Hiperescalando o MySQL-Compatible Aurora para lidar com o crescimento repentino do tráfego

AWS Recomendações

AWS Recomendações: Hiperescalando o MySQL-Compatible Aurora para lidar com o crescimento repentino do tráfego

Table of Contents

Hiperescalar o Aurora Edição Compatível com MySQL para lidar com o crescimento repentino do tráfego

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

Outubro de 2022

Seus negócios na Internet podem enfrentar uma situação de hipercrescimento sem precedentes com a qual sua infraestrutura de aplicações existente não é capaz de lidar. Isso pode acontecer em resposta a vários fatores, como um anúncio sobre seu produto que resulta em um interesse maior do que o esperado ou uma mudança repentina nos padrões de compra do cliente, de presencial para online.

Para lidar com essas cargas inesperadas, é necessário realizar o hiperajuste de escala da sua infraestrutura. A escalabilidade no nível de aplicação envolve a adição de mais servidores ou pods. Mas a parte mais difícil para atingir o hiperajuste de escala é o banco de dados. Você pode escalar sua instância de banco de dados para o maior tamanho de instância disponível, mas isso pode não resolver seu problema.

Se você executa sua workload de banco de dados no Amazon Aurora Edição Compatível com MySQL, este guia fornece recomendações podem ser implementadas para realizar o hiperajuste de escala em seu banco de dados para atender ao crescimento repentino. Nem todas essas recomendações são práticas recomendadas de longo prazo. Se você espera um hipercrescimento e tem tempo para planejar como lidar com a situação, consulte as publicações no blog listadas na seção Recursos. Essas postagens podem ajudar você a implementar práticas recomendadas de longo prazo. Por outro lado, se você enfrentar um hipercrescimento inesperado, esse guia ajudará a gerenciar sua carga e alcançar uma estabilidade razoável enquanto planeja e implementa soluções de longo prazo para sua empresa.

Observe que as recomendações descritas neste guia exigirão a ajuda de sua equipe de desenvolvimento na implementação.

Resultados de negócios desejados

As abordagens abordadas neste guia ajudarão você a fazer o seguinte:

- Estabilize seus negócios em caso de hipercrescimento inesperado. Obtenha estabilidade suficiente para implementar as práticas recomendadas de longo prazo para o hipercrescimento.

- Evite perdas financeiras. Interrupções repentinas em um ambiente com hiperajuste de escala podem levar a uma queda nas transações comerciais realizadas em sua aplicação por seus clientes. Isso pode levar a perdas financeiras substanciais em alguns casos. Um ambiente estabilizado com hiperajuste de escala é fundamental para evitar interrupções prolongadas que resultam na perda de negócios.

Como gerenciar conexões

À medida que a demanda por sua aplicação cresce, o tráfego de front-end aumenta. Em um cenário típico, você configura o escalonamento automático na camada de aplicação para lidar com essa explosão de tráfego de entrada. Como resultado, a camada de aplicação começa a ser escalada automaticamente, e mais servidores de aplicações (instâncias) são adicionados para atender ao aumento do tráfego. Como todos os servidores de aplicações têm configurações pré-definidas do pool de conexões de banco de dados, o número de conexões de entrada com o banco de dados cresce proporcionalmente às instâncias recém-implantadas.

Por exemplo, 20 servidores de aplicações configurados com 200 conexões de banco de dados cada abririam um total de 4.000 conexões de banco de dados. Se o pool de aplicações for escalado para até 200 instâncias (por exemplo, durante os horários de pico), a contagem total de conexões chegará a 40.000. Em uma workload típica, a maioria dessas conexões provavelmente está ociosa. Mas o aumento nas conexões pode limitar a capacidade de escalabilidade do seu banco de dados do Amazon Aurora Edição Compatível com MySQL. Isso ocorre porque até mesmo conexões ociosas consomem memória e outros recursos do servidor, como descritores de arquivos. O Aurora Edição Compatível com MySQL normalmente usa menos memória do que o MySQL Community Edition para manter o mesmo número de conexões. No entanto, o uso de memória para conexões ociosas ainda não é zero.

Variáveis de configuração

É possível controlar o número de conexões de entrada permitidas em seu banco de dados com duas variáveis principais de configuração do servidor: `max_connections` e `max_connect_errors`.

Variável de configuração max_connections

A variável de configuração `max_connections` limita o número de conexões de banco de dados para cada instância do MySQL. A prática recomendada é configurá-la um pouco acima do número máximo de conexões que você espera abrir em cada instância de banco de dados.

Se você também habilitou o `performance_schema`, tenha muito cuidado com a configuração `max_connections`. As estruturas de memória do Performance Schema são dimensionadas automaticamente com base nas variáveis de configuração do servidor, incluindo `max_connections`. Quanto maior a configuração da variável, mais memória o Performance Schema usa. Em casos extremos, isso pode causar out-of-memory problemas em tipos de instâncias

menores. Observe que ativar o Performance Insights habilitará automaticamente o Performance Schema.

Variável de configuração max_connect_errors

A variável de configuração `max_connect_errors` determina quantas solicitações sucessivas de conexão interrompida são permitidas de um determinado host cliente. Se o host cliente exceder o número especificado de tentativas sucessivas de conexão malsucedidas, o servidor o bloqueará. Tentativas adicionais de conexão desse cliente gerarão um erro.

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

Se você tiver erros de “o host está bloqueado”, evite aumentar o valor da `max_connect_errors` variável. Em vez disso, investigue os contadores de diagnóstico do servidor na variável de status `aborted_connects` e a tabela `host_cache`. Use as informações coletadas para identificar e corrigir clientes com problemas de conexão. Além disso, observe que esse parâmetro não terá efeito se `skip_name_resolve` estiver definido como 1 (padrão).

Consulte o Manual de referência do MySQL para obter detalhes sobre:

- Variável Max_connect_errors
- Erro “O host está bloqueado”
- Variável de status Aborted_connects
- Tabela Host_cache

Implementar um pool de conexões

Um evento de escalabilidade pode adicionar mais servidores de aplicações, o que, por sua vez, pode fazer com que o servidor de banco de dados exceda o número de conexões ativas totalmente carregadas. A adição de um pool de conexões ou camada de proxy entre os servidores de aplicações e o banco de dados age como um funil, reduzindo o número total de conexões no banco de dados. O objetivo principal de um proxy é a reutilização de conexões de banco de dados por meio de multiplexação.

De um lado, o proxy se conecta ao banco de dados com um número controlado de conexões. Por outro, o proxy aceita conexões de aplicações. Ele também fornece recursos adicionais, como cache

de consultas, buffer de conexão, reescrita e roteamento de consultas e balanceamento de carga. A camada do pool de conexões precisa ser configurada para manter o número máximo de conexões com o banco de dados abaixo do número totalmente carregado. O Amazon RDS Proxy é um proxy totalmente gerenciado que pode ser implementado para essa finalidade. O Amazon RDS Proxy não exige alterações de código para a maioria das aplicações, e você não precisa gerenciar nenhuma infraestrutura extra para implementar a solução.

Também é possível explorar os seguintes proxies de terceiros que podem ser usados com o Aurora Edição Compatível com MySQL:

- ProxySQL
- MariaDB MaxScale
- Scale Arc
- Heimdall Data

Evite tempestades de conexão

Considere como seu pool de conexões se comporta no caso de um banco de dados sobrecarregado ou uma réplica ficar muito para trás em relação ao nó primário. Ao configurar seu servidor proxy ou pools de conexão, certifique-se de não redefinir todo o pool de conexões com base em respostas lentas do banco de dados (causadas por problemas subjacentes de hardware ou armazenamento ou restrições de recursos de banco de dados).

O início repentino de centenas de conexões gera uma tempestade de conexões porque um grande número de solicitações de novas conexões com o banco de dados são todas iniciadas ao mesmo tempo. A tempestade consome muitos recursos. Criar uma nova conexão de banco de dados no MySQL é uma operação cara porque o back-end troca vários pacotes de rede para o handshake inicial, gera um novo processo, aloca memória, manipula a autenticação e assim por diante. Se um grande número de solicitações for recebido em um curto período de tempo, o banco de dados poderá aparentar não estar respondendo.

O MySQL tem um mecanismo de proteção contra esse aumento nas solicitações de conexão. A variável `back_log` pode ser definida como o número de solicitações que podem ser empilhadas durante um curto período de tempo antes que o MySQL pare momentaneamente de responder a novas solicitações. O valor é imposto por um thread de manipulações de conexão que, por si só, pode ser sobrecarregado por uma tempestade de conexões. Para obter mais informações, consulte o Manual de referência do MySQL.

Se sua conexão estiver configurada para ser redefinida quando o banco de dados estiver lento, você iniciará o ciclo repetidamente. Da mesma forma, se você antecipar um aumento repentino no tráfego do banco de dados em determinados momentos do dia (por exemplo, quando o mercado de ações abre), pré-aqueça seu pool de conexões para que você não tente abrir muitas conexões ao mesmo tempo em que uma carga de tráfego elevada está começando.

Ajustar sua workload de consulta

Uma workload bem ajustada ajudará você a obter uma solução estável para o hipercrescimento. Se sua workload não estiver bem ajustada, não importa a potência do cluster Amazon Aurora utilizado, haverá gargalos que degradarão a performance e afetarão a experiência do usuário da sua aplicação. A prática recomendada é ter um processo implementado desde o início que ajude a identificar e ajustar consultas problemáticas em seu sistema.

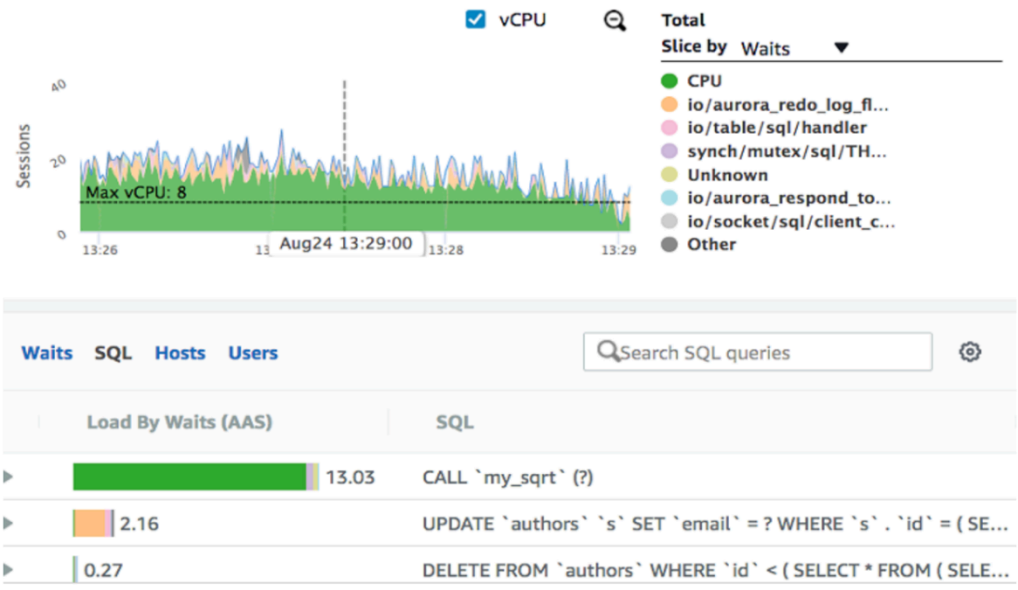
Rastrear e ajustar consultas problemáticas em sua workload

Ao se deparar com um hipercrescimento, ter uma workload bem ajustada já é meio caminho percorrido. Para entender a natureza das workloads em tempo real e os problemas de performance que seu cluster Aurora está enfrentando, garanta que sua equipe reúna consultas problemáticas das instâncias de gravação e leitura. Essas consultas problemáticas precisam ser ajustadas para que sua workload seja executada em um estado ideal. O Amazon Aurora MySQL-Compatible Edition oferece duas maneiras de fazer isso:

- Insights de Performance
- Log de consultas lentas

Usar o Performance Insights

O Performance Insights rastreia a carga na instância de gravação ou leitura do Aurora com base na média de sessões ativas (AAS). O valor do AAS é calculado por meio de amostragem e o número de sessões ativas que estão aguardando que uma CPU pegue sua workload de consulta e a processe. O Performance Insights fornece uma interface gráfica na qual você pode verificar as instruções SQL que estão causando a maior carga por conta de esperas por sessões ativas.



Na captura de tela anterior, a chamada para o procedimento armazenado `my_sqrt` está fazendo com que uma média de 13,03 sessões aguardem o processamento de suas cargas. A próxima etapa lógica é ajustar esse procedimento. É necessário identificar instruções SQL em seus leitores e gravadores que estão causando carga em suas respectivas instâncias e ajustá-las para melhorar a performance até que os valores de AAS caiam e permaneçam abaixo da linha pontilhada máxima de vCPU no Performance Insights. Se você atingiu um limite máximo com seus esforços de ajuste e continua observando o AAS acima da linha Max vCPU, é possível selecionar uma classe de instância maior para lidar com sua workload. Não opte por uma instância maior sem antes tentar ajustar sua workload de consultas, pois o aumento do tráfego começará a expor as falhas criadas por consultas incorretas na workload.

Usando registros de consulta lentos e publicando-os no CloudWatch

O log de consultas lentas é um recurso nativo do MySQL que complementa o Performance Insights. A prática recomendada é usar esses dois métodos para evitar consultas problemáticas que podem causar estragos em suas instâncias. O log de consultas lentas registra qualquer consulta mais lenta do que a variável dinâmica `long_query_time`. Essa variável pode ser configurada sem que seja necessário reiniciar suas instâncias de cluster.

Para fornecer flexibilidade e isolamento no exercício de ajuste, use grupos de parâmetros separados para suas instâncias de gravação e de leitura. Isso é especialmente importante quando a divisão de leitura-gravação é usada. Estabeleça um limite confortável para `long_query_time` em suas

instâncias de cluster com base em suas necessidades. Ao ajustar sua carga, você pode buscar valores agressivos na variável `long_query_time` porque você pode definir o limite no nível de milissegundos. Com alta simultaneidade e uma workload bem ajustada, quase todas as suas consultas devem ser executadas em milissegundos.

Quando você configura o Amazon Aurora MySQL-Compatible Edition para registrar consultas lentas em um arquivo, o Aurora MySQL-Compatible grava os registros de consulta lentos no sistema de MySQL-Compatible arquivos do Aurora e os retém por 24 horas. Para reter os registros de consulta lentos por um período mais longo para análise, publique-os na Amazon CloudWatch. Você também pode criar um CloudWatch painel para monitorar suas consultas lentas. Para obter mais informações, consulte a postagem do blog Criação de um CloudWatch painel da Amazon para monitorar o Amazon RDS e o Amazon Aurora MySQL. Além de analisar suas consultas lentas na Amazon CloudWatch, você pode criar um perfil de registros de consultas lentas usando pt-query-digest, uma ferramenta do Percona Toolkit.

Você também pode optar por automatizar esse processo de download e criação de perfil de consultas para maior eficiência em sua equipe. Sua equipe deve verificar as consultas que são executadas com frequência e por intervalos mais longos e priorizar seu ajuste. Procure um estado em que poucas consultas sejam registradas no log de consultas lentas e você possa reduzir o valor de `long_query_time` para se tornar mais agressivo à medida que você entende e ajusta a workload.

Orientações para ajuste de consultas

Depois que você identificar consultas problemáticas em sua workload, cada consulta deverá ser ajustada. Use as orientações a seguir sobre ajuste para ajudar a executar a workload com mais eficiência.

Minimize o número de linhas verificadas

Por mais básico que pareça, esse é um ótimo conselho para usar ao ajustar consultas. Use a instrução EXPLAIN e revise a coluna de linhas para ver quantas linhas o otimizador verifica em cada união. Tente reduzir o número de linhas verificadas criando um índice ideal e, em seguida, explique novamente sua consulta para confirmar seu trabalho. Para obter mais informações, consulte a documentação do MySQL.

Se você usa tabelas particionadas, consulte-as sempre com a cláusula WHERE, a qual permite a remoção de partições para que o otimizador não precise verificar cada partição. Se a cláusula

WHERE contiver uma constante para a coluna particionada, o otimizador saberá qual partição procurar e isso tornará sua consulta mais eficiente.

Outra faceta dessa recomendação é o design do seu banco de dados. Quanto menos tabelas em sua consulta, mais rápida ela será. Se você puder desnormalizar o design do banco de dados, poderá fazer com que o otimizador verifique menos linhas, o que resultará em uma melhor performance da consulta.

Minimizar uso de tabelas temporárias e tabelas temporárias em disco

O MySQL-Compatible otimizador Aurora cria tabelas temporárias na RAM e no disco se não conseguir obter os resultados desejados da sua consulta diretamente dos índices. Consequentemente, uma grande parte do ajuste é ter os índices certos que atendam à workload. No entanto, pode haver consultas em sua workload que não podem depender apenas de índices. Portanto, algumas operações podem ser executadas em um arquivo temporário. Isso é bom, desde que você os mantenha no mínimo e garanta que poucas tabelas sejam criadas no disco. O MySQL cria tabelas em disco quando a tabela temporária é muito grande para ser armazenada na memória. A lógica usada pelo MySQL para verificar o tamanho da tabela temporária interna é a menor dos dois valores de variáveis tmp_table_size e max-heap-table-size. É possível ajustar essas variáveis para um valor ideal com base na sua workload para que, nos casos em que não é possível evitar tabelas temporárias, você as envie para o disco somente em raras ocasiões.

Evite classificar arquivos

Se a workload tiver muitas consultas `ORDER BY`, a melhor maneira de resolvê-las é usar os índices corretos em suas tabelas. Certifique-se de que seus índices de várias colunas sejam bem projetados para evitar a classificação em arquivos. A classificação não poderá ocorrer em uma coluna se as colunas anteriores não forem verificadas com constantes (`in`, `>`, `<`, `!=` e `BETWEEN` não permitirão a classificação na próxima coluna à direita). A melhor maneira de classificar no MySQL é colocar um índice de várias colunas que posiciona as colunas que contêm valores constantes fornecidos na consulta à esquerda da coluna de classificação em uma estrutura contígua. Se, em última instância, sua consulta não conseguir retornar resultados sem uma classificação de arquivo, mova a classificação para a aplicação.

Evite executar consultas de agregação em alta simultaneidade

A workload pode ter um pequeno número de consultas de agregação para atender a algumas funcionalidades da aplicação. Esse caso de uso exige muita cautela. O mecanismo InnoDB é voltado

para cargas adequadas de processamento de transações online (OLTP), mas até mesmo algumas consultas agrupadas em alta simultaneidade podem sobrecarregar a CPU e degradar rapidamente a performance do cluster. Para resolver casos de uso em que conjuntos de resultados agregados são necessários, pré-agregue os dados em tabelas prontas para leitura para que você possa evitar completamente consultas de agrupamento.

Testar as consultas quanto à simultaneidade

Ao ajustar consultas individuais, lembre-se de que essas consultas são executadas simultaneamente em várias vCPUs no Aurora. MySQL-Compatible Sua consulta pode ser executada em alguns milissegundos em seu ambiente de teste em execuções únicas. Mas esse não é o quadro completo. Certifique-se de testar a consulta com o nível esperado de simultaneidade no cluster de produção e avaliar sua performance. Libere a consulta para produção somente quando ela atender às suas metas de simultaneidade. Certifique-se de usar o otimizador `hint sql_no_cache` em seus scripts de teste para evitar buscar resultados no cache. É possível usar ferramentas como mysqlslap para realizar o teste em simultâneo e comparar os resultados.

Ajustar workloads de gravação

A implementação do balanceamento de carga e a liberação da instância de gravação ajudarão sua workload de gravação a apresentar uma melhor performance durante os picos mais altos. Para obter uma melhor performance de gravação em alta simultaneidade, siga estas etapas adicionais.

Mover a integridade referencial para a camada de aplicação

Embora as verificações de integridade referencial sejam importantes, com o hiperajuste de escala elas podem ser prejudiciais à sua workload. Para cada gravação, verificações extras devem ser realizadas antes que a gravação em si seja executada, o que resulta em baixa performance. Se a aplicação exigir verificações rigorosas de integridade, integre-as na camada de aplicação para evitar que elas restrinjam suas gravações.

Evite usar chaves primárias pesadas

Mantenha suas chaves primárias leves. O mecanismo de armazenamento InnoDB anexa a chave primária a todos os outros índices criados por você em sua tabela. Quando sua chave primária é grande, o tamanho do índice é afetado. O armazenamento e a recuperação de páginas de dados diminuirão se a chave primária for muito grande. Um exemplo comum é o uso de identificadores

universalmente exclusivos como chaves primárias. Essa não é uma boa abordagem se seu objetivo é performance em um ambiente com hiperajuste de escala.

Usar troca de partições para carregar dados em tabelas particionadas

Se você estiver gravando grandes conjuntos de dados em tabelas particionadas, a combinação entre LOAD DATA FROM S3 e a troca de partições pode melhorar a performance porque a tabela principal não está sendo acessada para as inserções. A troca de partições envolve uma linguagem de definição de dados (DDL) e coloca um bloqueio de metadados em sua tabela. Certifique-se de que isso seja feito quando houver poucas ou nenhuma consulta em execução na tabela para que a DDL de troca de partições possa implementar o bloqueio de metadados sem esperas. A troca em si leva apenas milissegundos para ser concluída.

Remover índices não utilizados

O InnoDB otimiza seus planos de consulta com base no crescimento dos dados, e é uma boa ideia verificar se há índices não utilizados em seu banco de dados e removê-los. Os índices não utilizados consomem E/S quando a aplicação grava dados em uma tabela. Verifique a lista de índices não utilizados e certifique-se de que eles não sejam índices usados em situações raras, como relatórios trimestrais. Além disso, observe que alguns índices são usados para impor exclusividade ou ordenação e também devem ser considerados.

Garanta que as versões antigas da linha sejam limpas com eficiência

Na implementação do InnoDB do controle de simultaneidade multiversão (MVCC), quando um registro é modificado, a versão atual (antiga) dos dados que estão sendo modificados é registrada pela primeira vez como desfazer registro em um log de desfazer. Um valor crescente do comprimento da lista de histórico (HLL) indica que os segmentos de coleta de resíduos do InnoDB (segmentos de limpeza) não estão acompanhando a workload de gravação ou que a limpeza está bloqueada por uma consulta ou transação de longa duração. Quando a coleta de resíduos é bloqueada ou atrasada, o banco de dados pode desenvolver um atraso substancial na limpeza que pode afetar negativamente a performance da consulta. Você pode usar as recomendações a seguir para otimizar o processo de limpeza.

- Mantenha as transações pequenas.
- Para consultas de leitura, use o nível de isolamento READ COMMITTED.
- Aumente o número de threads de limpeza (innodb_purge_threads e innodb_purge_batch_size). Observe que o ajuste desses parâmetros requer uma reinicialização.

- Monitore o HLL regularmente e resolva quaisquer problemas de workload que impeçam o progresso da coleta de resíduos.

Certifique-se de que o registro em log não cause contenção adicional

O log geral de consultas registra as conexões e desconexões do cliente, bem como todas as declarações recebidas pelo servidor na ordem em que foram recebidas. Quando ativado, o registro em log é síncrono, o que pode levar a uma penalidade substancial de performance em um sistema ocupado. A menos que seja necessário, recomendamos desativar o log geral.

O log de consultas lentas registra instruções que demoraram mais do que o número de segundos long_query_time para executar com a configuração padrão de 10 segundos. Quando a configuração é definida como 0, todas as instruções são registradas de forma síncrona, o que pode levar a uma penalidade de performance em bancos de dados com muita atividade.

Distribuir a carga

Melhorar os tempos de resposta e aumentar os recursos disponíveis para fluxos de trabalho críticos pode exigir a eliminação de cargas externas. Muitas das soluções abordadas nesta seção são decisões que envolvem algum comprometimento. Elas têm consequências para a aplicação e devem ser cuidadosamente consideradas. Considere usar essas soluções nas seguintes situações:

- Você já está usando instâncias do tamanho máximo, especialmente para a instância de banco de dados gravável principal.
- Como último recurso para fornecer espaço suficiente a curto prazo para implementar outras mudanças.

Essas alterações incluem:

- Afastar o tráfego de leitura não crítico da instância de banco de dados principal.
- Arquivar ou limpar dados antigos.
- Remover a integridade referencial.
- Desativar o binlog (se estiver em uso).
- Adiar processos não críticos de extração, transformação e carga (ETL).
- Suspender ou reduzir a prioridade de recursos não essenciais da aplicação.

Antes de realizar essas ações, avalie-as no contexto de metas e riscos comerciais de longo prazo.

Separar workloads de leitura e gravação

Uma técnica comum ao executar aplicações baseados em MySQL é transferir as operações de leitura da instância do banco de dados de gravação (principal) para uma ou mais réplicas do banco de dados somente de leitura. Ao descarregar as leituras, é possível reduzir a carga geral na instância do banco de dados principal e abrir espaço para gravações. Certifique-se de direcionar apenas leituras que não dependam de consistência de leitura após gravação imediata para réplicas. Uma abordagem mais eficiente é mover todo o tráfego de leitura para a réplica e planejar a repetição da leitura após gravação em caso de atraso na replicação. Há workloads de leitura independentes que podem ser descarregadas, como serviços de relatórios. Outras leituras exigirão alterações no nível da aplicação, onde o contexto do motivo pelo qual a leitura foi emitida é bem conhecido.

Uma abordagem alternativa é implementar uma solução de proxy de banco de dados como intermediária entre a aplicação e o banco de dados, o que pode fornecer a função de divisão de leitura e gravação e roteamento de consultas, sem o reconhecimento da aplicação. ProxySQL, MariaDB, ScaleArc MaxScale e Heimdall Data são algumas das soluções de proxy compatíveis com MySQL disponíveis. Esses produtos oferecem vários recursos adicionais, como armazenamento em cache ou multiplexação de conexões. Quando você está enfrentando um aumento repentino no tráfego e implementando uma nova tecnologia em sua pilha de aplicativos, recomendamos começar com a funcionalidade básica para dividir read/write consultas e testar o comportamento e o desempenho antes de usar recursos adicionais que possam ter efeitos colaterais indesejados.

Arquivar ou limpar dados mais antigos

Outra técnica para melhorar a performance do banco de dados é transferir dados históricos para outra tabela, banco de dados ou Amazon Simple Storage Service (Amazon S3). Muitos bancos de dados retêm todos os dados em linha durante todo o histórico da aplicação. Em circunstâncias normais, em uma aplicação típica voltada para o usuário, isso fornece aos usuários a capacidade de ver todos os seus pedidos históricos. Quando a demanda aumenta repentinamente, muitos usuários ativos na aplicação provavelmente são novos ou estão focados em fazer um novo pedido. Se os dados históricos residem online em uma única tabela contendo bilhões de linhas, isso aumenta. A tabela provavelmente também possui índices grandes. Tanto os dados quanto os índices da tabela são armazenados em uma estrutura de árvore. Ter mais entradas em uma tabela requer mais níveis na árvore, o que requer mais I/O operações para acessar as linhas. Isso aumenta o tempo de acesso para encontrar registros individuais. Mais importante ainda, faz com que grandes

partes desnecessárias do índice residam no cache da página do banco de dados (pool de buffer do InnoDB).

Arquivar dados antigos em uma tabela separada, em um banco de dados separado ou no Amazon S3 pode reduzir o tempo de acesso dos usuários finais, liberar cache precioso e melhorar a performance geral da aplicação. O arquivamento de dados antigos antes do evento (por exemplo, retendo somente 30 ou 90 dias) limita o tamanho das tabelas e dos índices em tabelas críticas. Essa alteração exige que a aplicação seja modificada para consultar os dados mais antigos de um local secundário somente para o pequeno subconjunto de usuários que solicitam explicitamente a visualização dos dados históricos.

Adiar processos de ETL não críticos

As extrações do sistema de banco de dados principal para processos de ETL podem apresentar risco de estabilidade para workloads altamente transacionais e simultâneas durante condições de hiperajuste de escala. Os processos de ETL são conhecidos pelas seguintes características:

- Long-running transações
- Bloqueios amplos
- CPU, memória e I/O consumo
- Contenção com workloads transacionais que atendem ao caminho crítico do usuário final.

Processos de ETL que são variáveis ou imprevisíveis (por exemplo, consultas como `INSERT INTO ... SELECT FROM ...;`) aumentam a variabilidade geral na carga e na contenção, aumentando o risco de estabilidade.

Sempre que possível, adie ou reduza a frequência com que os processos de ETL são executados, especialmente se eles não estiverem fornecendo funções críticas. Para processos críticos de ETL, modifique-os para que operem em unidades de trabalho limitadas, como o processamento de lotes de números fixos de linhas (por exemplo, usando cláusulas `LIMIT`), usar transações separadas ou extrair quantidades menores de dados por um longo período de tempo para reduzir as demandas de pico de recursos da instância de banco de dados.

Desativar os recursos menos essenciais da aplicação

A degradação suave da experiência do usuário ou a remoção de recursos não essenciais ao lidar com um aumento de solicitações conserva os recursos do banco de dados para funções críticas.

Isso pode exigir uma pequena mudança na experiência do cliente, mas um fluxo diferente é melhor do que o site se tornar inativo. Talvez a personalização na primeira página do site, que sempre faz uma chamada ao banco de dados, possa ser temporariamente desativada. Ou você pode parar de apresentar ofertas personalizadas para clientes recorrentes enquanto seleciona produtos. A desativação temporária dos recursos pode permitir que a página seja armazenada em cache ou entregue sem exigir acesso ao banco de dados.

Outros exemplos incluem um front-end com um mecanismo de pesquisa que verifica as alterações de dados que exigem uma chamada ao banco de dados. Reduzir a frequência da pesquisa resultará imediatamente em menos chamadas ao banco de dados. As interfaces de usuário que exigem paginação ou oferecem aos usuários a opção de recuperar conjuntos de resultados ordenados mais distantes do resultado principal exigem chamadas ao banco de dados cada vez mais caras. Limite o número de páginas em um conjunto de resultados para proteger o banco de dados das chamadas de banco de dados mais caras. Use sinalizadores de recurso na camada de aplicação para permitir que a equipe de operações desative ou reduza os recursos à medida que a carga da aplicação ou do banco de dados aumenta.

ajuste de parâmetros

Além dos parâmetros relacionados à conexão, há alguns parâmetros que você pode ajustar para melhorar o desempenho do seu cluster Amazon Aurora MySQL compatível com o MySQL quando confrontado com o crescimento excessivo. Ao alterar qualquer parâmetro do banco de dados, é importante usar as seguintes práticas recomendadas:

- Mude um parâmetro por vez para que você possa medir o impacto da mudança.
- Alguns parâmetros podem exigir um período de aquecimento para manifestar seu efeito após serem alterados. Considere isso ao observar e medir a performance do seu cluster compatível com o Aurora Edição Compatível com MySQL.
- Evite valores de parâmetros incorretos, o que pode impedir a inicialização da instância do banco de dados.

Esses parâmetros incluem:

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

Para obter diretrizes sobre como configurar esses parâmetros, consulte Parâmetros de configuração do Aurora MySQL, Recomendações para recursos do MySQL no Aurora MySQL e Comportamento da tabela temporária no Aurora MySQL na documentação. AWS

Recursos

- Série Jornada rumo a adoção de uma arquitetura nativa de nuvem: 1 — Preparar as aplicações para o crescimento acelerado (publicação no blog)
- Série Jornada rumo a adoção de uma arquitetura nativa de nuvem: 2 — Maximizar a throughput do sistema (publicação no blog)
- Usar Amazon RDS Proxy
- Estratégias de armazenamento em cache
- Ativar e desativar o Performance Insights
- O mecanismo de armazenamento InnoDB
- Réplicas do Aurora
- MariaDB Connector/J
- MariaDB MaxScale
- ProxySQL
- Heimdall Data

Histórico do documento

A tabela a seguir descreve alterações significativas feitas neste guia. Se desejar receber notificações sobre futuras atualizações, inscreva-se em um feed RSS.

Alteração	Descrição	Data
Publicação inicial	—	5 de outubro de 2022

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.