Laporan Resmi AWS

Pola Desain Praktik Terbaik: Mengoptimalkan Performa Amazon S3

Pola Desain Praktik Terbaik: Mengoptimalkan Performa Amazon S3: Laporan Resmi AWS

Table of Contents

Pola Desain Praktik Terbaik: Mengoptimalkan Performa Amazon S3

Tanggal publikasi pertama: Juni 2019 (Revisi Dokumen)

Abstrak

Saat membangun aplikasi untuk mengunggah dan mengambil penyimpanan dari Amazon S3, ikuti panduan praktik terbaik AWS untuk mengoptimalkan performa. AWS juga menawarkan Pola Desain Performa yang lebih lengkap.

Pengantar

Aplikasi Anda dapat melakukan ribuan transaksi permintaan per detik saat mengunggah dan mengambil penyimpanan dari Amazon S3. Amazon S3 secara otomatis menskalakan ke tingkat permintaan tinggi. Misalnya, aplikasi Anda dapat mencapai setidaknya 3.500 permintaan `PUT/COPY/POST/DELETE` dan 5.500 `GET/HEAD` per detik per prefiks dalam sebuah bucket. Jumlah prefiks dalam bucket tidak terbatas. Anda dapat meningkatkan performa baca atau tulis dengan memparalelkan pembacaan. Misalnya, jika Anda membuat 10 prefiks di bucket Amazon S3 untuk memparalelkan pembacaan, Anda dapat menskalakan performa baca menjadi 55.000 permintaan baca per detik.

Beberapa aplikasi danau data di Amazon S3 memindai jutaan atau miliaran objek untuk kueri yang menjalankan banyak data dalam ukuran petabyte. Aplikasi danau data ini mencapai kecepatan transfer instans tunggal yang memaksimalkan penggunaan antarmuka jaringan untuk instans Amazon EC2 mereka, yang dapat mencapai 100 Gb/d pada satu instans. Aplikasi ini kemudian menggabungkan throughput di beberapa instans untuk mendapatkan beberapa terabit per detik.

Aplikasi lainnya peka terhadap latensi, seperti aplikasi olahpesan media sosial. Aplikasi ini dapat mencapai latensi objek kecil yang konsisten (dan latensi first- byte-out untuk objek yang lebih besar) kira-kira 100-200 milidetik.

Layanan AWS lainnya juga dapat membantu mempercepat kinerja untuk arsitektur aplikasi yang berbeda. Misalnya, jika Anda menginginkan tingkat transfer yang lebih tinggi melalui koneksi HTTP tunggal atau latensi milidetik satu digit, gunakan Amazon CloudFront atau Amazon ElastiCache untuk caching dengan Amazon S3.

Selain itu, jika Anda ingin transportasi data cepat dalam jarak jauh antara klien dan bucket S3, gunakan Amazon S3 Transfer Acceleration. Transfer Acceleration menggunakan lokasi edge terdistribusi secara global di CloudFront untuk mempercepat transportasi data yang terpisah secara geografis.

Jika beban kerja Amazon S3 Anda menggunakan enkripsi sisi server dengan AWS Key Management Service (SSE-KMS), lihat AWS KMS Limits dalam AWS Key Management ServicePanduan Developer untuk informasi tentang tingkat permintaan yang didukung untuk kasus penggunaan Anda.

Topik berikut menjelaskan panduan dan pola desain praktik terbaik untuk mengoptimalkan performa aplikasi yang menggunakan Amazon S3.

Panduan ini menggantikan panduan sebelumnya tentang mengoptimalkan performa untuk Amazon S3. Misalnya, sebelumnya panduan performa Amazon S3 merekomendasikan penamaan prefiks dengan karakter yang di-hash untuk mengoptimalkan performa pengambilan data yang sering. Anda sudah tidak perlu memberi nama prefiks pada performa secara acak, dan dapat menggunakan penamaan berdasarkan tanggal berurutan untuk prefiks Anda. Baca Panduan Performa dan Pola Desain Performa untuk informasi terbaru tentang pengoptimalan performa untuk Amazon S3.

Panduan Performa untuk Amazon S3

Untuk mendapatkan performa terbaik dari aplikasi Anda di Amazon S3, AWS merekomendasikan pedoman berikut.

Topik

Ukur Performa

Saat mengoptimalkan performa, lihat persyaratan throughput jaringan, CPU, dan Dynamic Random Access Memory (DRAM). Bergantung pada campuran permintaan untuk sumber daya yang berbeda ini, sebaiknya evaluasi tipe instans Amazon EC2 yang berbeda. Untuk informasi lebih lanjut tentang tipe instans, lihat Tipe Instans di Panduan Pengguna Amazon EC2 untuk Instans Linux.

Panduan ini juga membantu untuk melihat waktu pencarian DNS, latensi, dan kecepatan transfer data menggunakan alat analisis HTTP saat mengukur performa.

Skalakan Koneksi Penyimpanan secara Horizontal

Menyebarkan permintaan di banyak koneksi adalah pola desain umum untuk menskalakan performa secara horizontal. Saat Anda membangun aplikasi berperforma tinggi, anggap Amazon S3 sebagai sistem terdistribusi yang sangat besar, bukan sebagai titik akhir jaringan tunggal seperti server penyimpanan tradisional. Anda dapat mencapai performa terbaik dengan menerbitkan beberapa permintaan bersamaan ke Amazon S3. Sebarkan permintaan ini melalui koneksi terpisah untuk memaksimalkan bandwidth yang dapat diakses dari Amazon S3. Amazon S3 tidak membatasi jumlah koneksi yang dibuat ke bucket Anda.

Gunakan Pengambilan Rentang Byte

Dengan header Range HTTP dalam permintaan GET Object, Anda dapat mengambil rentang byte dari suatu objek, hanya mentransfer bagian yang ditentukan. Anda dapat menggunakan koneksi bersamaan ke Amazon S3 untuk mengambil rentang byte yang berbeda dari objek yang sama. Hal ini membantu Anda mencapai throughput agregat yang lebih tinggi dibandingkan permintaan keseluruhan objek tunggal. Mengambil rentang yang lebih kecil dari objek besar juga memungkinkan aplikasi Anda untuk meningkatkan waktu percobaan ulang saat permintaan terganggu. Untuk informasi lebih lanjut, lihat Mendapatkan Objek.

Ukuran umum untuk permintaan rentang byte adalah 8 MB atau 16 MB. Jika objek diletakkan (PUT) menggunakan unggahan multipart, ini adalah praktik yang baik untuk menempatkan (GET) objek tersebut ke ukuran bagian yang sama (atau setidaknya selaras dengan batas bagian) untuk performa terbaik. Permintaan GET dapat langsung mengatasi masing-masing bagian; misalnya, `GET ?partNumber=N`.

Permintaan Percobaan Ulang untuk Aplikasi Peka Latensi

Waktu tunggu dan percobaan ulang agresif membantu mendorong konsistensi latensi. Dengan skala besar Amazon S3, jika permintaan pertama lambat, permintaan yang dicoba ulang kemungkinan akan mengambil jalur yang berbeda dan cepat berhasil. AWS SDK memiliki nilai batas waktu dan percobaan ulang yang dapat dikonfigurasi yang dapat Anda sesuaikan dengan toleransi aplikasi tertentu.

Gabungkan Amazon S3 (Penyimpanan) dan Amazon EC2 (Komputasi) di Wilayah AWS yang Sama

Meskipun nama bucket S3 secara global unik, setiap bucket disimpan di Wilayah yang Anda pilih saat membuat bucket. Untuk mengoptimalkan performa, sebaiknya akses bucket dari instans Amazon EC2 di Wilayah AWS yang sama jika memungkinkan. Hal ini membantu mengurangi latensi jaringan dan biaya transfer data.

Untuk informasi lebih lanjut tentang biaya transfer data, lihat Harga Amazon S3.

Gunakan Amazon S3 Transfer Acceleration untuk Meminimalkan Latensi Karena Jarak

Amazon S3 Transfer Acceleration mengelola transfer file yang cepat, mudah, dan aman antara klien dan bucket S3 yang terpisah jauh. Transfer Acceleration memanfaatkan lokasi edge yang terdistribusi secara global di Amazon CloudFront. Saat data sampai di lokasi edge, data tersebut dirutekan ke Amazon S3 melalui jalur jaringan yang dioptimalkan. Transfer Acceleration sangat ideal untuk mentransfer data dalam ukuran gigabyte ke terabyte secara teratur di seluruh benua. Layanan ini juga berguna bagi klien yang melakukan pengunggahan ke bucket terpusat dari seluruh dunia.

Anda dapat menggunakan alat Perbandingan Kecepatan Amazon S3 Transfer Acceleration untuk membandingkan kecepatan antara pengunggahan yang dipercepat dan tidak dipercepat di seluruh Wilayah Amazon S3. Alat Perbandingan Kecepatan menggunakan unggahan multipart untuk mentransfer file dari peramban Anda ke berbagai Wilayah Amazon S3 dengan dan tanpa menggunakan Amazon S3 Transfer Acceleration.

Gunakan AWS SDK Versi Terbaru

AWS SDK menyediakan dukungan bawaan untuk banyak rekomendasi panduan untuk mengoptimalkan performa Amazon S3. SDK menyediakan API yang lebih sederhana untuk memanfaatkan Amazon S3 dari dalam aplikasi dan diperbarui secara rutin sesuai praktik terbaik terbaru. Misalnya, SDK menyertakan logika untuk mencoba kembali permintaan pada kesalahan HTTP 503 secara otomatis dan berinvestasi dalam kode untuk merespons dan beradaptasi dengan koneksi yang lambat.

SDK juga menyediakan Transfer Manager, yang mengotomatisasi koneksi penskalaan horizontal untuk mencapai ribuan permintaan per detik, menggunakan permintaan rentang byte jika sesuai. Gunakan AWS SDK versi terbaru untuk mendapatkan fitur pengoptimalan performa terbaru.

Anda juga dapat mengoptimalkan performa saat menggunakan permintaan REST API HTTP. Saat menggunakan REST API, Anda harus mengikuti praktik terbaik yang sama yang merupakan bagian dari SDK. Izinkan batas waktu dan percobaan ulang pada permintaan lambat, dan beberapa koneksi untuk memungkinkan pengambilan data objek secara paralel. Untuk informasi tentang penggunaan REST API, lihat Referensi API Amazon Simple Storage Service.

Pola Desain Performa untuk Amazon S3

Saat mendesain aplikasi untuk mengunggah dan mengambil penyimpanan dari Amazon S3, gunakan pola desain praktik terbaik kami untuk mencapai performa terbaik aplikasi Anda. Kami juga menawarkan Panduan Performa yang dapat Anda gunakan sebagai pertimbangan saat merencanakan arsitektur aplikasi.

Untuk mengoptimalkan performa, Anda dapat menggunakan pola desain berikut.

Topik

Menggunakan Caching untuk Konten yang Sering Diakses

Banyak aplikasi yang menyimpan data di Amazon S3 memberikan data yang bersifat “set fungsional” yang diminta berulang kali oleh pengguna. Jika beban kerja mengirimkan permintaan GET berulang untuk set objek umum, Anda dapat menggunakan cache seperti: Amazon CloudFront, Amazon ElastiCache, atau AWS Elemental MediaStore untuk mengoptimalkan performa. Adopsi cache yang berhasil akan menghasilkan latensi rendah dan tingkat transfer data yang tinggi. Aplikasi yang menggunakan caching juga mengirimkan lebih sedikit permintaan langsung ke Amazon S3, sehingga dapat membantu mengurangi biaya permintaan.

Amazon CloudFront adalah jaringan pengiriman konten (CDN) cepat yang secara transparan menyimpan data dari Amazon S3 dalam banyak points of presence (POP) yang terdistribusi secara geografis. Meskipun objek dapat diakses dari beberapa Wilayah, atau melalui internet, CloudFront memungkinkan data di-cache dekat dengan pengguna yang mengakses objek. Hal ini dapat menghasilkan tingginya performa pengiriman konten populer Amazon S3. Untuk informasi tentang CloudFront, lihat Panduan Developer Amazon CloudFront.

Amazon ElastiCache adalah cache dalam memori terkelola. Dengan ElastiCache, Anda dapat menyediakan instans Amazon EC2 yang meng-cache objek dalam memori. Caching ini mengurangi

banyak latensi GET dan meningkatkan throughput unduhan secara signifikan. Untuk menggunakan ElastiCache, ubah logika aplikasi untuk mengisi cache dengan objek panas dan memeriksa cache untuk objek panas sebelum memintanya dari Amazon S3. Untuk mengetahui contoh penggunaan ElastiCache untuk meningkatkan performa GET Amazon S3, lihat posting blog Turbocharge Amazon S3 with Amazon ElastiCache for Redis.

AWS Elemental MediaStore adalah sistem distribusi konten dan caching yang dibuat khusus untuk alur kerja video dan pengiriman media dari Amazon S3. MediaStore menyediakan API penyimpanan end-to-end khusus untuk video, dan direkomendasikan untuk beban kerja video yang peka terhadap performa. Untuk informasi tentang MediaStore, lihat Panduan Pengguna AWS Elemental MediaStore.

Batas Waktu dan Percobaan Ulang untuk Aplikasi Peka Latensi

Ada situasi tertentu saat aplikasi menerima respons dari Amazon S3 yang menunjukkan bahwa perlu adanya percobaan ulang. Amazon S3 memetakan nama objek dan bucket ke data objek yang terkait dengannya. Jika aplikasi menghasilkan tingkat permintaan yang tinggi (biasanya terus di tingkat dengan lebih dari 5.000 permintaan per detik untuk sedikit objek), kemungkinan akan menerima respons perlambatan HTTP 503. Jika kesalahan ini terjadi, setiap AWS SDK mengimplementasikan logika percobaan ulang otomatis menggunakan backoff eksponensial. Jika Anda tidak menggunakan AWS SDK, Anda harus menerapkan logika percobaan ulang saat menerima kesalahan HTTP 503. Untuk informasi tentang teknik back-off, lihat Percobaan Ulang Kesalahan dan Backoff Eksponensial di AWS di Referensi Umum Amazon Web Services.

Amazon S3 secara otomatis menskalakan sebagai respons terhadap tingkat permintaan baru yang berkelanjutan, mengoptimalkan performa secara dinamis. Saat Amazon S3 mengoptimalkan tingkat permintaan baru secara internal, Anda akan menerima respons permintaan HTTP 503 sementara hingga pengoptimalan selesai. Setelah Amazon S3 mengoptimalkan performa tingkat permintaan baru secara internal, semua permintaan umumnya dilayani tanpa percobaan ulang.

Untuk aplikasi yang peka terhadap latensi, Amazon S3 menyarankan untuk melacak dan mencoba kembali operasi yang lebih lambat secara berulang. Saat mencoba kembali permintaan, sebaiknya gunakan koneksi baru ke Amazon S3 dan melakukan pencarian DNS baru.

Saat membuat permintaan berukuran besar (misalnya, lebih dari 128 MB), sebaiknya lacak throughput yang dicapai dan coba kembali 5 persen permintaan yang paling lambat. Jika Anda membuat permintaan yang lebih kecil (misalnya, kurang dari 512 KB), yang latensi mediannya biasanya berada dalam rentang puluhan milidetik, sebaiknya coba kembali operasi GET atau PUT setelah 2 detik. Jika diperlukan pengulangan tambahan, praktik terbaiknya adalah back off. Sebagai

contoh, sebaiknya percobaan ulang setelah 2 detik dan percobaan ulang kedua setelah 4 detik tambahan.

Jika aplikasi Anda membuat permintaan berukuran tetap ke Amazon S3, maka waktu respons untuk setiap permintaan ini lebih konsisten. Dalam hal ini, strategi sederhananya adalah dengan mengidentifikasi 1 persen permintaan paling lambat dan mencobanya lagi. Bahkan satu percobaan ulang bisa efektif dalam mengurangi latensi.

Jika Anda menggunakan AWS Key Management Service (AWS KMS) untuk enkripsi sisi server, lihat Kuota dalam AWS Key Management Service Panduan Developer untuk informasi tentang tingkat permintaan yang didukung untuk kasus penggunaan Anda.

Penskalaan Horizontal dan Paralelisasi Permintaan untuk Throughput Tinggi

Amazon S3 adalah sistem terdistribusi yang sangat besar. Untuk membantu Anda memanfaatkan skalanya, sebaiknya skalakan permintaan paralel secara horizontal ke titik akhir layanan Amazon S3. Selain mendistribusikan permintaan dalam Amazon S3, jenis pendekatan penskalaan ini membantu mendistribusikan beban melalui beberapa jalur dalam jaringan.

Untuk transfer throughput tinggi, Amazon S3 menyarankan penggunaan aplikasi yang menggunakan beberapa koneksi ke data GET atau PUT secara paralel. Misalnya, ini didukung oleh Amazon S3 Transfer Manager di AWS Java SDK, dan sebagian besar AWS SDK lainnya menyediakan konstruksi serupa. Untuk beberapa aplikasi, Anda dapat memperoleh koneksi paralel dengan meluncurkan beberapa permintaan secara bersamaan di thread atau instans aplikasi yang berbeda. Pendekatan terbaik yang harus dilakukan tergantung pada aplikasi dan struktur objek yang Anda akses.

Anda dapat menggunakan SDK AWS untuk menerbitkan permintaan GET dan PUT secara langsung, bukan menggunakan pengelolaan transfer di AWS SDK. Dengan pendekatan ini, Anda dapat menyesuaikan beban kerja secara langsung, dan masih mendapat manfaat dari dukungan SDK untuk percobaan ulang dan penanganan respons HTTP 503 yang mungkin terjadi. Umumnya, saat Anda mengunduh objek besar dalam suatu Wilayah dari Amazon S3 ke Amazon EC2, sebaiknya buat permintaan bersamaan untuk rentang byte suatu objek pada granularitas 8–16 MB. Buat satu permintaan bersamaan untuk setiap 85–90 MB/dtk throughput jaringan yang diinginkan. Untuk mensaturasi kartu antarmuka jaringan (NIC) 10 Gb/d, Anda dapat menggunakan sekitar 15 permintaan bersamaan melalui koneksi terpisah. Anda dapat menaikkan skala permintaan bersamaan melalui lebih banyak koneksi untuk mensaturasi NIC yang lebih cepat, seperti NIC 25 Gb/d atau 100 Gb/d.

Mengukur performa penting saat Anda mengatur jumlah permintaan yang akan diterbitkan secara bersamaan. Sebaiknya mulai dengan satu permintaan sekaligus. Ukur bandwidth jaringan yang dicapai dan penggunaan sumber daya lainnya yang digunakan oleh aplikasi Anda dalam memproses data. Anda kemudian dapat mengidentifikasi sumber daya hambatan (yaitu, sumber daya dengan penggunaan tertinggi), dan jumlah permintaan yang mungkin berguna. Misalnya, jika pemrosesan satu permintaan memerlukan penggunaan CPU 25 persen, artinya empat permintaan bersamaan dapat diakomodasi.

Pengukuran sangat penting, dan sebaiknya konfirmasi penggunaan sumber daya seiring meningkatnya permintaan.

Jika aplikasi Anda menerbitkan permintaan langsung ke Amazon S3 menggunakan REST API, sebaiknya gunakan kumpulan koneksi HTTP dan gunakan kembali setiap koneksi untuk serangkaian permintaan. Dengan menghindari pengaturan koneksi per-permintaan, Anda tidak perlu melakukan slow-start TCP dan handshake Lapisan Soket Aman (SSL) pada setiap permintaan. Untuk informasi tentang menggunakan REST API, lihat Pengenalan REST API Amazon S3.

Terakhir, perhatikan DNS dan periksa kembali apakah permintaan sudah disebar di seluruh kolam alamat IP Amazon S3. Kueri DNS untuk siklus Amazon S3 melalui daftar besar titik akhir IP. Namun resolver cache atau kode aplikasi yang menggunakan ulang satu alamat IP tidak mendapat manfaat dari keragaman alamat dan penyeimbangan beban yang menyertainya. Alat utilitas jaringan seperti alat baris perintah `netstat` dapat menunjukkan alamat IP yang digunakan untuk komunikasi dengan Amazon S3, dan kami menyediakan panduan untuk konfigurasi DNS. Untuk informasi lebih lanjut tentang panduan ini, lihat Perutean permintaan.

Menggunakan Amazon S3 Transfer Acceleration untuk Mempercepat Transfer Data di Wilayah Berbeda

Amazon S3 Transfer Acceleration efektif untuk meminimalkan atau menghapus latensi yang disebabkan oleh jarak geografis antara klien yang tersebar di seluruh dunia dan aplikasi regional menggunakan Amazon S3. Transfer Acceleration menggunakan lokasi edge yang terdistribusi secara global di CloudFront untuk transportasi data. Jaringan edge AWS memiliki points of presence (POP) di lebih dari 50 lokasi. Saat ini, layanan ini digunakan untuk mendistribusikan konten melalui CloudFront dan untuk merespons kueri DNS yang dibuat ke Amazon Route 53 dengan cepat.

Jaringan edge juga membantu mempercepat transfer data masuk dan keluar dari Amazon S3. Jaringan ini sangat ideal untuk aplikasi yang mentransfer data di atau antarbenua, memiliki koneksi internet cepat, menggunakan objek besar, atau memiliki banyak konten yang harus diunggah.

Saat data sampai di lokasi edge, data tersebut dirutekan ke Amazon S3 melalui jalur jaringan yang dioptimalkan. Secara umum, semakin jauh Anda dari Wilayah Amazon S3, semakin tinggi peningkatan kecepatan yang dapat Anda peroleh dengan menggunakan Transfer Acceleration.

Anda dapat mengatur Transfer Acceleration pada bucket baru atau yang sudah ada. Anda dapat menggunakan titik akhir Amazon S3 Transfer Acceleration terpisah untuk menggunakan lokasi edge AWS. Cara terbaik untuk menguji apakah Transfer Acceleration membantu performa permintaan klien adalah menggunakan alat Perbandingan Kecepatan Amazon S3 Transfer Acceleration. Konfigurasi dan syarat jaringan bervariasi dari waktu ke waktu dan berbagai lokasi. Jadi, Anda dikenakan biaya hanya untuk transfer yang ada potensi Amazon S3 Transfer Acceleration untuk meningkatkan performa pengunggahan Anda. Untuk informasi tentang penggunaan Transfer Acceleration dengan AWS SDK yang berbeda, lihat Contoh Amazon S3 Transfer Acceleration.

Kontributor

Kontributor dokumen ini meliputi:

- Mai-Lan Tomsen Bukovec, VP, Amazon S3
- Andy Warfield, Kepala Rekayasawan Senior (Senior Principal Engineer), Amazon S3
- Tim Harris, Kepala Rekayasawan (Principal Engineer), Amazon S3

Revisi Dokumen

Untuk mendapatkan notifikasi tentang pembaruan laporan resmi ini, silakan berlangganan umpan RSS.

update-history-change	update-history-description	update-history-date
Diperbarui	Ditinjau untuk akurasi teknis	10 Maret 2021
Publikasi pertama	Publikasi pertama	1 Juni 2019

Pemberitahuan

Pelanggan bertanggung jawab untuk membuat penilaian independen mereka sendiri atas informasi dalam dokumen ini. Dokumen ini: (a) hanya disediakan sebagai informasi, (b) berisi penawaran produk dan praktik AWS saat ini, yang dapat berubah tanpa pemberitahuan, dan (c) tidak menjadi komitmen atau jaminan apa pun dari AWS dan afiliasi, pemasok, atau pemberi lisensinya. Produk atau layanan AWS disediakan “sebagaimana adanya” tanpa jaminan, representasi, atau ketentuan apa pun, baik tersurat maupun tersirat. Tanggung jawab dan kewajiban AWS kepada pelanggannya dikendalikan oleh perjanjian AWS, dan dokumen ini bukan bagian dari, juga tidak mengubah, perjanjian apa pun antara AWS dan pelanggannya.