo reproduce.
    c. Nodo de origen: aplica el conjunto de escritura y confirma la transacción.
2. Asincrónico
    a. Otros nodos: certifican el conjunto de escritura.
    b. Otros nodos: aplican el conjunto de escritura y confirman la transacción.

Mientras que los conjuntos de escritura se reproducen de modo sincrónico, se certifican y aplican de modo asincrónico. Sin embargo, la certificación es determinista: tiene éxito en cada nodo o falla en cada nodo. Así, una transacción se compromete en cada nodo o se revierte en cada nodo.

OPTIMIZACIONES DE RENDIMIENTO

Replicación

Replicación paralela

Los hilos IO y SQL en las réplicas son responsables de reproducir los eventos del registro binario. El hilo IO en las réplicas solicita eventos del registro binario del maestro y los escribe en el registro de retransmisión. El hilo SQL en las réplicas lee los eventos del registro binario del registro de retransmisión y los ejecuta, uno a la vez. Sin embargo, cuando se activa la replicación paralela, la réplica utilizará un grupo de hilos de trabajo para ejecutar múltiples eventos del registro binario al mismo tiempo.

Por *defecto*, las réplicas ejecutan los eventos del registro binario en orden. Existen dos modos: conservador (por *defecto*) y optimista. En el modo conservador, la replicación paralela se limita a una confirmación de grupo. Si varias transacciones poseen el mismo ID de confirmación, se ejecutaron en paralelo en el maestro y se ejecutarán en paralelo en las réplicas. Sin embargo, las transacciones se confirmarán en orden. Si dos transacciones no poseen el mismo id de confirmación, la segunda transacción no se ejecutará hasta que la primera transacción se encuentre en la fase de confirmación, lo que garantiza que las transacciones se confirmen en el mismo orden en que las confirmó el maestro.

En el modo optimista, se ejecutarán múltiples transacciones en paralelo sin importar el ID de confirmación. Sin embargo, dentro de las excepciones están incluidas las sentencias DDL, las sentencias DML no transaccionales y las transacciones en las que se ejecutó una espera de bloqueo de fila en el *maestro*. Si se detecta un conflicto entre dos transacciones (por ejemplo, dos transacciones intentan actualizar la misma fila), la primera transacción se confirmará y la segunda transacción se revertirá y se volverá a intentar.

Nota

La variable del sistema replica-parallel-threads debe tener un valor superior a 0 para activar la replicación paralela. Además, el maestro se puede configurar para esperar *n* microsegundos o *n* transacciones, antes de una confirmación de grupo. Si hay muchas transacciones concurrentes en el maestro, y el número de hilos en las réplicas es lo suficientemente alto, el aumento de la espera puede reducir el IO del disco de registro binario en el maestro y permitir a las réplicas confirmar las transacciones con mayor rapidez.

Consejo

Si existen pocos conflictos, se puede cambiar el modo a optimista para mejorar el rendimiento.

Variables del sistema

Variable	Valores	Predeterminado
replica-parallel-mode	optimistic | conservative | aggressive | minimal | none	conservative
replica-parallel-threads	0 - n (max: 16 383)	0
binlog_commit_wait_count	0 - n (max: 18446744073709551615)	0
binlog_commit_wait_usec	0 - n (max: 18446744073709551615)	100 000

Aceleración de la lectura

El primario puede limitar la replicación de registros binarios y limitar el ancho de banda de la replicación, reducir la carga del maestro al añadir varias réplicas o cuando varias réplicas intentan reproducir muchos eventos del registro binario al mismo tiempo.

Variables del sistema

Variable	Valores	Predeterminado
read_binlog_speed_limit	0 (unlimited) - n (kb, max: 18446744073709551615)	0

Los clúster

Registro InnoDB asincrónico

InnoDB se vacía en el disco al confirmar una transacción (*por defecto*). Sin embargo, con el agrupamiento, las transacciones se tornan duraderas mediante la replicación sincrónica. El nodo de origen puede fallar antes de que el registro de InnoDB se vacíe en el disco y la transacción continuará confirmándose en los otros nodos. Al escribir y volcar los registros de InnoDB al disco de modo asincrónico, se puede mejorar el rendimiento de escritura.

Variables del sistema

Variable	Valores	Predeterminado
innodb_flush_log_at_trx_commit	0 (escritura y descarga una vez por segundo) 1 (escritura y descarga durante la confirmación) 2 (escritura durante la confirmación, descarga una vez	1

CONCLUSIÓN

MariaDB Enterprise admite múltiples estrategias de alta disponibilidad para adaptarse a las garantías de rendimiento, durabilidad y integridad de los casos de uso individuales, e incluye el *proxy* de base de datos más avanzado del mercado para simplificar y mejorar la alta disponibilidad, reduciendo aún más tanto el impacto en el negocio como la carga administrativa del tiempo de inactividad de la infraestructura, planificado o no.

Tanto si se configura con replicación maestro/replica asincrónica para simplificar y mejorar el rendimiento, como si se configura con clúster multimaestros para garantizar la disponibilidad y la integridad, o algo intermedio, MariaDB Enterprise tiene tanto la flexibilidad como la capacidad de satisfacer los complejos requisitos de la empresa en cada caso.

MariaDB combina el liderazgo en ingeniería y la innovación de la comunidad para crear soluciones de base de datos de código abierto para aplicaciones modernas, satisfaciendo las necesidades actuales y dando soporte a las posibilidades del futuro sin sacrificar el SQL, el rendimiento, la fiabilidad o la seguridad.