Aurora a hiperescala MySQL-Compatible para gestionar el crecimiento repentino del tráfico

AWS Orientación prescriptiva

AWS Orientación prescriptiva: Aurora a hiperescala MySQL-Compatible para gestionar el crecimiento repentino del tráfico

Table of Contents

Hiperescalar Aurora MySQL-Compatible para administrar el crecimiento repentino del tráfico

Oliver Francis, Shyam Sunder Rakhecha y Vikram singh Rai, Amazon Web Services (AWS)

Octubre de 2022

Su negocio en Internet podría enfrentarse a un panorama de hipercrecimiento sin precedentes, panorama que su infraestructura de aplicaciones actual no es capaz de administrar. Esto puede darse por varios factores, como un anuncio sobre su producto que genere un interés superior al esperado o un cambio repentino en los patrones de compra de los clientes, que pasan de ser presenciales a comprar en línea.

Para hacer frente a estas cargas inesperadas, debe hiperescalar su infraestructura. Escalar el nivel de la aplicación implica sumar más servidores o módulos. Sin embargo, lo más difícil para lograr el hiperescalado es la base de datos. Puede escalar la instancia de la base de datos al tamaño de instancia más grande disponible, pero es posible que eso no resuelva el problema.

Si ejecuta la carga de trabajo de su base de datos en una edición de Amazon Aurora MySQL-Compatible Edition, esta guía ofrece recomendaciones que puede implementar para hiperescalar su base de datos y hacer frente a un hipercrecimiento repentino. No todas estas recomendaciones son prácticas recomendadas a largo plazo. Si prevé un hipercrecimiento y tiene tiempo para planificar cómo abordarlo, consulte las publicaciones de blog que figuran en la sección Recursos. Esas publicaciones pueden ayudarlo a implementar las prácticas recomendadas a largo plazo. Por otro lado, si se enfrenta a un hipercrecimiento inesperado, esta guía lo ayudará a administrar su carga y a lograr una estabilidad razonable mientras planifica e implementa soluciones a largo plazo para su empresa.

Tenga en cuenta que las recomendaciones que se describen en esta guía requerirán la ayuda de su equipo de desarrollo para implementarlas.

Resultados empresariales específicos

Los enfoques que se incluyen en esta guía lo ayudarán a hacer lo siguiente:

- Estabilice su negocio en caso de un hipercrecimiento inesperado. Logre la estabilidad suficiente para implementar las prácticas recomendadas a largo plazo en materia de hipercrecimiento.

- Evite pérdidas financieras. Las interrupciones repentinas en un entorno hiperescalado pueden provocar una disminución de las transacciones comerciales que los clientes realizan en su aplicación. En algunos casos, esto puede provocar pérdidas financieras sustanciales. Un entorno de hiperescala estabilizado es clave para evitar interrupciones prolongadas que provoquen una pérdida de negocio.

Administración de conexiones

A medida que crece la demanda de su aplicación, aumenta el tráfico de frontend. En un escenario típico, se configura el escalado automático en el nivel de la aplicación para manejar esa ráfaga de tráfico entrante. Como resultado, el nivel de aplicación comienza a escalarse automáticamente y se agregan más servidores de aplicaciones (instancias) para hacer frente al aumento del tráfico. Como todos los servidores de aplicaciones tienen una configuración de grupo de conexiones de base de datos preconfigurada, el número de conexiones entrantes a la base de datos aumenta en proporción a las instancias recién implementadas.

Por ejemplo, 20 servidores de aplicaciones configurados con 200 conexiones de base de datos cada uno abrirían un total de 4000 conexiones de base de datos. Si el grupo de aplicaciones se escala verticalmente hasta 200 instancias (por ejemplo, durante las horas pico), el número total de conexiones alcanzará las 40 000. En una carga de trabajo típica, es probable que la mayoría de estas conexiones estén inactivas. Sin embargo, el aumento de las conexiones podría limitar la capacidad de escalar de la base de datos de Amazon Aurora MySQL-Compatible Edition. Esto se debe a que incluso las conexiones inactivas utilizan memoria y otros recursos del servidor, como los descriptores de archivos. Aurora, compatible con MySQL, suele utilizar menos memoria que MySQL Community Edition para mantener el mismo número de conexiones. Sin embargo, el uso de memoria para las conexiones inactivas aún no es cero.

Variables de configuración

Puede controlar el número de conexiones entrantes permitidas a la base de datos con dos variables principales de configuración del servidor: `max_connections` y `max_connect_errors`.

Variable de configuración max_connections

La variable de configuración `max_connections` limita el número de conexiones a bases de datos para cada instancia de MySQL. La práctica recomendada es establecerlo un poco por encima del número máximo de conexiones que se espera abrir en cada instancia de base de datos.

Si también ha activado `performance_schema`, tenga mucho cuidado con la configuración de `max_connections`. Las estructuras de memoria del esquema de rendimiento se dimensionan automáticamente en función de las variables de configuración del servidor, entre las que se incluyen `max_connections`. Cuanto más alta establezca la variable, más memoria utilizará Performance Schema. En casos extremos, esto puede provocar out-of-memory problemas en tipos de instancias

más pequeñas. Tenga en cuenta que al habilitar Performance Insights, se habilitará Performance Schema automáticamente.

Variable de configuración max_connect_errors

La variable de configuración `max_connect_errors` determina cuántas solicitudes de conexión interrumpidas sucesivas se permiten desde un host cliente determinado. Si el host cliente supera el número especificado de intentos de conexión sucesivos fallidos, el servidor lo bloquea. Los intentos de conexión posteriores de ese cliente producen un error.

```
Host 'host_name' is blocked because of many connection errors. Unblock with
'mysqladmin flush-hosts'
```

Si se produce el error «el host está bloqueado», evite aumentar el valor de la `max_connect_errors` variable. En cambio, investigue los contadores de diagnóstico del servidor en la variable de estado `aborted_connects` y la tabla `host_cach`. Utilice la información recopilada para identificar y corregir los clientes que tengan problemas de conexión. Además, tenga en cuenta que este parámetro no tiene ningún efecto si `skip_name_resolve` está configurado en `1` (predeterminado).

Consulte el manual de referencia de MySQL para obtener más información sobre lo siguiente:

- Variable max_connect_errors
- Error «El host está bloqueado»
- Variable de estado aborted_connect
- Tabla Host_cache

Implementar la agrupación de conexiones

Un evento de escalado podría agregar más servidores de aplicaciones, lo que, a su vez, podría provocar que el servidor de base de datos supere el número de conexiones activas completamente cargadas. La adición de un grupo de conexiones o una capa de proxy entre los servidores de aplicaciones y la base de datos actúa como un cuello de botella, ya que reduce el número total de conexiones de la base de datos. El objetivo principal de un proxy es volver a utilizar las conexiones de bases de datos mediante la multiplexación.

Por un lado, el proxy se conecta a la base de datos con un número controlado de conexiones. Por otro lado, el proxy acepta conexiones de aplicaciones. También ofrece características adicionales,

como el almacenamiento en caché de consultas, el almacenamiento en búfer de conexiones, la reescritura y el enrutamiento de consultas y el equilibrio de carga. La capa del grupo de conexiones debe configurarse para mantener el número máximo de conexiones a la base de datos por debajo del número de conexiones completamente cargadas. Amazon RDS Proxy es un proxy totalmente administrado que puede implementar para este fin. Amazon RDS Proxy no requiere cambios de código para la mayoría de las aplicaciones y no es necesario administrar ninguna infraestructura adicional para implementar la solución.

También puede explorar los siguientes proxy de terceros que se pueden usar con Aurora MySQL-Compatible:

- ProxySQL
- MariaDB MaxScale
- ScaleARC
- Heimdall Data

Evite las tormentas de conexiones

Tenga en cuenta cómo se comporta su grupo de conexiones en caso de que una base de datos esté sobrecargada o una réplica quede demasiado alejada del nodo principal. Al configurar el servidor proxy o los grupos de conexiones, asegúrese de no restablecer todo el grupo de conexiones en función de las respuestas lentas de la base de datos (causadas por problemas de equipo o almacenamiento o por limitaciones de recursos de la base de datos).

El inicio repentino de cientos de conexiones genera una tormenta de conexiones porque un gran número de solicitudes de nuevas conexiones a la base de datos se inician todas al mismo tiempo. La tormenta consume muchos recursos. Crear una nueva conexión de base de datos en MySQL es una operación costosa, porque el backend intercambia varios paquetes de red para la conexión inicial, genera un nuevo proceso, asigna memoria, gestiona la autenticación, etc. Si se recibe una gran cantidad de solicitudes en un periodo corto, puede parecer que la base de datos no responde.

MySQL tiene un mecanismo para protegerse contra este aumento en las solicitudes de conexión. La variable `back_log` se puede establecer en el número de solicitudes que se pueden acumular durante un breve periodo antes de que MySQL deje de responder de forma temporal a las nuevas solicitudes. El valor lo impone un subproceso de administración de conexiones, que a su vez podría verse abrumado por una tormenta de conexiones. Para obtener más información, consulte el MySQL Reference Manual.

Si la conexión está configurada para restablecerse cuando la base de datos es lenta, iniciará el ciclo una y otra vez. Del mismo modo, si prevé un aumento repentino del tráfico de la base de datos en determinados momentos del día (por ejemplo, al abrir el mercado de valores), prepare previamente el grupo de conexiones para no tener que abrir muchas conexiones cuando se inicia una gran carga de tráfico.

Ajustar la carga de trabajo de consultas

Una carga de trabajo bien ajustada le permitirá lograr una solución estable para el hipercrecimiento. Si su carga de trabajo no está bien ajustada, independientemente de la potencia del clúster de Amazon Aurora que utilice, se encontrará con cuellos de botella que perjudicarán el rendimiento y la experiencia de usuario de su aplicación. La práctica recomendada es contar desde el principio con un proceso que lo ayude a identificar y ajustar las consultas problemáticas en el sistema.

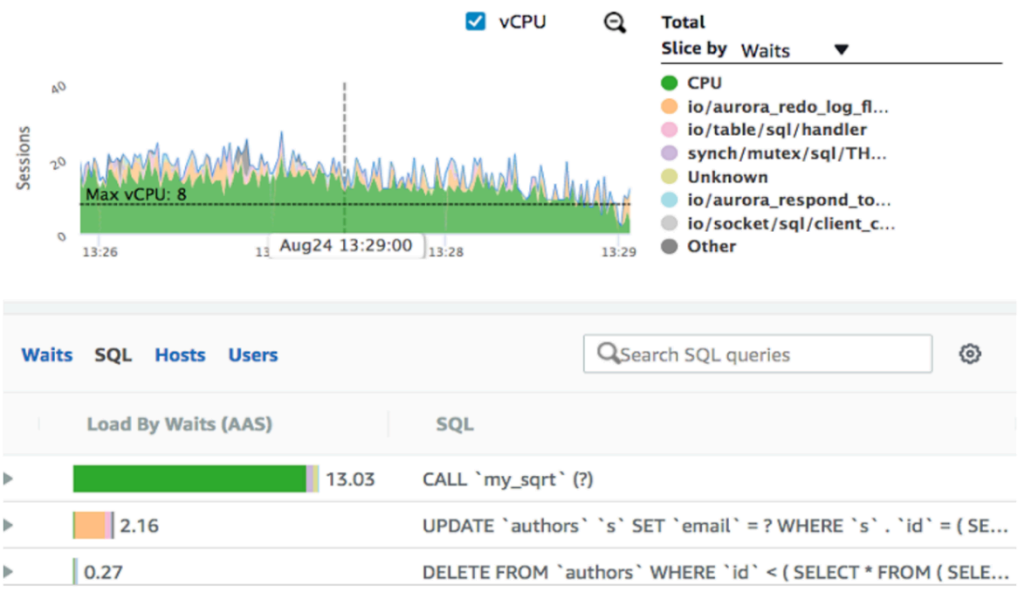
Realizar un seguimiento y ajustar las consultas problemáticas de la carga de trabajo

Cuando se enfrenta a un hipercrecimiento, tener una carga de trabajo bien ajustada es tener la mitad de la batalla ganada. Para comprender la naturaleza de las cargas de trabajo en tiempo real y los problemas de rendimiento a los que se enfrenta su clúster de Aurora, asegúrese de que su equipo recopile las consultas problemáticas de las instancias de escritor y lector. Estas consultas problemáticas deben ajustarse para que la carga de trabajo se ejecute en un estado óptimo. Amazon Aurora MySQL-Compatible Edition le ofrece dos formas de lograrlo:

- Información de rendimiento
- Registros de consultas lentas

Uso de Performance Insights

Performance Insights realiza un seguimiento de la carga en la instancia de escritura o lector de Aurora en función del promedio de sesiones activas (AAS). El valor de AAS se calcula mediante el muestreo y el número de sesiones activas que esperan que una CPU recoja la carga de trabajo de consultas y la procese. Performance Insights brinda una interfaz gráfica en la que puede comprobar las instrucciones SQL que provocan la mayor carga mediante esperas a las sesiones activas.



En la captura de pantalla anterior, la llamada al procedimiento almacenado `my_sqrt` está provocando que una media de 13,03 sesiones deban esperar a que se procesen sus cargas. El siguiente paso lógico es ajustar este procedimiento. Debe identificar las instrucciones SQL en sus lectores y escritores que causan una carga en las instancias respectivas y ajustarlas para mejorar el rendimiento hasta que los valores de AAS caigan y se mantengan por debajo del límite flexible de máxima de vCPU en Performance Insights. Si sus esfuerzos de ajuste han llegado al límite y sigue viendo que el AAS supera el límite máximo de vCPU, puede optar por una clase de instancias más grande para gestionar su carga de trabajo. No opte por una instancia más grande sin antes intentar ajustar la carga de trabajo de las consultas, ya que el aumento del tráfico resaltará las deficiencias que generan las consultas erróneas en la carga de trabajo.

Utilizar registros de consultas lentos y publicarlos en CloudWatch

El registro de consultas lento es una característica nativa de MySQL y complementa a Performance Insights. La práctica recomendada consiste en utilizar estos dos métodos para anticiparse a las consultas problemáticas que pueden causar problemas en las instancias. El registro de consultas lentas registra cualquier consulta que sea más lenta que la variable dinámica `long_query_time`. Esta variable se puede configurar sin necesidad de reiniciar las instancias del clúster.

Para aportar flexibilidad y aislamiento en el ejercicio de ajuste, utilice grupos de parámetros separados para las instancias de escritor y lector. Esto es importante en especial si se utiliza también la división de lectura-escritura. Establezca un límite cómodo para `long_query_time` en

las instancias de su clúster en función de sus necesidades. A medida que vaya ajustando su carga, puede optar por valores rígidos en la variable `long_query_time`, porque puede establecer el umbral en el nivel de milisegundos. Con una alta simultaneidad y una carga de trabajo bien ajustada, casi todas las consultas deberían ejecutarse en milisegundos.

Al configurar Amazon Aurora MySQL-Compatible Edition para que registre las consultas lentas en un archivo, Aurora MySQL-Compatible escribirá los registros de consultas lentas en el sistema de MySQL-Compatible archivos Aurora y los conservará durante 24 horas. Para conservar los registros de consultas lentas durante más tiempo para su análisis, publícalos en Amazon CloudWatch. También puedes crear un CloudWatch panel de control para monitorear tus consultas lentas. Para obtener más información, consulte la entrada del blog Creating an Amazon CloudWatch Dashboard to Monitoring Amazon RDS y Amazon Aurora MySQL. Además de analizar tus consultas lentas en Amazon CloudWatch, puedes perfilar los registros de consultas lentas con pt-query-digest, una herramienta de Percona Toolkit.

También puede optar por automatizar este proceso de descarga y creación de perfiles de las consultas para aumentar la eficiencia de su equipo. Su equipo debe comprobar si las consultas se ejecutan con frecuencia y durante intervalos más largos, y priorizar su ajuste. Intente lograr un estado en el que se registren muy pocas consultas en su registro de consultas lento y reducir el `long_query_time` para ser más agresivo a medida que comprende y ajusta su carga de trabajo.

Guía para ajustar las consultas

Después de identificar las consultas problemáticas en su carga de trabajo, debe ajustar cada consulta. Use las siguientes directrices de ajuste para ayudar a que su carga de trabajo se ejecute de manera más eficiente.

Minimice el número de filas escaneadas

Por básico que parezca, este es un gran consejo a considerar cuando se ajustan las consultas. Use la instrucción EXPLAIN y revise la columna de filas para ver cuántas filas escanea el optimizador en cada unión. Intente reducir el número de filas escaneadas al crear un índice óptimo y, a continuación, vuelva a explicar la consulta para confirmar su trabajo. Para obtener más información, consulte la documentación de MySQL.

Si utiliza tablas particionadas, consúltelas siempre con la cláusula WHERE, que permite reducir las particiones para que el optimizador no tenga que escanear cada partición. Si la cláusula WHERE

contiene una constante para la columna particionada, el optimizador sabrá qué partición buscar, lo que hace que la consulta sea más eficaz.

Otra faceta de este consejo es el diseño de la base de datos. Cuantas menos tablas haya en la consulta, más rápida será la consulta. Si puede desnormalizar el diseño de la base de datos, logrará que el optimizador escanee menos filas, lo que se traduce en un rendimiento de consulta más rápido.

Minimice el uso de tablas temporales y de tablas temporales en el disco

El MySQL-Compatible optimizador Aurora crea tablas temporales tanto en la RAM como en el disco si no puede obtener los resultados deseados de la consulta directamente de los índices. Por lo tanto, una gran parte del ajuste consiste en disponer de los índices correctos que se adapten a su carga de trabajo. Sin embargo, es posible que haya consultas en la carga de trabajo que no puedan basarse únicamente en los índices, por lo que algunas operaciones pueden realizarse en un archivo temporal. Esto está bien siempre y cuando los mantenga al mínimo y se asegure de que se creen muy pocas tablas en el disco. MySQL crea tablas en el disco cuando el tamaño de la tabla temporal es demasiado grande para guardarla en la memoria. La lógica que utiliza MySQL para comprobar el tamaño de la tabla temporal interna es el menor de los dos valores variables tmp_table_size y max-heap-table-size. Puede ajustar estas variables a un valor óptimo en función de su carga de trabajo, de modo que, en los casos en que no pueda evitar las tablas temporales, las coloque en el disco solo en pocas ocasiones.

Puede evitar también los tipos de archivos

Si su carga de trabajo tiene muchas consultas `ORDER BY`, la mejor forma de resolverlas es utilizar los índices correctos en las tablas. Asegúrese de que los índices de varias columnas estén bien diseñados para evitar que se clasifiquen en archivos. No se puede ordenar una columna si las columnas anteriores no están escaneadas con constantes (`in`, >, <, `!=` y `BETWEEN` no permitirán ordenar en la siguiente columna a la derecha). La forma óptima de ordenar en MySQL es colocar un índice de varias columnas que ubique las columnas que contienen valores constantes en la consulta a la izquierda de la columna de clasificación en una estructura contigua. Si, como último recurso, la consulta no puede devolver resultados sin ordenar los archivos, mueva la clasificación a la aplicación.

Evite ejecutar consultas de agregación con un alto nivel de simultaneidad

Es posible que su carga de trabajo tenga un número reducido de consultas de agregación para atender algunas funciones de la aplicación. Este caso de uso requiere mucha cautela. El motor

InnoDB está diseñado para soportar cargas de procesamiento de transacciones en línea (OLTP) adecuadas, pero incluso unas cuantas consultas agrupadas por grupos con una alta simultaneidad pueden suponer una gran carga para la CPU y perjudicar rápidamente el rendimiento del clúster. Para resolver los casos de uso en los que se requieren conjuntos de resultados agregados, agregue previamente los datos en tablas listas para su lectura, para evitar por completo las consultas agrupadas por grupos.

Pruebe la simultaneidad de las consultas

Al ajustar consultas individuales, recuerde que estas consultas se ejecutan simultáneamente en varias vCPU de Aurora. MySQL-Compatible La consulta puede ejecutarse en unos pocos milisegundos en el entorno de prueba en ejecuciones únicas. Pero este no es el panorama completo. Asegúrese de probar la consulta con el nivel de simultaneidad esperado en su clúster de producción y compare su rendimiento. Lance la consulta a producción solo cuando cumpla sus objetivos de simultaneidad. Asegúrese de utilizar el optimizador `hint sql_no_cache` en sus scripts de prueba para evitar recuperar los resultados de la caché. Puede utilizar herramientas como mysqlslap para realizar la prueba de forma simultánea y comparar los resultados.

Ajuste de las cargas de trabajo de escritura

La implementación del equilibrio de carga y la liberación de la instancia de escritura ayudarán a que la carga de trabajo de escritura tenga un mejor rendimiento durante los picos de uso. Para lograr un mejor rendimiento de escritura con un alto nivel de simultaneidad, siga estos pasos adicionales.

Mueva la integridad referencial a la capa de aplicación

Si bien las comprobaciones de integridad referencial son importantes, con el hiperescalado podrían ser perjudiciales para su carga. Para cada escritura, se deben realizar escaneos adicionales antes de ejecutar la propia escritura, lo que se traduce en un rendimiento deficiente. Si su aplicación requiere controles de integridad estrictos, inclúyalos en la capa de aplicación para evitar que se limiten las escrituras.

Evite el uso de claves principales pesadas

Mantenga sus claves principales ligeras. El motor de almacenamiento InnoDB anexa la clave principal a todos los demás índices que cree en la tabla. Cuando la clave principal es grande, afecta al tamaño del índice. El almacenamiento y la recuperación de las páginas de datos se ralentizarán si la clave principal es muy grande. Un ejemplo común es el uso de identificadores únicos universales

como claves principales. Este no es un buen enfoque si el objetivo es el rendimiento en un entorno hiperescalado.

Use el intercambio de particiones para cargar datos en tablas particionadas

Si va a escribir grandes conjuntos de datos en tablas particionadas, la combinación de CARGAR DATOS DE S3 e intercambio de particiones puede mejorar el rendimiento, ya que las inserciones no acceden a la tabla principal. El intercambio de particiones utiliza un lenguaje de definición de datos (DDL) y bloquea los metadatos de la tabla. Asegúrese de que esto se haga cuando haya pocas consultas o ninguna en ejecución en la tabla, de modo que el DDL del intercambio de particiones pueda obtener el bloqueo de metadatos sin esperas. El intercambio en sí solo tarda unos milisegundos en completarse.

Eliminar los índices que no se utilizan

InnoDB optimiza sus planes de consultas en función del crecimiento de sus datos, y es una buena idea comprobar si hay índices no utilizados en su base de datos y eliminarlos. Los índices no utilizados consumen E/S cuando la aplicación escribe datos en una tabla. Consulte la lista de índices no utilizados y compruebe que no se trata de índices que se utilicen en pocas ocasiones, como en los informes trimestrales. Además, considere que algunos índices se utilizan para garantizar la unicidad o el orden, y también deben tenerse en cuenta.

Asegúrese de que las versiones antiguas de las filas se depuren de manera eficiente

En la implementación InnoDB del control de concurrencia multiversión (MVCC), cuando se modifica un registro, la versión actual (antigua) de los datos que se están modificando se registra primero como deshacer un registro en un registro de anulación de registro. Un valor creciente de longitud de lista de historial (HLL) indica que los subprocesos de recopilación de elementos no utilizados de InnoDB (subprocesos de depuración) no están a la altura de la carga de trabajo de escritura o que la depuración está bloqueada por una consulta o transacción de larga duración. Cuando la recopilación de elementos no utilizados se bloquea o retrasa, la base de datos puede sufrir un retraso de depuración considerable que puede afectar negativamente al rendimiento de las consultas. Puede utilizar los siguientes consejos para optimizar el proceso de depuración.

- Mantenga las transacciones pequeñas.
- Para las consultas de lectura, utilice el nivel de aislamiento READ COMMITTED.

- Aumente el número de subprocesos de depuración (innodb_purge_threads e innodb_purge_batch_size). Tenga en cuenta que el ajuste de estos parámetros requiere un reinicio.
- Monitoree el HLL con frecuencia y resuelva cualquier problema de carga de trabajo que impida que la recopilación de elementos no utilizados avance.

Asegúrese de que el registro no cause más problemas

El registro de consultas general registra las conexiones y desconexiones de los clientes, así como todas las instrucciones recibidas por el servidor en el orden en que se recibieron. Cuando está activado, el registro es sincrónico, lo que puede suponer una penalización sustancial del rendimiento en un sistema ocupado. A menos que sea necesario, se recomienda desactivar el registro general.

El registro de consultas lentas registra las instrucciones que tardaron más que el número de segundos de ejecución long_query_time, con la configuración predeterminada de 10 segundos. Cuando la configuración se establece en 0, todas las instrucciones se registran de forma sincrónica, lo que puede provocar una penalización del rendimiento en las bases de datos ocupadas.

Disminuya la carga

Para mejorar los tiempos de respuesta y aumentar los recursos disponibles para los flujos de trabajo críticos, es posible que deba eliminar la carga superflua. Muchas de las soluciones que se tratan en esta sección son decisiones de compensación. Tienen consecuencias para la aplicación y deben considerarse con detenimiento. Puede considerar también estos métodos de estas soluciones en los siguientes casos:

- Ya se encuentra en las instancias de mayor tamaño, en especial en la instancia principal de base de datos en la que se puede escribir.
- Como último recurso, puede ofrecer suficiente margen de maniobra a corto plazo para implementar otros cambios.

Los cambios inmediatos incluyen los siguientes:

- Aleje el tráfico de lectura no crítico de la instancia de base de datos principal.
- Archive o depure los datos antiguos.
- Mueva la integridad referencial.

- Deshabilite el registro binario (si está en uso).
- Aplace los procesos de extracción, transformación y carga (ETL) no críticos.
- Suspenda o disminuya las características no esenciales de la aplicación.

Antes de emprender estas acciones, evalúelas en el contexto de los objetivos y riesgos empresariales a largo plazo.

Cargas de trabajo de lectura y escritura independientes

Una técnica común cuando se ejecutan aplicaciones con tecnología MySQL es descargar las operaciones de lectura de la instancia de base de datos del escritor (principal) y pasarlas a una o más réplicas de bases de datos de solo lectura. Al descargar las lecturas, puede reducir la carga general de la instancia de base de datos principal y dejar espacio para las escrituras. Asegúrese de centrarse solo en las lecturas que no dependan de la constancia inmediata de lectura después de escritura de las réplicas. Un enfoque más eficiente es mover todo el tráfico de lectura a la réplica y planificar reintentar la lectura después de la escritura en caso de que se retrase la replicación. Existen cargas de trabajo de lectura independientes que se pueden reducir, como los servicios de generación de informes. Otras lecturas requerirán cambios en el nivel de la aplicación, donde se conoce bien el contexto por el que se emitió la lectura.

Un enfoque alternativo es implementar una solución proxy de base de datos como intermediaria entre la aplicación y la base de datos, que pueda aportar la función de dividir lectura-escritura y enrutar consultas, sin que la aplicación lo sepa. ProxySQL, MariaDB MaxScale, ScaleARC y Heimdall Data son algunas de las soluciones de proxy compatibles con MySQL disponibles. Estos productos ofrecen varias características adicionales, como el almacenamiento en caché o la multiplexación de conexiones. Cuando experimente un aumento repentino del tráfico e implemente una nueva tecnología en su conjunto de aplicaciones, le recomendamos que comience con la funcionalidad básica de dividir las read/write consultas y probar el comportamiento y el rendimiento antes de utilizar funciones adicionales que podrían tener efectos secundarios no deseados.

Archive o depure los datos antiguos

Otra técnica para mejorar el rendimiento de la base de datos es descargar los datos históricos en otra tabla, base de datos o Amazon Simple Storage Service (Amazon S3). Muchas bases de datos conservan todos los datos en línea durante todo el historial de la aplicación. En circunstancias normales, en una aplicación típica orientada al usuario, esto permite a los usuarios ver todos sus pedidos históricos. Cuando la demanda aumenta repentinamente, es probable que muchos usuarios

activen la aplicación sean nuevos o estén centrados en realizar un nuevo pedido. Si los datos históricos se encuentran en línea, en una sola tabla que contiene miles de millones de filas, esto se agrava. Es probable que la tabla también tenga índices grandes. Tanto los datos de la tabla como los índices se almacenan en una estructura de árbol. Tener más entradas en una tabla requiere más niveles en el árbol, lo que requiere más I/O operaciones para acceder a las filas. Esto aumenta el tiempo de acceso para buscar registros individuales. Y lo que es más importante, hace que grandes partes innecesarias del índice residan en la caché de páginas de la base de datos (grupo de búferes de InnoDB).

Archivar los datos antiguos en una tabla independiente, en una base de datos independiente o en Amazon S3 puede reducir los tiempos de acceso de los usuarios finales, liberar memoria caché y mejorar el rendimiento general de las aplicaciones. Archivar los datos antiguos antes del evento (por ejemplo, conservarlos solo durante 30 o 90 días) limita el tamaño de las tablas e índices de las tablas críticas. Este cambio requiere que se modifique la aplicación para consultar los datos más antiguos desde una ubicación secundaria solo para el pequeño subconjunto de usuarios que solicitan explícitamente ver datos históricos.

Aplace los procesos de ETL que no sean críticos

Las extracciones del sistema de base de datos principal para los procesos de ETL pueden suponer un riesgo de estabilidad en el caso de cargas de trabajo simultáneas y con un alto nivel de transacciones en condiciones de hiperescala. Los procesos de ETL son conocidos por las siguientes características:

- Long-running transacciones
- Bloqueos amplios
- CPU, memoria y I/O consumo
- Contención de las cargas de trabajo transaccionales que dan servicio a las rutas críticas de los usuarios finales.

Procesos de ETL que son variables o impredecibles (por ejemplo, consultas como `INSERT INTO ... SELECT FROM ...;`) aumentan la variabilidad general de la carga y la contención, lo que aumenta el riesgo de estabilidad.

Siempre que sea posible, aplace o reduzca la frecuencia con la que se ejecutan los procesos de ETL, en especial si no brindan funciones críticas. En el caso de los procesos ETL críticos, modifíquelos para que funcionen en unidades de trabajo limitadas, como procesar lotes de números

fijos de filas (por ejemplo, mediante cláusulas `LIMIT`), usar transacciones independientes o extraer cantidades más pequeñas de datos durante un periodo prolongado, para reducir los picos de demanda de recursos de la instancia de base de datos.

Desactive las características menos críticas de las aplicaciones

Reducir la experiencia del usuario o eliminar características no esenciales al manejar un aumento de solicitudes permite conservar los recursos de la base de datos para las características críticas. Esto puede requerir un leve cambio en la experiencia del cliente, pero un flujo diferente es mejor que el hecho de que el sitio esté inactivo. Quizás pueda deshabilitar de forma temporal la personalización de la página principal del sitio, que siempre hace una llamada a la base de datos. O bien, puede dejar de presentar ofertas personalizadas a los clientes habituales mientras selecciona los productos. La desactivación temporal de las características puede permitir que la página se almacene en caché o se entregue sin necesidad de acceder a la base de datos.

Otros ejemplos incluyen una interfaz con un mecanismo de sondeo que comprueba los cambios en los datos que requieren una llamada a la base de datos. La reducción de la frecuencia de sondeo se traducirá inmediatamente en menos llamadas a la base de datos. Las interfaces de usuario que requieren paginación o que ofrecen a los usuarios la opción de recuperar conjuntos de resultados ordenados más alejados del resultado principal requieren llamadas a las bases de datos cada vez más costosas. Limite el número de páginas de un conjunto de resultados para proteger la base de datos de las llamadas más costosas a la base de datos. Utilice indicadores de características en la capa de aplicación para permitir que el equipo de operaciones desactive o reduzca las características a medida que aumenta la carga de la aplicación o la base de datos.

Ajuste de parámetros

Además de los parámetros relacionados con la conexión, hay varios parámetros que puede ajustar para mejorar el rendimiento de su clúster Amazon Aurora MySQL Edition compatible con MySQL en caso de hipercrecimiento. Al cambiar cualquier parámetro de la base de datos, es importante seguir las siguientes prácticas recomendadas:

- Cambie un parámetro a la vez para poder medir el impacto del cambio.
- Es posible que algunos parámetros requieran un periodo de preparación para que surtan efecto después de cambiarlos. Tenga esto en cuenta al observar y medir el rendimiento de su clúster de Aurora MySQL-Compatible.
- Evite valores de parámetros incorrectos, ya que podrían impedir que la instancia de base de datos se inicie.

Entre estos parámetros se incluyen los siguientes:

- `sync_binlog`
- `table_open_cache`
- `innodb_sync_array_size`
- `innodb_flush_log_at_trx_commit`
- `autocommit`
- `tmp_table_size`
- `max_heap_table_size`

Para obtener instrucciones sobre la configuración de estos parámetros, consulte los parámetros de configuración de Aurora MySQL, las recomendaciones para las funciones de MySQL en Aurora MySQL y el comportamiento de las tablas temporales en Aurora MySQL en la AWS documentación.

Recursos

- Serie sobre el recorrido hacia la adopción de una arquitectura nativa en la nube: parte 1: preparar sus aplicaciones para el hipercrecimiento (publicación de blog)
- Serie sobre el recorrido hacia la adopción de una arquitectura nativa en la nube: parte 2: cómo maximizar el rendimiento del sistema (publicación de blog)
- Uso de Amazon RDS Proxy
- Estrategias de almacenamiento en caché
- Activación y desactivación de Performance Insights
- Motor de almacenamiento de InnoDB
- Réplicas de Aurora
- MariaDB Connector/J
- MariaDB MaxScale
- ProxySQL
- Heimdall Data

Historial de documentos

En la siguiente tabla, se describen cambios significativos de esta guía. Si quiere recibir notificaciones de futuras actualizaciones, puede suscribirse a las notificaciones RSS.

Cambio	Descripción	Fecha
Publicación inicial	—	5 de octubre de 2022

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.