Hyperscaling MySQL-Compatible Aurora untuk menangani pertumbuhan lalu lintas yang tiba-tiba

AWS Panduan Preskriptif

AWS Panduan Preskriptif: Hyperscaling MySQL-Compatible Aurora untuk menangani pertumbuhan lalu lintas yang tiba-tiba

Table of Contents

Hyperscaling Aurora MySQL kompatibel untuk menangani pertumbuhan lalu lintas mendadak

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

Oktober 2022

Bisnis Anda di internet mungkin menghadapi hypergrowth yang belum pernah terjadi sebelumnya, yang infrastruktur aplikasi Anda saat ini tidak mampu menangani. Ini dapat terjadi sebagai respons terhadap berbagai faktor, seperti iklan tentang produk Anda yang menghasilkan minat lebih dari yang diharapkan atau perubahan tiba-tiba dalam pola pembelian pelanggan dari tatap muka ke online.

Untuk mengatasi beban tak terduga ini, Anda harus melakukan hyperscale infrastruktur Anda. Penskalaan tingkat aplikasi memerlukan penambahan lebih banyak server atau pod. Tetapi bagian yang paling sulit dalam mencapai hyperscaling adalah database. Anda dapat menskalakan instance database Anda ke ukuran instans terbesar yang tersedia, tetapi itu mungkin tidak menyelesaikan masalah Anda.

Jika Anda menjalankan beban kerja database di Amazon Aurora Edisi yang kompatibel dengan MySQL, panduan ini memberikan rekomendasi yang dapat Anda terapkan untuk melakukan hyperscale database Anda untuk memenuhi hiperpertumbuhan mendadak. Tidak semua rekomendasi ini adalah praktik terbaik jangka panjang. Jika Anda mengharapkan hypergrowth dan punya waktu untuk merencanakan untuk mengatasinya, lihat posting blog yang tercantum di Sumber dayabagian tersebut. Posting tersebut dapat membantu Anda menerapkan praktik terbaik jangka panjang. Di sisi lain, jika Anda menghadapi pertumbuhan berlebih yang tidak terduga, panduan ini akan membantu Anda mengelola beban Anda dan mencapai stabilitas yang wajar saat Anda merencanakan dan menerapkan solusi jangka panjang untuk bisnis Anda.

Perhatikan bahwa rekomendasi yang diuraikan dalam panduan ini akan memerlukan bantuan implementasi dari tim pengembangan Anda.

Hasil bisnis yang ditargetkan

Pendekatan yang tercakup dalam panduan ini akan membantu Anda melakukan hal berikut:

- Stabilkan bisnis Anda jika terjadi pertumbuhan berlebih yang tidak terduga. Mencapai stabilitas yang cukup untuk menerapkan praktik terbaik jangka panjang untuk pertumbuhan berlebih.

- Mencegah kerugian finansial. Gangguan mendadak pada lingkungan hyperscaled dapat menyebabkan penurunan transaksi bisnis yang dilakukan pada aplikasi Anda oleh pelanggan Anda. Hal ini dapat menyebabkan kerugian finansial yang besar dalam beberapa kasus. Lingkungan hiperscaled yang stabil adalah kunci untuk mencegah pemadaman lama yang mengakibatkan hilangnya bisnis.

Mengelola koneksi

Seiring permintaan untuk aplikasi Anda tumbuh, lalu lintas front-end meningkat. Dalam skenario tipikal, Anda mengatur penskalaan otomatis di tingkat aplikasi untuk menangani ledakan lalu lintas yang masuk. Akibatnya, tingkat aplikasi mulai skala otomatis, dan lebih banyak server aplikasi (instance) ditambahkan untuk memenuhi peningkatan lalu lintas. Karena semua server aplikasi memiliki pengaturan kumpulan koneksi database yang telah dikonfigurasi sebelumnya, jumlah koneksi masuk ke database bertambah sebanding dengan instance yang baru digunakan.

Misalnya, 20 server aplikasi yang dikonfigurasi dengan 200 koneksi database masing-masing akan membuka total 4.000 koneksi database. Jika kumpulan aplikasi menskalakan hingga 200 instance (misalnya, selama jam sibuk), jumlah total koneksi akan mencapai 40.000. Di bawah beban kerja yang khas, sebagian besar koneksi ini kemungkinan tidak digunakan. Tetapi lonjakan koneksi mungkin membatasi kemampuan database Edisi Amazon Aurora MySQL yang kompatibel dengan skala. Ini karena bahkan koneksi idle mengkonsumsi memori dan sumber daya server lainnya, seperti deskriptor file. Aurora MySQL kompatibel biasanya menggunakan lebih sedikit memori daripada MySQL Community Edition untuk mempertahankan jumlah koneksi yang sama. Namun, penggunaan memori untuk koneksi idle masih belum nol.

Variabel konfigurasi

Anda dapat mengontrol jumlah koneksi masuk yang diizinkan ke database Anda dengan dua variabel konfigurasi server utama: `max_connections` dan`max_connect_errors`.

Variabel konfigurasi max_connections

Variabel konfigurasi `max_connections` membatasi jumlah koneksi database untuk setiap instance MySQL. Praktik terbaik adalah mengaturnya sedikit lebih tinggi dari jumlah maksimum koneksi yang Anda harapkan untuk dibuka pada setiap instance database.

Jika Anda juga mengaktifkan`performance_schema`, berhati-hatilah dengan `max_connections` pengaturannya. Struktur memori Skema Kinerja berukuran secara otomatis berdasarkan variabel konfigurasi server, termasuk`max_connections`. Semakin tinggi Anda mengatur variabel, semakin banyak memori Performance Schema menggunakan. Dalam kasus ekstrim, ini dapat menyebabkan out-of-memory masalah pada jenis instance yang lebih kecil. Perhatikan bahwa mengaktifkan Performance Insights akan secara otomatis mengaktifkan Performance Schema.

Variabel konfigurasi max_connect_errors

Variabel konfigurasi `max_connect_errors` menentukan berapa banyak permintaan koneksi terputus berturut-turut yang diizinkan dari host klien tertentu. Jika host klien melebihi jumlah yang ditentukan dari upaya koneksi gagal berturut-turut, server memblokirnya. Upaya koneksi lebih lanjut dari klien itu menghasilkan kesalahan.

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

Jika Anda mengalami kesalahan “host diblokir”, hindari meningkatkan nilai `max_connect_errors` variabel. Sebagai gantinya, selidiki penghitung diagnostik server dalam variabel `aborted_connects` status dan tabel `host_cach` e. Gunakan informasi yang dikumpulkan untuk mengidentifikasi dan memperbaiki klien yang mengalami masalah koneksi. Juga, perhatikan bahwa parameter ini tidak berpengaruh jika `skip_name_resolve` diatur ke 1 (default).

Lihat Manual Referensi MySQL untuk detail tentang hal berikut:

- Variabel max_connect_errors
- Kesalahan “Host diblokir”
- Variabel status aborted_connects
- Tabel Host_cache

Menerapkan penyatuan koneksi

Peristiwa penskalaan dapat menambahkan lebih banyak server aplikasi, yang pada gilirannya dapat menyebabkan server DB melebihi nomor koneksi aktif yang dimuat penuh. Penambahan kumpulan koneksi atau lapisan proxy antara server aplikasi dan database bertindak seperti corong, mengurangi jumlah total koneksi pada database. Tujuan utama proxy adalah penggunaan kembali koneksi database dengan cara multiplexing.

Di satu sisi, proxy terhubung ke database dengan jumlah koneksi yang terkontrol. Di sisi lain, proxy menerima koneksi aplikasi. Ini juga menyediakan fitur tambahan, seperti cache kueri, buffering koneksi, penulisan ulang kueri dan perutean, dan penyeimbangan beban. Lapisan kolam koneksi perlu dikonfigurasi untuk menjaga jumlah maksimum koneksi ke database di bawah nomor yang dimuat penuh. Amazon RDS Proxy adalah proxy yang dikelola sepenuhnya yang dapat Anda terapkan untuk tujuan ini. Amazon RDS Proxy tidak memerlukan perubahan kode untuk sebagian besar aplikasi, dan Anda tidak perlu mengelola infrastruktur tambahan apa pun untuk mengimplementasikan solusinya.

Anda juga dapat menjelajahi proxy pihak ketiga berikut yang dapat digunakan dengan Aurora MySQL kompatibel:

- ProxySQL
- MariaDB MaxScale
- ScaleArc
- Data Heimdall

Hindari badai koneksi

Pertimbangkan bagaimana kumpulan koneksi Anda berperilaku jika database kelebihan beban atau replika jatuh terlalu jauh di belakang node utama. Saat mengonfigurasi server proxy atau kumpulan koneksi, pastikan Anda tidak mengatur ulang seluruh kumpulan koneksi berdasarkan respons basis data yang lambat (disebabkan oleh masalah perangkat keras atau penyimpanan yang mendasarinya atau kendala sumber daya DB).

Tiba-tiba memulai ratusan koneksi menghasilkan badai koneksi karena sejumlah besar permintaan untuk koneksi baru ke database semuanya dimulai pada saat yang sama. Badai itu intensif sumber daya. Membuat koneksi database baru di MySQL adalah operasi yang mahal karena backend bertukar beberapa paket jaringan untuk jabat tangan awal, memunculkan proses baru, mengalokasikan memori, menangani otentikasi, dan sebagainya. Jika sejumlah besar permintaan diterima dalam waktu singkat, database dapat tampak tidak responsif.

MySQL memiliki mekanisme untuk melindungi terhadap lonjakan permintaan koneksi seperti itu. `back_log`Variabel dapat diatur ke jumlah permintaan yang dapat ditumpuk selama waktu singkat sebelum MySQL sesaat berhenti menjawab permintaan baru. Nilai diberlakukan oleh utas penanganan koneksi, yang dengan sendirinya mungkin kewalahan oleh badai koneksi. Untuk informasi selengkapnya, lihat Manual Referensi MySQL.

Jika koneksi Anda dikonfigurasi untuk mengatur ulang ketika database lambat, Anda akan memulai siklus lagi dan lagi. Demikian pula, jika Anda mengantisipasi peningkatan mendadak dalam lalu lintas database pada waktu-waktu tertentu di siang hari (misalnya, ketika pasar saham terbuka), panaskan dulu kumpulan koneksi Anda sehingga Anda tidak mencoba membuka banyak koneksi pada saat yang sama ketika beban lalu lintas tinggi dimulai.

Menyetel beban kerja kueri Anda

Beban kerja yang disetel dengan baik akan membawa Anda jauh dalam mencapai solusi stabil untuk pertumbuhan berlebih. Jika beban kerja Anda tidak disetel dengan baik, apa pun kekuatan cluster Amazon Aurora yang Anda gunakan, Anda akan mengalami kemacetan yang akan menurunkan kinerja dan memengaruhi pengalaman pengguna aplikasi Anda. Praktik terbaik adalah memiliki proses sejak awal yang membantu Anda mengidentifikasi dan menyetel pertanyaan bermasalah di sistem Anda.

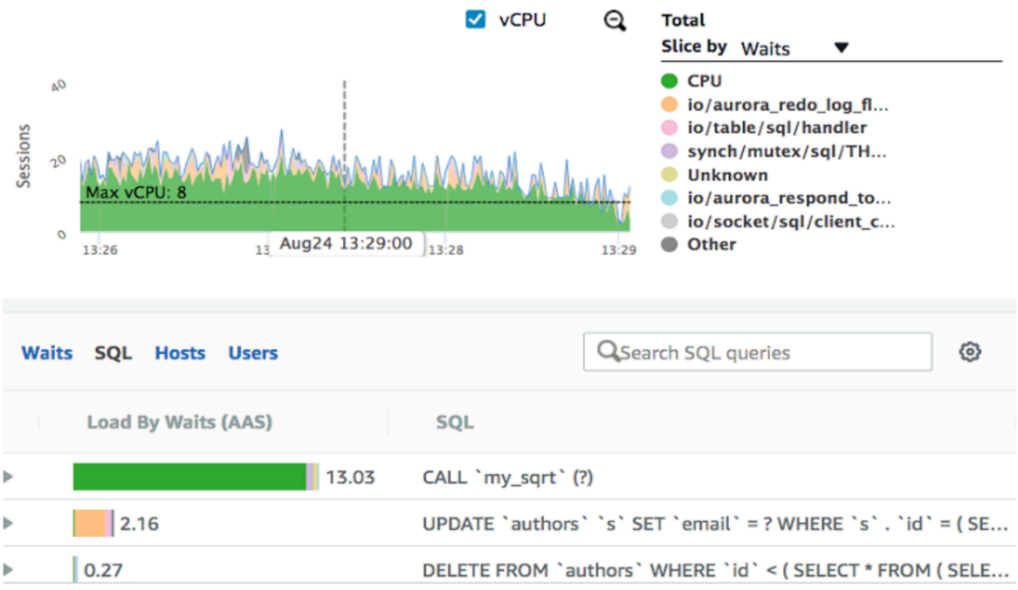
Melacak dan menyetel kueri bermasalah di beban kerja Anda

Saat menghadapi hypergrowth, memiliki beban kerja yang disetel dengan baik adalah setengah dari pertempuran yang dimenangkan. Untuk memahami sifat beban kerja real-time dan masalah kinerja yang dihadapi cluster Aurora Anda, pastikan bahwa tim Anda mengumpulkan pertanyaan bermasalah dari penulis dan instance pembaca Anda. Kueri bermasalah ini perlu disetel agar beban kerja Anda berjalan pada kondisi optimal. Amazon Aurora MySQL-Compatible Edition memberi Anda dua cara untuk mencapai hal ini:

- Wawasan Performa
- Log kueri lambat

Menggunakan Performance Insights

Performance Insights melacak beban pada penulis Aurora atau instance pembaca berdasarkan rata-rata sesi aktif (AAS). Nilai AAS dihitung dengan menggunakan sampling dan jumlah sesi aktif yang menunggu CPU untuk mengambil beban kerja kueri mereka dan memprosesnya. Performance Insights menyediakan antarmuka grafis tempat Anda dapat memeriksa pernyataan SQL yang menyebabkan beban tertinggi dengan menunggu sesi aktif.



Pada tangkapan layar sebelumnya, panggilan ke prosedur yang `my_sqrt` disimpan menyebabkan rata-rata 13, 03 sesi menunggu muatannya diproses. Langkah logis selanjutnya adalah menyetel prosedur ini. Anda harus mengidentifikasi pernyataan SQL pada pembaca dan penulis yang menyebabkan pemuatan pada instans masing-masing dan menyetelnya untuk meningkatkan kinerja hingga nilai AAS turun dan tetap berada di bawah garis putus-putus vCPU Maks di Performance Insights. Jika Anda telah mencapai batas tertinggi dengan upaya penyetelan Anda dan masih melihat AAS di atas garis vCPU Max, Anda dapat memilih kelas instans yang lebih besar untuk menangani beban kerja Anda. Jangan memilih instance yang lebih besar tanpa terlebih dahulu mencoba menyetel beban kerja kueri Anda, karena lalu lintas yang meningkat akan mulai mengekspos garis kesalahan yang dibuat oleh kueri buruk di beban kerja Anda.

Menggunakan log kueri lambat dan menerbitkannya ke CloudWatch

Log kueri lambat adalah fitur MySQL asli dan melengkapi Performance Insights. Praktik terbaik adalah menggunakan kedua metode ini untuk tetap berada di depan pertanyaan bermasalah yang dapat menyebabkan malapetaka pada instans Anda. Log kueri lambat mencatat kueri apa pun yang lebih lambat dari variabel dinamis`long_query_time`. Variabel ini dapat diatur tanpa restart ke instance cluster Anda.

Untuk memberikan fleksibilitas dan isolasi dalam latihan penyetelan, gunakan kelompok parameter terpisah untuk instance penulis dan pembaca Anda. Ini sangat penting jika Anda menggunakan pembagian baca-tulis. Siapkan batas yang nyaman untuk `long_query_time` instans klaster Anda

berdasarkan kebutuhan Anda. Saat Anda menyetel beban, Anda dapat membidik nilai agresif dalam `long_query_time` variabel, karena Anda dapat mengatur ambang batas pada tingkat milidetik. Dengan konkurensi tinggi dan beban kerja yang disetel dengan baik, hampir semua kueri Anda harus berjalan dalam milidetik.

Saat Anda mengatur Amazon Aurora MySQL-Compatible Edition untuk mencatat kueri lambat ke file, Aurora MySQL-Compatible akan menulis log kueri lambat ke sistem MySQL-Compatible file Aurora dan menyimpannya selama 24 jam. Untuk menyimpan log kueri lambat untuk jangka waktu yang lebih lama untuk analisis, publikasikan ke Amazon CloudWatch. Anda juga dapat membuat CloudWatch Dasbor untuk memantau kueri lambat Anda. Untuk informasi selengkapnya, lihat posting blog Membuat CloudWatch dasbor Amazon untuk memantau Amazon RDS dan Amazon Aurora MySQL. Selain menganalisis kueri lambat Anda di Amazon CloudWatch, Anda dapat membuat profil log kueri lambat dengan menggunakan pt-query-digest, alat di Percona Toolkit.

Anda juga dapat memilih untuk mengotomatiskan proses pengunduhan dan pembuatan profil kueri ini untuk efisiensi yang lebih tinggi di tim Anda. Tim Anda harus memeriksa kueri yang sering berjalan dan untuk interval yang lebih lama, dan memprioritaskan penyetelannya. Bertujuan untuk keadaan di mana sangat sedikit kueri yang dicatat di log kueri lambat Anda, dan Anda dapat mengurangi `long_query_time` untuk menjadi lebih agresif seperti yang Anda pahami dan sesuaikan beban kerja Anda.

Panduan untuk penyetelan kueri

Setelah Anda mengidentifikasi kueri bermasalah dalam beban kerja Anda, setiap kueri harus disetel. Gunakan panduan penyetelan berikut untuk membantu beban kerja Anda berjalan lebih efisien.

Minimalkan jumlah baris yang dipindai

Sepertinya dasar, ini adalah saran yang bagus untuk digunakan saat Anda menyetel kueri. Gunakan pernyataan EXPLOW, dan tinjau kolom baris untuk melihat berapa banyak baris yang dipindai pengoptimal pada setiap gabungan. Cobalah untuk mengurangi jumlah baris yang dipindai dengan membuat indeks optimal, lalu jelaskan kembali kueri Anda untuk mengonfirmasi pekerjaan Anda. Untuk informasi selengkapnya, lihat dokumentasi MySQL.

Jika Anda menggunakan tabel yang dipartisi, selalu kueri dengan klausa WHERE yang memungkinkan pemangkasan partisi sehingga pengoptimal tidak harus memindai setiap partisi. Jika klausa WHERE Anda berisi konstanta untuk kolom yang dipartisi, pengoptimal mengetahui partisi mana yang harus dicari, dan ini membuat kueri Anda lebih efisien.

Aspek lain dari saran ini adalah desain database Anda. Semakin sedikit tabel dalam kueri Anda, semakin cepat kueri Anda. Jika Anda dapat mendenormalisasi desain database Anda, Anda bisa mendapatkan pengoptimal untuk memindai lebih sedikit baris, menghasilkan kinerja kueri yang lebih cepat.

Minimalkan penggunaan tabel sementara dan tabel sementara pada disk

MySQL-Compatible Pengoptimal Aurora membuat tabel sementara baik pada RAM maupun pada disk jika tidak bisa mendapatkan hasil yang diinginkan dari kueri Anda langsung dari indeks. Akibatnya, sebagian besar penyetelan adalah memiliki indeks yang tepat yang melayani beban kerja Anda. Namun, mungkin ada kueri dalam beban kerja Anda yang tidak dapat hanya mengandalkan indeks, sehingga beberapa operasi mungkin dilakukan dalam file sementara. Ini baik-baik saja selama Anda menyimpannya seminimal mungkin, dan Anda memastikan bahwa sangat sedikit tabel yang dibuat pada disk. MySQL membuat tabel disk ketika ukuran tabel sementara terlalu besar untuk disimpan di memori. Logika yang MySQL gunakan untuk memeriksa ukuran tabel sementara internal adalah yang lebih kecil dari dua nilai variabel tmp_table_size dan max-heap-table-size. Anda dapat menyetel variabel-variabel ini ke nilai optimal berdasarkan beban kerja Anda sehingga dalam kasus di mana Anda tidak dapat mencegah tabel sementara, Anda mendorongnya ke disk hanya pada kesempatan yang jarang terjadi.

Hindari jenis file

Jika beban kerja Anda memiliki banyak `ORDER BY` kueri, cara terbaik untuk menyelesaikannya adalah dengan menggunakan indeks yang tepat pada tabel Anda. Pastikan indeks multicolumn Anda dirancang dengan baik untuk menghindari penyortiran file. Penyortiran tidak dapat terjadi pada kolom jika kolom sebelumnya tidak dipindai dengan konstanta (`in`,,, > <!=, dan tidak `BETWEEN` akan mengizinkan pengurutan pada kolom berikutnya ke kanan). Cara optimal untuk mengurutkan di MySQL adalah dengan menempatkan indeks multicolumn yang memposisikan kolom yang berisi nilai konstan yang disediakan dalam kueri di sebelah kiri kolom penyortiran dalam struktur yang berdekatan. Jika dalam upaya terakhir, kueri Anda tidak dapat mengembalikan hasil tanpa pengurutan file, pindahkan penyortiran ke aplikasi.

Hindari menjalankan kueri agregasi pada konkurensi tinggi

Beban kerja Anda mungkin memiliki sejumlah kecil kueri agregasi untuk memenuhi beberapa fungsi dalam aplikasi Anda. Kasus penggunaan ini membutuhkan banyak kehati-hatian. Mesin InnoDB diarahkan untuk pemuatan pemrosesan transaksi online (OLTP) yang tepat, tetapi bahkan beberapa kueri kelompok-demi pada konkurensi tinggi bisa sangat berat pada CPU dan dapat dengan cepat

menurunkan kinerja cluster Anda. Untuk mengatasi kasus penggunaan di mana Anda memerlukan kumpulan hasil agregat, pra-agregat data dalam tabel siap baca sehingga Anda dapat menghindari grup berdasarkan kueri sama sekali.

Uji kueri Anda untuk konkurensi

Saat menyetel kueri individual, ingatlah bahwa kueri ini berjalan bersamaan pada beberapa vCPU di Aurora. MySQL-Compatible Kueri Anda mungkin berjalan dalam beberapa milidetik di lingkungan pengujian Anda dalam satu kali proses. Tapi ini bukan gambaran keseluruhannya. Pastikan untuk menguji kueri Anda dengan tingkat konkurensi yang diharapkan pada klaster produksi Anda dan benchmark kinerjanya. Lepaskan kueri ke produksi hanya jika memenuhi tujuan konkurensi Anda. Pastikan Anda menggunakan pengoptimal `hint sql_no_cache` dalam skrip pengujian Anda sehingga Anda menghindari pengambilan hasil dari cache. Anda dapat menggunakan alat seperti mysqlslap untuk melakukan tes pada konkurensi dan benchmark hasilnya.

Menyetel beban kerja tulis

Menerapkan load-balancing dan membebaskan instance penulis akan membantu beban kerja tulis Anda berkinerja lebih baik selama lonjakan tinggi. Untuk mencapai kinerja penulisan yang lebih baik pada konkurensi tinggi, ikuti langkah-langkah tambahan ini.

Pindahkan integritas referensial ke lapisan aplikasi

Meskipun pemeriksaan integritas referensial penting, dengan hyperscaling, mereka dapat merusak beban Anda. Untuk setiap penulisan, pemindaian tambahan harus dilakukan sebelum penulisan itu sendiri dijalankan, dan ini menghasilkan kinerja yang buruk. Jika aplikasi Anda memerlukan pemeriksaan integritas yang ketat, buat mereka ke dalam lapisan aplikasi untuk mencegahnya membatasi penulisan Anda.

Hindari menggunakan kunci primer yang berat

Jaga agar kunci utama Anda tetap terang. Mesin penyimpanan InnoDB menambahkan kunci utama ke setiap indeks lain yang Anda buat di tabel Anda. Ketika kunci utama Anda besar, itu mempengaruhi ukuran indeks. Penyimpanan dan pengambilan halaman data akan melambat jika kunci utama cukup besar. Contoh umum adalah penggunaan pengidentifikasi unik universal sebagai kunci utama. Ini bukan pendekatan yang baik jika Anda menargetkan kinerja pada lingkungan hyperscaled.

Gunakan pertukaran partisi untuk memuat data ke dalam tabel yang dipartisi

Jika Anda menulis kumpulan besar data ke dalam tabel yang dipartisi, kombinasi LOAD DATA FROM S3 dan pertukaran partisi dapat meningkatkan kinerja, karena tabel utama tidak diakses untuk sisipan. Pertukaran partisi melibatkan bahasa definisi data (DDL), dan menempatkan kunci metadata di meja Anda. Pastikan bahwa ini dilakukan ketika ada permintaan minimal atau tidak ada yang berjalan di atas meja sehingga pertukaran partisi DDL dapat memperoleh kunci metadata tanpa menunggu. Pertukaran itu sendiri hanya membutuhkan milidetik untuk menyelesaikannya.

Hapus indeks yang tidak digunakan

InnoDB mengoptimalkan rencana kueri berdasarkan pertumbuhan data Anda, dan sebaiknya periksa indeks yang tidak digunakan di Database Anda dan menghapusnya. Indeks yang tidak digunakan menggunakan IO saat aplikasi menulis data ke dalam tabel. Periksa daftar indeks yang tidak digunakan, dan verifikasi bahwa itu bukan indeks yang digunakan dalam situasi yang jarang terjadi, seperti laporan triwulanan. Juga, perhatikan bahwa beberapa indeks digunakan untuk menegakkan keunikan atau pemesanan dan juga harus dipertimbangkan.

Pastikan versi baris lama dibersihkan secara efisien

Dalam implementasi InnoDB dari kontrol konkurensi multiversi (MVCC), ketika catatan dimodifikasi, versi (lama) saat ini dari data yang dimodifikasi pertama kali dicatat sebagai catatan undo dalam log undo. Nilai panjang daftar riwayat (HLL) yang berkembang menunjukkan bahwa utas pengumpulan sampah InnoDB (utas pembersihan) tidak mengikuti beban kerja penulisan, atau pembersihan diblokir oleh kueri atau transaksi yang berjalan lama. Ketika pengumpulan sampah diblokir atau ditunda, database dapat mengembangkan lag pembersihan substansional yang dapat berdampak negatif pada kinerja kueri. Anda dapat menggunakan rekomendasi berikut untuk mengoptimalkan proses pembersihan.

- Jaga transaksi tetap kecil.
- Untuk kueri baca, gunakan tingkat isolasi READ COMMITTED.
- Tingkatkan jumlah thread pembersihan (innodb_purge_threads dan innodb_purge_batch_size). Perhatikan bahwa menyetel parameter ini memerlukan reboot.
- Pantau HLL secara teratur, dan selesaikan masalah beban kerja apa pun yang mencegah pengumpulan sampah berkembang.

Pastikan logging tidak menyebabkan pertengkaran tambahan

Log kueri umum mencatat koneksi klien dan pemutusan juga sebagai tambahan untuk semua pernyataan yang diterima oleh server dalam urutan mereka diterima. Saat diaktifkan, logging sinkron, yang dapat menyebabkan penalti kinerja yang substansif pada sistem yang sibuk. Kecuali diperlukan, kami sarankan untuk menonaktifkan log umum.

Log kueri lambat mencatat pernyataan yang membutuhkan waktu lebih lama dari jumlah detik long_query_time untuk dijalankan, dengan pengaturan default 10 detik. Ketika pengaturan diatur ke 0, semua pernyataan dicatat secara sinkron, yang dapat menyebabkan penalti kinerja pada database yang sibuk.

Beban gudang

Meningkatkan waktu respons dan meningkatkan sumber daya yang tersedia untuk alur kerja penting mungkin memerlukan penghapusan beban asing. Banyak solusi yang tercakup dalam bagian ini adalah keputusan trade-off. Mereka memiliki konsekuensi terhadap aplikasi, dan mereka harus dipertimbangkan dengan cermat. Pertimbangkan ini menggunakan solusi ini dalam situasi berikut:

- Anda sudah berada di instance ukuran terbesar, terutama untuk instance database utama yang dapat ditulis.
- Sebagai upaya terakhir untuk menyediakan ruang kepala yang cukup dalam jangka pendek untuk menerapkan perubahan lainnya.

Perubahan langsung meliputi yang berikut:

- Pindahkan lalu lintas baca nonkritis dari instans DB utama.
- Arsipkan atau bersihkan data lama.
- Hapus integritas referensial.
- Nonaktifkan binlog (jika digunakan).
- Menunda proses ekstrak, transformasi, beban (ETL) non-kritis.
- Menangguhkan atau menurunkan fitur aplikasi yang tidak penting.

Sebelum melakukan tindakan ini, evaluasi mereka dalam konteks tujuan dan risiko bisnis jangka panjang.

Pisahkan beban kerja baca dan tulis

Teknik umum saat menjalankan aplikasi yang didukung oleh MySQL adalah dengan membongkar operasi baca dari instance database penulis (primer) dan ke satu atau lebih replika database read-only. Dengan membongkar pembacaan, Anda dapat mengurangi beban keseluruhan pada instance database utama dan memberi ruang untuk penulisan. Pastikan untuk menargetkan hanya pembacaan yang tidak bergantung pada konsistensi baca-setelah-tulis langsung untuk replika. Pendekatan yang lebih efisien adalah memindahkan semua lalu lintas baca ke replika dan berencana untuk mencoba kembali baca-setelah-tulis jika terjadi penundaan replikasi. Ada beban kerja baca independen yang dapat diturunkan, seperti layanan pelaporan. Pembacaan lain akan memerlukan perubahan di tingkat aplikasi, di mana konteks mengapa pembacaan dikeluarkan sudah diketahui dengan baik.

Pendekatan alternatif adalah menerapkan solusi proxy database sebagai perantara antara aplikasi dan database, yang dapat menyediakan fungsi pemisahan baca-tulis dan perutean kueri, tanpa kesadaran aplikasi. ProxySQL, MariaDB, ScaleArc MaxScaledan Heimdall Data adalah beberapa solusi proxy kompatibel MySQL yang tersedia. Produk ini menawarkan beberapa fitur tambahan, seperti caching atau multiplexing koneksi. Ketika Anda mengalami lonjakan lalu lintas yang tiba-tiba dan menerapkan teknologi baru di tumpukan aplikasi Anda, sebaiknya mulai dengan fungsionalitas dasar untuk memisahkan read/write kueri dan menguji perilaku dan kinerja sebelum menggunakan fitur tambahan yang mungkin memiliki efek samping yang tidak diinginkan.

Arsipkan atau bersihkan data lama

Teknik lain untuk meningkatkan kinerja database adalah dengan membongkar data historis ke tabel lain, database, atau Amazon Simple Storage Service (Amazon S3). Banyak database menyimpan semua data sebaris untuk seluruh riwayat aplikasi. Dalam keadaan normal dalam aplikasi yang dihadapi pengguna biasa, ini memberi pengguna kemampuan untuk melihat semua pesanan historis mereka. Ketika permintaan melonjak tiba-tiba, banyak pengguna yang aktif pada aplikasi mungkin baru atau fokus untuk menempatkan pesanan baru. Jika data historis berada online, dalam satu tabel yang berisi miliaran baris, senyawa ini. Tabel tersebut kemungkinan juga memiliki indeks yang besar. Baik data tabel dan indeks disimpan dalam struktur pohon. Memiliki lebih banyak entri dalam tabel membutuhkan lebih banyak level pada pohon, yang membutuhkan lebih banyak I/O operasi untuk mengakses baris. Ini meningkatkan waktu akses untuk menemukan catatan individu. Lebih penting lagi, ini menyebabkan sebagian besar indeks yang tidak dibutuhkan menjadi penduduk di cache halaman database (kumpulan buffer InnoDB).

Mengarsipkan data lama ke tabel terpisah, database terpisah, atau Amazon S3 dapat mengurangi waktu akses bagi pengguna akhir, membebaskan cache berharga, dan meningkatkan kinerja aplikasi secara keseluruhan. Mengarsipkan data yang lebih lama, sebelum acara (misalnya, mempertahankan hanya 30 atau 90 hari) membatasi ukuran tabel dan indeks pada tabel kritis. Perubahan ini mengharuskan aplikasi dimodifikasi untuk menanyakan data lama dari lokasi sekunder hanya untuk sebagian kecil pengguna yang secara eksplisit meminta untuk melihat data historis.

Menunda proses ETL yang tidak kritis

Ekstraksi dari sistem database utama untuk proses ETL dapat menghadirkan risiko stabilitas untuk beban kerja yang sangat transaksional dan bersamaan selama kondisi hyperscale. Proses ETL dikenal dengan karakteristik berikut:

- Long-running transaksi
- Kunci yang luas
- CPU, memori, dan I/O konsumsi
- Perselisihan dengan beban kerja transaksional yang melayani jalur pengguna akhir yang kritis.

Proses ETL yang bervariasi atau tidak dapat diprediksi (misalnya, kueri seperti `INSERT INTO ... SELECT FROM ...;`) menambah variabilitas keseluruhan dalam beban dan pertengkaran, meningkatkan risiko stabilitas.

Jika memungkinkan, tunda atau kurangi frekuensi di mana proses ETL berjalan, terutama jika mereka tidak menyediakan fungsi penting. Untuk proses ETL kritis, modifikasi mereka sehingga mereka beroperasi dalam unit kerja terbatas, seperti batch pemrosesan jumlah baris tetap (misalnya, menggunakan `LIMIT` klausa), menggunakan transaksi terpisah, atau menarik sejumlah kecil data selama periode waktu yang lama untuk mengurangi permintaan sumber daya puncak dari instans DB.

Matikan fitur aplikasi yang kurang kritis

Dengan anggun merendahkan pengalaman pengguna atau menghapus fitur noncore saat menangani lonjakan permintaan menghemat sumber daya database untuk fungsi-fungsi penting. Ini mungkin memerlukan sedikit perubahan dalam pengalaman pelanggan, tetapi aliran yang berbeda lebih baik daripada situs yang sedang down. Mungkin personalisasi di halaman depan situs yang selalu melakukan panggilan database dapat dinonaktifkan sementara. Atau Anda dapat berhenti

menyajikan penawaran khusus kepada pelanggan yang kembali saat memilih produk. Mematikan sementara fitur dapat memungkinkan halaman untuk di-cache atau dikirimkan tanpa memerlukan akses database.

Contoh lain termasuk frontend yang memiliki mekanisme polling memeriksa perubahan data yang memerlukan panggilan database. Mengurangi frekuensi polling akan segera menghasilkan lebih sedikit panggilan database. Antarmuka pengguna yang memerlukan pagination atau memberi pengguna opsi untuk mengambil set hasil yang diurutkan lebih jauh dari hasil teratas memerlukan panggilan database yang semakin mahal. Batasi jumlah halaman dalam set hasil untuk melindungi database dari panggilan database yang paling mahal. Gunakan bendera fitur di lapisan aplikasi untuk memungkinkan tim operasi mematikan atau menurunkan fitur saat beban aplikasi atau basis data meningkat.

Penyetelan parameter

Terlepas dari parameter terkait koneksi, ada beberapa parameter yang dapat Anda atur untuk meningkatkan kinerja cluster Edisi Amazon Aurora MySQL yang kompatibel dengan Amazon Aurora Anda saat dihadapkan dengan hypergrowth. Saat mengubah parameter database apa pun, penting untuk menggunakan praktik terbaik berikut:

- Ubah satu parameter pada waktu sehingga Anda dapat mengukur dampak perubahan.
- Beberapa parameter mungkin memerlukan periode pemanasan untuk mewujudkan efeknya setelah diubah. Pertimbangkan ini ketika Anda mengamati dan mengukur kinerja cluster yang kompatibel dengan Aurora MySQL Anda.
- Hindari nilai parameter yang salah, yang dapat mencegah instance database memulai.

Parameter ini meliputi:

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

Untuk panduan tentang mengonfigurasi parameter ini, lihat Parameter konfigurasi Aurora MySQL, Rekomendasi untuk fitur MySQL di AuroraMySQL, dan perilaku tabel sementara di Aurora MySQL dalam dokumentasi. AWS

Sumber daya

- Perjalanan untuk Mengadopsi Seri Arsitektur Cloud-Native: #1 - Mempersiapkan Aplikasi Anda untuk Hypergrowth (posting blog)
- Perjalanan untuk Mengadopsi Seri Arsitektur Cloud-Native: #2 - Memaksimalkan Throughput Sistem (posting blog)
- Menggunakan Amazon RDS Proxy
- Strategi pembuatan cache
- Mengaktifkan dan menonaktifkan Performance Insights
- Mesin Penyimpanan InnoDB
- Replika Aurora
- Konektor MariaDB/J
- MariaDB MaxScale
- ProxySQL
- Data Heimdall

Riwayat dokumen

Tabel berikut menjelaskan perubahan signifikan pada panduan ini. Jika Anda ingin diberi tahu tentang pembaruan masa depan, Anda dapat berlangganan umpan RSS.

Perubahan	Deskripsi	Tanggal
Publikasi awal	—	5 Oktober 2022

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.