Praktik terbaik untuk penyetelan kinerja AWS Glue untuk pekerjaan Apache Spark

: Praktik terbaik untuk penyetelan kinerja AWS Glue untuk pekerjaan Apache Spark

Table of Contents

Praktik terbaik untuk penyetelan kinerja AWS Glue untuk pekerjaan Apache Spark

Roman Myers, Takashi Onikura, dan Noritaka Sekiyama, Amazon Web Services ()AWS

Desember 2023 (riwayat dokumen)

AWS Glue menyediakan opsi berbeda untuk kinerja tuning. Panduan ini mendefinisikan topik utama untuk penyetelan Apache AWS Glue Spark. Ini kemudian memberikan strategi dasar bagi Anda untuk mengikuti ketika menyetel ini AWS Glue untuk pekerjaan Apache Spark. Gunakan panduan ini untuk mempelajari cara mengidentifikasi masalah kinerja dengan menafsirkan metrik yang tersedia di. AWS Glue Kemudian gabungkan strategi untuk mengatasi masalah ini, memaksimalkan kinerja dan meminimalkan biaya.

Panduan ini mencakup praktik penyetelan berikut:

- Kapasitas klaster skala
- Gunakan AWS Glue versi terbaru
- Kurangi jumlah pemindaian data
- Paralelisasi tugas
- Minimalkan overhead perencanaan
- Optimalkan shuffle
- Optimalkan fungsi yang ditentukan pengguna

Topik utama di Apache Spark

Bagian ini menjelaskan konsep dasar Apache Spark dan topik utama untuk penyetelan AWS Glue kinerja Apache Spark. Penting untuk memahami konsep dan topik ini sebelum membahas strategi penyetelan dunia nyata.

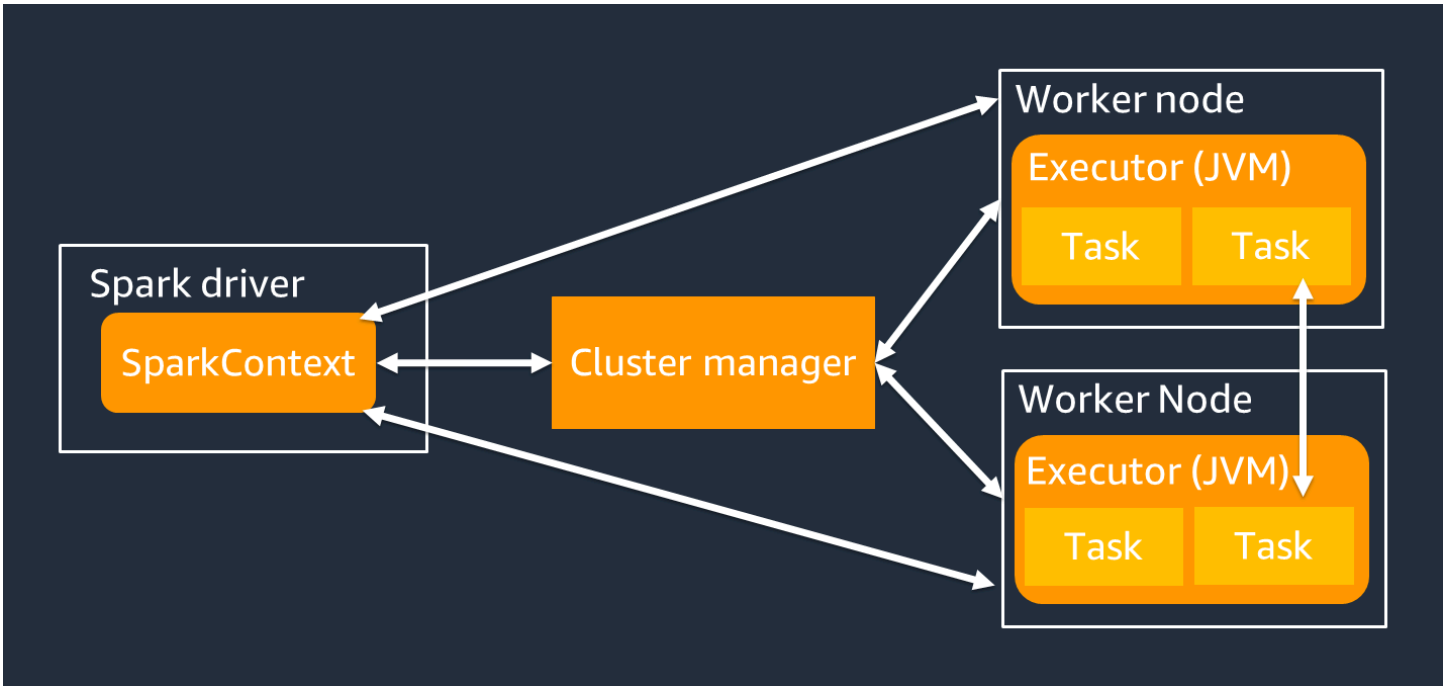
Arsitektur

Driver Spark terutama bertanggung jawab untuk membagi aplikasi Spark Anda menjadi tugas-tugas yang dapat diselesaikan pada pekerja individu. Pengemudi Spark memiliki tanggung jawab sebagai berikut:

- Berjalan `main()` di kode Anda
- Menghasilkan rencana eksekusi
- Penyediaan pelaksana Spark bersama dengan pengelola klaster, yang mengelola sumber daya di cluster
- Menjadwalkan tugas dan meminta tugas untuk pelaksana Spark
- Mengelola kemajuan dan pemulihan tugas

Anda menggunakan `SparkContext` objek untuk berinteraksi dengan driver Spark untuk menjalankan pekerjaan Anda.

Seorang eksekutor Spark adalah pekerja untuk menyimpan data dan menjalankan tugas yang diteruskan dari driver Spark. Jumlah pelaksana Spark akan naik dan turun dengan ukuran cluster Anda.

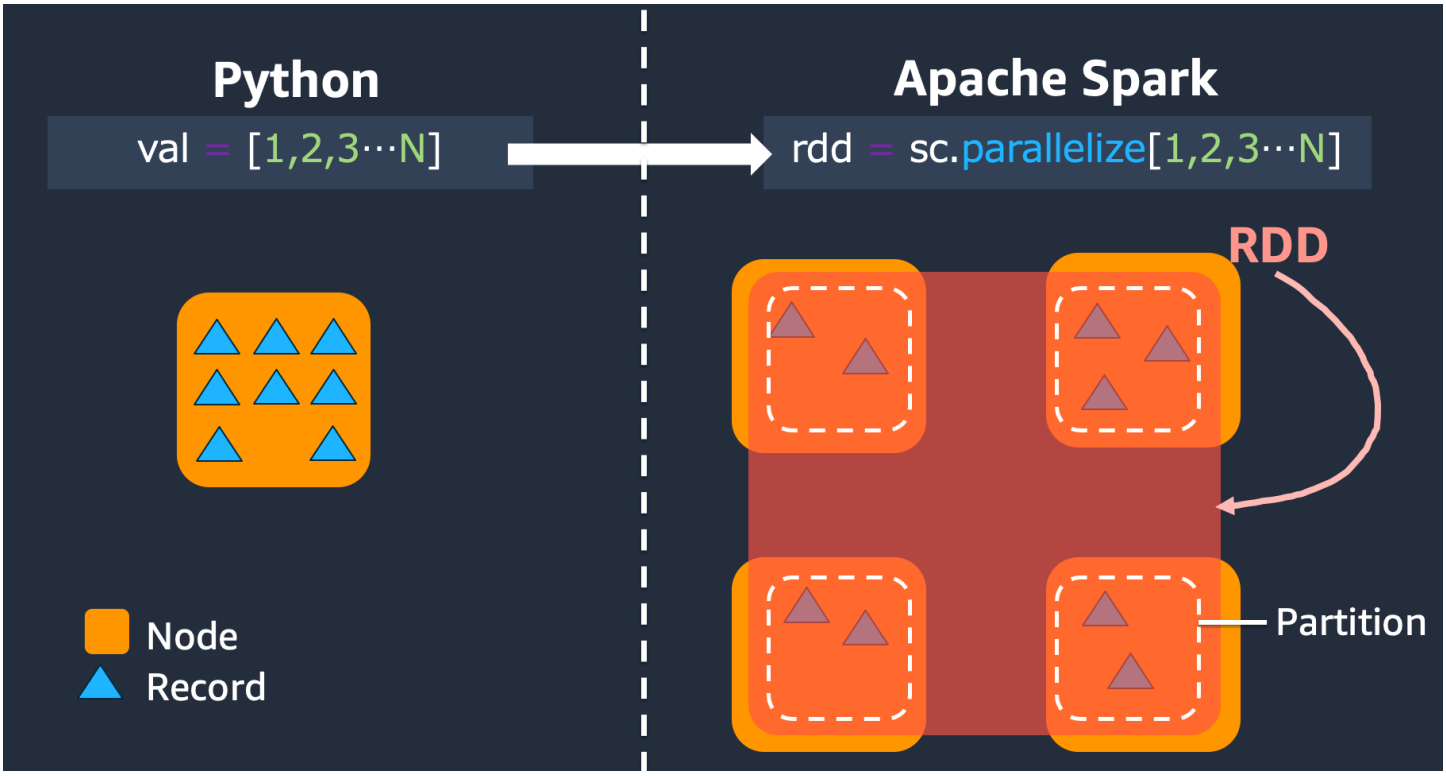


> **Note**
>
> Seorang eksekutor Spark memiliki beberapa slot sehingga banyak tugas untuk diproses secara paralel. Spark mendukung satu tugas untuk setiap inti CPU virtual (vCPU) secara default. Misalnya, jika seorang eksekutor memiliki empat core CPU, ia dapat menjalankan empat tugas bersamaan.

Dataset terdistribusi yang tangguh

Spark melakukan pekerjaan kompleks menyimpan dan melacak kumpulan data besar di seluruh pelaksana Spark. Saat Anda menulis kode untuk pekerjaan Spark, Anda tidak perlu memikirkan detail penyimpanan. Spark menyediakan abstraksi kumpulan data terdistribusi tangguh (RDD), yang merupakan kumpulan elemen yang dapat dioperasikan secara paralel dan dapat dipartisi di seluruh pelaksana Spark cluster.

Gambar berikut menunjukkan perbedaan dalam cara menyimpan data dalam memori ketika skrip Python dijalankan di lingkungan tipikal dan ketika dijalankan dalam kerangka Spark (). PySpark



- Python — Menulis `val = [1,2,3...N]` dalam skrip Python menyimpan data dalam memori pada mesin tunggal tempat kode berjalan.
- PySpark— Spark menyediakan struktur data RDD untuk memuat dan memproses data yang didistribusikan di seluruh memori pada beberapa pelaksana Spark. Anda dapat menghasilkan RDD dengan kode seperti`rdd = sc.parallelize[1,2,3...N]`, dan Spark dapat secara otomatis mendistribusikan dan menyimpan data dalam memori di beberapa pelaksana Spark.

  Dalam banyak AWS Glue pekerjaan, Anda menggunakan RDD melalui AWS Glue DynamicFramesdan Spark. DataFrames Ini adalah abstraksi yang memungkinkan Anda untuk menentukan skema data dalam RDD dan melakukan tugas tingkat yang lebih tinggi dengan informasi tambahan itu. Karena mereka menggunakan RDD secara internal, data didistribusikan secara transparan dan dimuat ke beberapa node dalam kode berikut:

  - DynamicFrame

```
dyf= glueContext.create_dynamic_frame.from_options(
    's3', {"paths": [ "s3://<YourBucket>/<Prefix>/"]},
    format="parquet",
    transformation_ctx="dyf"
)
```

- DataFrame

```
df = spark.read.format("parquet")
    .load("s3://<YourBucket>/<Prefix>")
```

RDD memiliki beberapa fitur berikut:

- RDD terdiri dari data yang dibagi menjadi beberapa bagian yang disebut partisi. Setiap eksekutor Spark menyimpan satu atau lebih partisi dalam memori, dan data didistribusikan di beberapa pelaksana.
- RDD tidak dapat diubah, artinya tidak dapat diubah setelah dibuat. Untuk mengubah DataFrame, Anda dapat menggunakan transformasi, yang didefinisikan di bagian berikut.
- RDD mereplikasi data di seluruh node yang tersedia, sehingga mereka dapat secara otomatis pulih dari kegagalan node.

Evaluasi malas

RDD mendukung dua jenis operasi: transformasi, yang membuat kumpulan data baru dari yang sudah ada, dan tindakan, yang mengembalikan nilai ke program driver setelah menjalankan perhitungan pada dataset.

- Transformasi — Karena RDD tidak dapat diubah, Anda dapat mengubahnya hanya dengan menggunakan transformasi.

  Misalnya, `map` adalah transformasi yang melewati setiap elemen dataset melalui fungsi dan mengembalikan RDD baru yang mewakili hasil. Perhatikan bahwa `map` metode tidak mengembalikan output. Spark menyimpan transformasi abstrak untuk masa depan, daripada membiarkan Anda berinteraksi dengan hasilnya. Spark tidak akan bertindak berdasarkan transformasi sampai Anda memanggil suatu tindakan.
- Tindakan — Menggunakan transformasi, Anda membangun rencana transformasi logis Anda. Untuk memulai perhitungan, Anda menjalankan tindakan seperti,`write`,`count`, `show` atau. `collect`

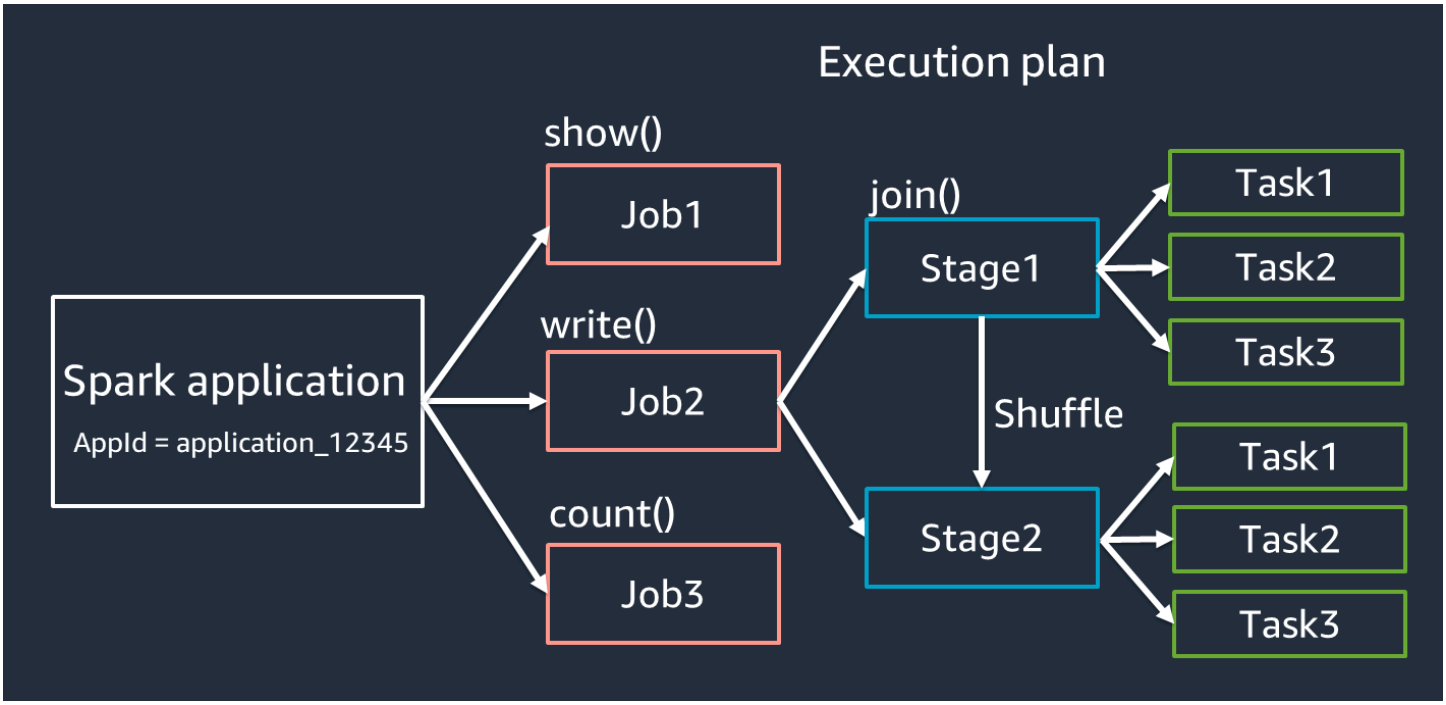
  Semua transformasi di Spark malas, karena mereka tidak langsung menghitung hasilnya. Sebagai gantinya, Spark mengingat serangkaian transformasi yang diterapkan ke beberapa dataset dasar, seperti objek Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon

S3). Transformasi dihitung hanya ketika suatu tindakan membutuhkan hasil untuk dikembalikan ke pengemudi. Desain ini memungkinkan Spark berjalan lebih efisien. Misalnya, pertimbangkan situasi di mana kumpulan data yang dibuat melalui `map` transformasi hanya dikonsumsi oleh transformasi yang secara substansional mengurangi jumlah baris, seperti. `reduce` Anda kemudian dapat meneruskan kumpulan data yang lebih kecil yang telah mengalami kedua transformasi ke driver, alih-alih meneruskan kumpulan data yang dipetakan yang lebih besar.

Terminologi aplikasi Spark

Bagian ini mencakup terminologi aplikasi Spark. Driver Spark membuat rencana eksekusi dan mengontrol perilaku aplikasi dalam beberapa abstraksi. Istilah berikut ini penting untuk pengembangan, debugging, dan penyetelan kinerja dengan UI Spark.

- Aplikasi — Berdasarkan sesi Spark (konteks Spark). Diidentifikasi oleh ID unik seperti`<application_XXX>`.
- Pekerjaan — Berdasarkan tindakan yang dibuat untuk RDD. Pekerjaan terdiri dari satu tahapan atau lebih.
- Tahapan — Berdasarkan shuffle yang dibuat untuk RDD. Sebuah panggung terdiri dari satu atau lebih tugas. Shuffle adalah mekanisme Spark untuk mendistribusikan kembali data sehingga dikelompokkan secara berbeda di seluruh partisi RDD. Transformasi tertentu, sepertijoin(), membutuhkan pengocokan. Shuffle dibahas secara lebih rinci dalam praktik penyetelan Optimalkan shuffles.
- Tugas — Tugas adalah unit minimum pemrosesan yang dijadwalkan oleh Spark. Tugas dibuat untuk setiap partisi RDD, dan jumlah tugas adalah jumlah maksimum eksekusi simultan di panggung.

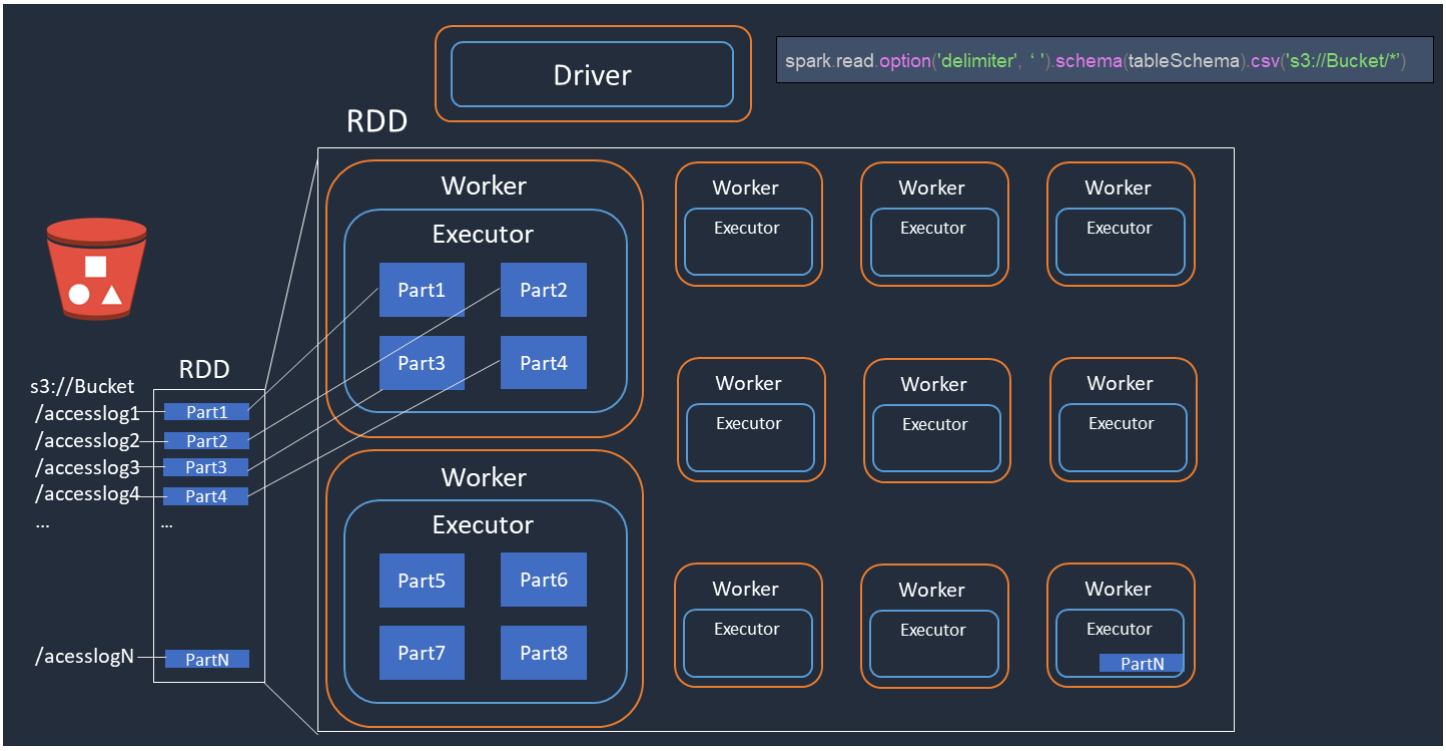


> **Note**
>
> Tugas adalah hal yang paling penting untuk dipertimbangkan ketika mengoptimalkan paralelisme. Jumlah skala tugas dengan jumlah RDD

Paralelisme

Spark memparalelkan tugas untuk memuat dan mengubah data.

Pertimbangkan contoh di mana Anda melakukan pemrosesan terdistribusi file log akses (bernama`accesslog1 ... accesslogN`) di Amazon S3. Diagram berikut menunjukkan aliran pemrosesan terdistribusi.

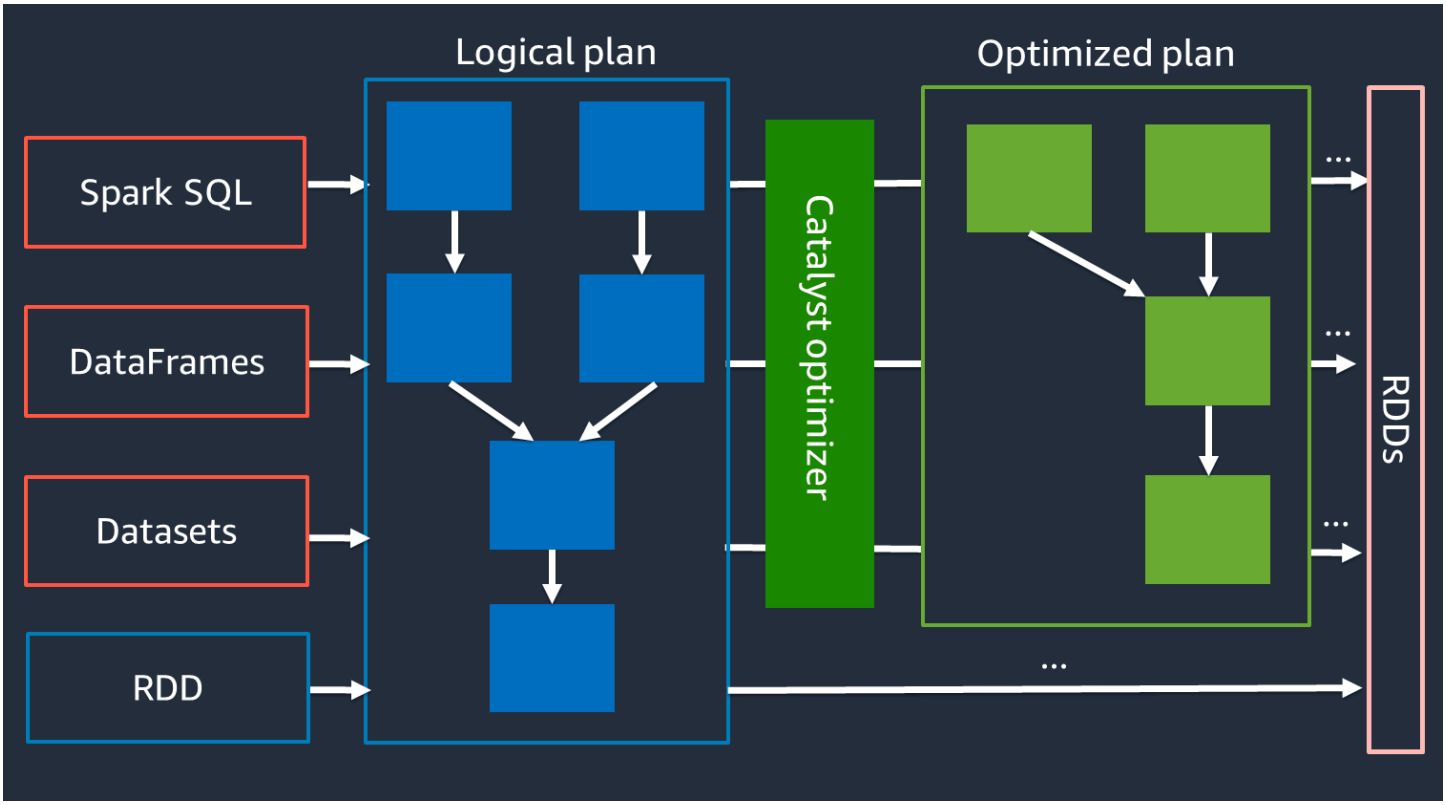


1. Driver Spark membuat rencana eksekusi untuk pemrosesan terdistribusi di banyak pelaksana Spark.
2. Driver Spark memberikan tugas setiap pelaksana berdasarkan rencana eksekusi. Secara default, driver Spark membuat partisi RDD (masing-masing sesuai dengan tugas Spark) untuk setiap objek S3 (). `Part1 ... N` Kemudian driver Spark memberikan tugas kepada setiap eksekutor.
3. Setiap tugas Spark mengunduh objek S3 yang ditetapkan dan menyimpannya dalam memori di partisi RDD. Dengan cara ini, beberapa pelaksana Spark mengunduh dan memproses tugas yang ditugaskan secara paralel.

Untuk detail selengkapnya tentang jumlah awal partisi dan pengoptimalan, lihat bagian Parallelize task.

Pengoptimal katalis

Secara internal, Spark menggunakan mesin yang disebut Catalyst Optimizer untuk mengoptimalkan rencana eksekusi. Catalyst memiliki pengoptimal kueri yang dapat Anda gunakan saat menjalankan API Spark tingkat tinggi, seperti Spark SQL,, dan DatasetsDataFrame, seperti yang dijelaskan dalam diagram berikut.



Karena pengoptimal Catalyst tidak bekerja secara langsung dengan API RDD, API tingkat tinggi umumnya lebih cepat daripada API RDD tingkat rendah. Untuk gabungan yang kompleks, pengoptimal Catalyst dapat secara signifikan meningkatkan kinerja dengan mengoptimalkan rencana menjalankan pekerjaan. Anda dapat melihat rencana pekerjaan Spark yang dioptimalkan di tab SQL UI Spark.

Eksekusi Kueri Adaptif

Pengoptimal Catalyst melakukan optimasi runtime melalui proses yang disebut Adaptive Query Execution. Eksekusi Kueri Adaptif menggunakan statistik runtime untuk mengoptimalkan kembali rencana run kueri saat pekerjaan Anda berjalan. Adaptive Query Execution menawarkan beberapa solusi untuk tantangan kinerja, termasuk menggabungkan partisi pasca-shuffle, mengonversi gabungan sort-merge menjadi broadcast join, dan pengoptimalan skew join, seperti yang dijelaskan di bagian berikut.

Eksekusi Kueri Adaptif tersedia di AWS Glue 3.0 dan yang lebih baru, dan diaktifkan secara default di AWS Glue 4.0 (Spark 3.3.0) dan yang lebih baru. Eksekusi Kueri Adaptif dapat dihidupkan dan dimatikan dengan menggunakan `spark.conf.set("spark.sql.adaptive.enabled", "true")` kode Anda.

Menggabungkan partisi pasca-pengocokan

Fitur ini mengurangi partisi RDD (coalesce) setelah setiap shuffle berdasarkan statistik output. `map` Ini menyederhanakan penyetelan nomor partisi shuffle saat menjalankan kueri. Anda tidak perlu mengatur nomor partisi acak agar sesuai dengan kumpulan data Anda. Spark dapat memilih nomor partisi shuffle yang tepat saat runtime setelah Anda memiliki jumlah partisi shuffle awal yang cukup besar.

Menggabungkan partisi pasca-shuffle diaktifkan ketika keduanya `spark.sql.adaptive.enabled` dan disetel ke true. `spark.sql.adaptive.coalescePartitions.enabled` Untuk informasi selengkapnya, lihat dokumentasi Apache Spark.

Mengonversi gabungan sort-merge menjadi broadcast join

Fitur ini mengenali ketika Anda menggabungkan dua kumpulan data dengan ukuran yang sangat berbeda, dan mengadopsi algoritma gabungan yang lebih efisien berdasarkan informasi tersebut. Untuk detail selengkapnya, lihat dokumentasi Apache Spark. Strategi bergabung dibahas di bagian Optimalkan shuffles.

Skew bergabung optimasi

Kemiringan data adalah salah satu kemacetan paling umum untuk pekerjaan Spark. Ini menggambarkan situasi di mana data condong ke partisi RDD tertentu (dan akibatnya, tugas tertentu), yang menunda waktu pemrosesan keseluruhan aplikasi. Ini sering dapat menurunkan kinerja operasi gabungan. Fitur pengoptimalan gabungan miring secara dinamis menangani gabungan miring dalam penggabungan sortir dengan membagi (dan mereplikasi jika diperlukan) tugas miring menjadi tugas yang kira-kira berukuran genap.

Fitur ini diaktifkan ketika `spark.sql.adaptive.skewJoin.enabled` disetel ke true. Untuk detail selengkapnya, lihat dokumentasi Apache Spark. Kemiringan data dibahas lebih lanjut di bagian Optimalkan shuffles.

Selidiki masalah kinerja dengan menggunakan UI Spark

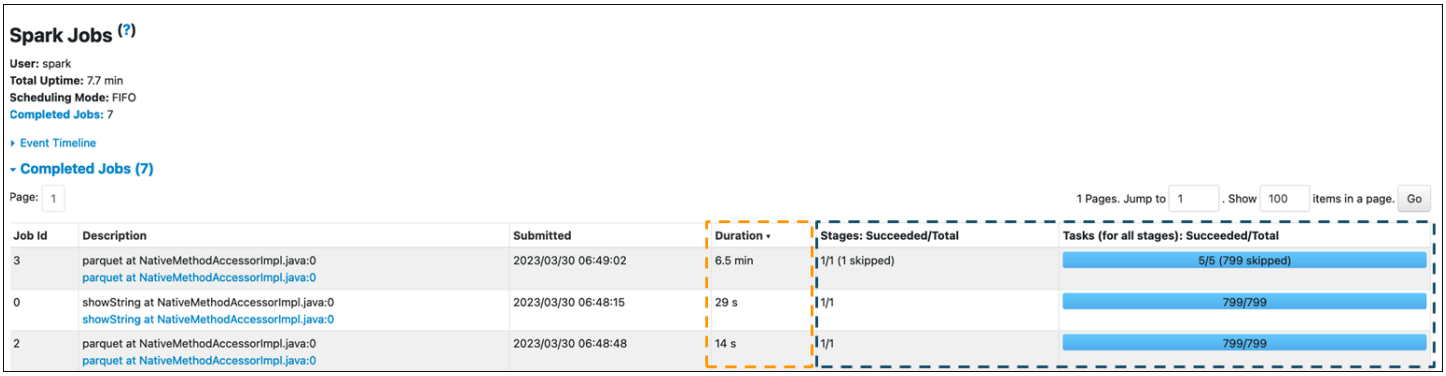
Sebelum Anda menerapkan praktik terbaik apa pun untuk menyesuaikan kinerja AWS Glue pekerjaan Anda, kami sangat menyarankan agar Anda membuat profil kinerja dan mengidentifikasi kemacetan. Ini akan membantu Anda fokus pada hal-hal yang benar.

Untuk analisis cepat, CloudWatch metrik Amazon memberikan tampilan dasar metrik pekerjaan Anda. Spark UI memberikan tampilan yang lebih dalam untuk penyetelan kinerja. Untuk menggunakan UI Spark dengan AWS Glue, Anda harus mengaktifkan Spark UI untuk pekerjaan Anda AWS Glue. Setelah Anda terbiasa dengan Spark UI, ikuti strategi untuk menyetel kinerja pekerjaan Spark untuk mengidentifikasi dan mengurangi dampak kemacetan berdasarkan temuan Anda.

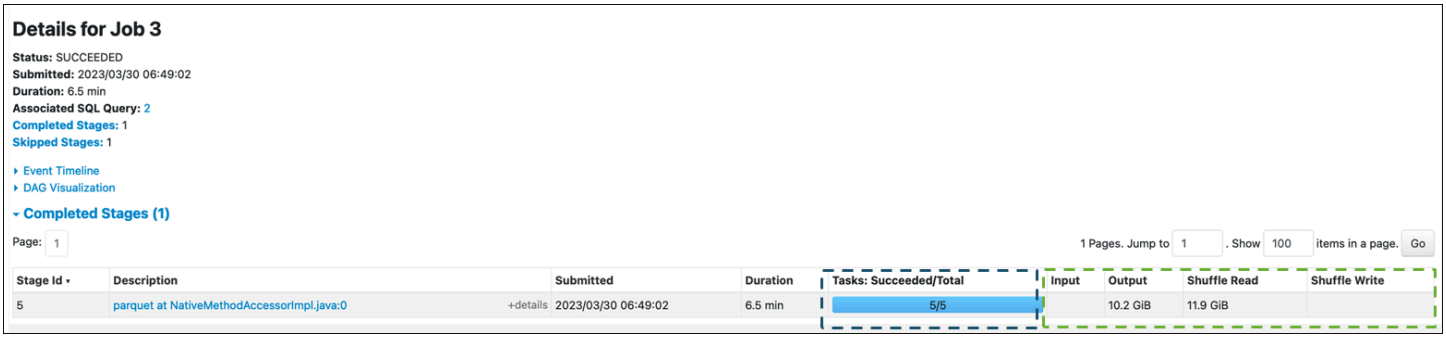
Identifikasi kemacetan dengan menggunakan UI Spark

Saat Anda membuka UI Spark, aplikasi Spark tercantum dalam tabel. Secara default, Nama Aplikasi AWS Glue pekerjaan adalah`nativespark-<Job Name>-<Job Run ID>`. Pilih aplikasi Spark target berdasarkan ID job run untuk membuka tab Jobs. Jalankan pekerjaan yang tidak lengkap, seperti streaming job run, tercantum di Tampilkan aplikasi yang tidak lengkap.

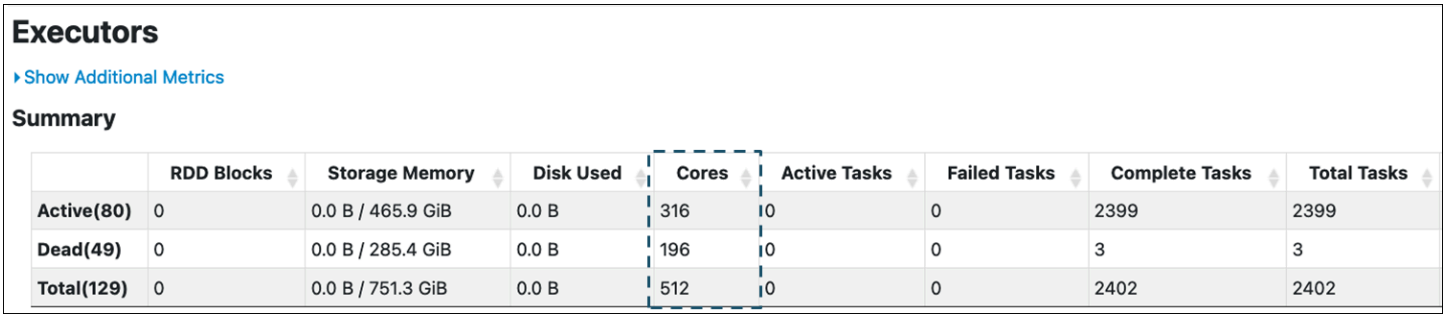
Tab Jobs menunjukkan ringkasan semua pekerjaan di aplikasi Spark. Untuk menentukan setiap tahap atau kegagalan tugas, periksa jumlah total tugas. Untuk menemukan kemacetan, urutkan dengan memilih Durasi. Telusuri detail pekerjaan yang sudah berjalan lama dengan memilih tautan yang ditampilkan di kolom Deskripsi.



Halaman Details for Job mencantumkan tahapan. Di halaman ini, Anda dapat melihat wawasan keseluruhan seperti durasi, jumlah tugas yang berhasil dan total, jumlah input dan output, dan jumlah shuffle read dan shuffle write.



Tab Executor menunjukkan kapasitas cluster Spark secara detail. Anda dapat memeriksa jumlah total core. Cluster yang ditunjukkan pada tangkapan layar berikut berisi 316 core aktif dan total 512 core. Secara default, setiap inti dapat memproses satu tugas Spark secara bersamaan.



	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks
Active(80)	0	0.0 B / 465.9 GiB	0.0 B	316	0	0	2399	2399
Dead(49)	0	0.0 B / 285.4 GiB	0.0 B	196	0	0	3	3
Total(129)	0	0.0 B / 751.3 GiB	0.0 B	512	0	0	2402	2402

Berdasarkan nilai yang 5/5 ditunjukkan pada halaman Details for Job, tahap 5 adalah tahap terpanjang, tetapi hanya menggunakan 5 core dari 512. Karena paralelisme untuk tahap ini sangat rendah, tetapi membutuhkan banyak waktu, Anda dapat mengidentifikasinya sebagai hambatan. Untuk meningkatkan kinerja, Anda ingin memahami alasannya. Untuk mempelajari lebih lanjut tentang cara mengenali dan mengurangi dampak kemacetan kinerja umum, lihat Strategi untuk menyetel kinerja pekerjaan Spark.

Strategi untuk menyetel kinerja pekerjaan Spark

Saat mempersiapkan untuk menyetel parameter, gunakan praktik terbaik berikut:

- Tentukan tujuan kinerja Anda sebelum mulai mengidentifikasi masalah.
- Gunakan metrik untuk mengidentifikasi masalah sebelum mencoba mengubah parameter penyetelan.

Untuk hasil yang paling konsisten saat menyetel pekerjaan, kembangkan strategi dasar untuk pekerjaan penyetelan Anda.

Strategi dasar untuk penyetelan kinerja

Umumnya, penyetelan kinerja dilakukan dalam alur kerja berikut:

1. Tentukan tujuan kinerja.
2. Ukur metrik.
3. Identifikasi kemacetan.
4. Kurangi dampak kemacetan.
5. Ulangi langkah 2-4 sampai Anda mencapai target yang diinginkan.

Pertama, tentukan tujuan kinerja Anda. Misalnya, salah satu tujuan Anda mungkin menyelesaikan AWS Glue pekerjaan dalam waktu 3 jam. Setelah Anda menentukan tujuan Anda, ukur metrik kinerja pekerjaan. Identifikasi tren dalam metrik dan kemacetan untuk memenuhi tujuan. Secara khusus, mengidentifikasi kemacetan paling penting untuk pemecahan masalah, debugging, dan penyetelan kinerja. Selama menjalankan aplikasi Spark, Spark mencatat status dan statistik setiap tugas di log peristiwa Spark.

Di AWS Glue, Anda dapat melihat metrik Spark melalui UI Web Spark yang disediakan oleh server riwayat Spark. AWS Glue untuk pekerjaan Spark dapat mengirim log peristiwa Spark ke lokasi yang Anda tentukan di Amazon S3. AWS Glue juga menyediakan contoh AWS CloudFormation template dan Dockerfile untuk memulai server riwayat Spark pada EC2 instance Amazon atau komputer lokal Anda, sehingga Anda dapat menggunakan UI Spark dengan log peristiwa.

Setelah Anda menentukan sasaran kinerja dan mengidentifikasi metrik untuk menilai tujuan tersebut, Anda dapat mulai mengidentifikasi dan memulihkan kemacetan dengan menggunakan strategi di bagian berikut.

Praktik tuning untuk kinerja pekerjaan Spark

Anda dapat menggunakan strategi berikut untuk penyetelan kinerja AWS Glue untuk pekerjaan Spark:

- AWS Glue sumber daya:
  - Kapasitas klaster skala
  - Gunakan AWS Glue versi terbaru
- Aplikasi percikan:
  - Kurangi jumlah pemindaian data
  - Paralelisasi tugas
  - Optimalkan shuffle
  - Minimalkan overhead perencanaan
  - Optimalkan fungsi yang ditentukan pengguna

Sebelum Anda menggunakan strategi ini, Anda harus memiliki akses ke metrik dan konfigurasi untuk pekerjaan Spark Anda. Anda dapat menemukan informasi ini di AWS Glue dokumentasi.

Dari perspektif AWS Glue sumber daya, Anda dapat mencapai peningkatan kinerja dengan menambahkan AWS Glue pekerja dan menggunakan AWS Glue versi terbaru.

Dari perspektif aplikasi Apache Spark, Anda memiliki akses ke beberapa strategi yang dapat meningkatkan kinerja. Jika data yang tidak perlu dimuat ke dalam cluster Spark, Anda dapat menghapusnya untuk mengurangi jumlah data yang dimuat. Jika Anda memiliki sumber daya cluster Spark yang kurang digunakan dan Anda memiliki I/O data yang rendah, Anda dapat mengidentifikasi tugas untuk diparalelkan. Anda mungkin juga ingin mengoptimalkan operasi transfer data berat seperti bergabung jika mereka membutuhkan waktu yang cukup lama. Anda juga dapat mengoptimalkan rencana kueri pekerjaan Anda atau mengurangi kompleksitas komputasi tugas Spark individual.

Untuk menerapkan strategi ini secara efisien, Anda harus mengidentifikasi kapan strategi tersebut berlaku dengan berkonsultasi dengan metrik Anda. Untuk detail selengkapnya, lihat masing-

masing bagian berikut. Teknik-teknik ini bekerja tidak hanya untuk tuning kinerja tetapi juga untuk memecahkan masalah khas seperti out-of-memory (OOM) kesalahan.

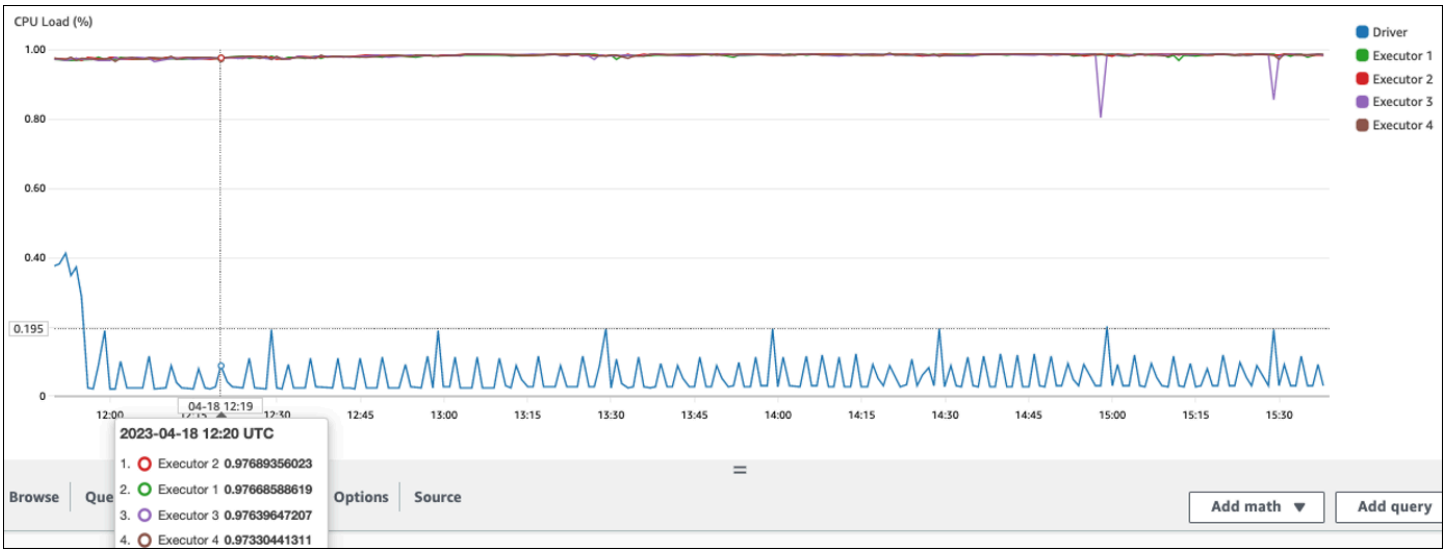
Kapasitas klaster skala

Jika pekerjaan Anda memakan terlalu banyak waktu, tetapi pelaksana menghabiskan sumber daya yang cukup dan Spark membuat sejumlah besar tugas relatif terhadap inti yang tersedia, pertimbangkan untuk menskalakan kapasitas cluster. Untuk menilai apakah ini sesuai, gunakan metrik berikut.

CloudWatch metrik

- Periksa Pemanfaatan CPUBeban dan Memori untuk menentukan apakah pelaksana mengkonsumsi sumber daya yang cukup.
- Periksa berapa lama pekerjaan telah berjalan untuk menilai apakah waktu pemrosesan terlalu lama untuk memenuhi tujuan kinerja Anda.

Dalam contoh berikut, empat pelaksana berjalan pada CPU beban lebih dari 97 persen, tetapi pemrosesan belum selesai setelah sekitar tiga jam.

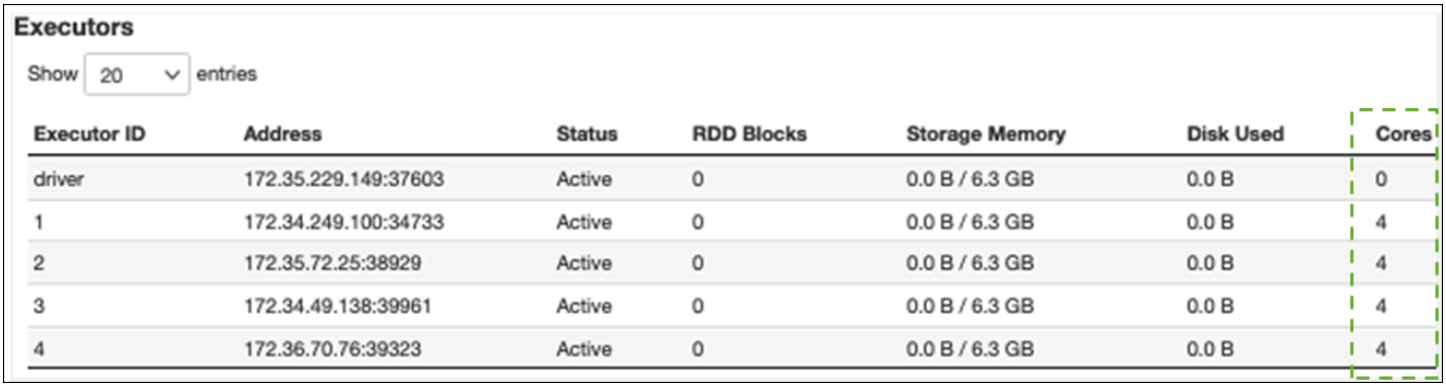


> **Note**
>
> Jika CPU beban rendah, Anda mungkin tidak akan mendapat manfaat dari penskalaan kapasitas cluster.

Spark UI

Pada tab Job atau tab Stage, Anda dapat melihat jumlah tugas untuk setiap pekerjaan atau tahap. Dalam contoh berikut, Spark telah membuat `58100` tugas.

Pada tab Executor, Anda dapat melihat jumlah total pelaksana dan tugas. Pada tangkapan layar berikut, setiap pelaksana Spark memiliki empat inti dan dapat melakukan empat tugas secara bersamaan.



Dalam contoh ini, jumlah tugas Spark (jauh `58100`) lebih besar daripada 16 tugas yang dapat diproses oleh pelaksana secara bersamaan (4 pelaksana × 4 core).

Jika Anda mengamati gejala-gejala ini, pertimbangkan untuk menskalakan cluster. Anda dapat menskalakan kapasitas cluster dengan menggunakan opsi berikut:

- Aktifkan AWS Glue Auto Scaling — Auto Scaling tersedia untuk mengekstrak, mengubah, dan memuat ETL () dan pekerjaan streaming AWS Glue Anda AWS Glue di versi 3.0 atau yang lebih baru. AWS Glue secara otomatis menambahkan dan menghapus pekerja dari cluster tergantung pada jumlah partisi pada setiap tahap atau tingkat di mana microbatch dihasilkan pada pekerjaan yang dijalankan.

  Jika Anda mengamati situasi di mana jumlah pekerja tidak bertambah meskipun Auto Scaling diaktifkan, pertimbangkan untuk menambahkan pekerja secara manual. Namun, perhatikan bahwa penskalaan secara manual untuk satu tahap dapat menyebabkan banyak pekerja menganggur selama tahap selanjutnya, dengan biaya lebih untuk nol perolehan kinerja.

Setelah mengaktifkan Auto Scaling, Anda dapat melihat jumlah pelaksana dalam metrik pelaksana. CloudWatch Gunakan metrik berikut untuk memantau permintaan pelaksana dalam aplikasi Spark:

- `glue.driver.ExecutorAllocationManager.executors.numberAllExecutors`
- `glue.driver.ExecutorAllocationManager.executors.numberMaxNeededExecutors`

Untuk informasi selengkapnya tentang metrik, lihat Memantau AWS Glue menggunakan CloudWatch metrik Amazon.

- Skalakan: Tingkatkan jumlah AWS Glue pekerja - Anda dapat meningkatkan jumlah AWS Glue pekerja secara manual. Tambahkan pekerja hanya sampai Anda mengamati pekerja yang menganggur. Pada saat itu, menambahkan lebih banyak pekerja akan meningkatkan biaya tanpa meningkatkan hasil. Untuk informasi selengkapnya, lihat Parallelize task.
- Tingkatkan: Gunakan tipe pekerja yang lebih besar — Anda dapat secara manual mengubah jenis instans AWS Glue pekerja Anda untuk menggunakan pekerja dengan lebih banyak inti, memori, dan penyimpanan. Jenis pekerja yang lebih besar memungkinkan Anda untuk menskalakan secara vertikal dan menjalankan pekerjaan integrasi data intensif, seperti transformasi data intensif memori, agregasi miring, dan pemeriksaan deteksi entitas yang melibatkan petabyte data.

  Peningkatan skala juga membantu dalam kasus di mana driver Spark membutuhkan kapasitas yang lebih besar — misalnya, karena rencana permintaan pekerjaan cukup besar. Untuk informasi selengkapnya tentang jenis dan kinerja pekerja, lihat postingan AWS Big Data Blog Scale your AWS Glue for Apache Spark jobs dengan tipe pekerja baru yang lebih besar G.4X dan G.8X.

  Menggunakan pekerja yang lebih besar juga dapat mengurangi jumlah total pekerja yang dibutuhkan, yang meningkatkan kinerja dengan mengurangi shuffle dalam operasi intensif seperti bergabung.

Gunakan AWS Glue versi terbaru

Kami merekomendasikan menggunakan AWS Glue versi terbaru. Ada beberapa pengoptimalan dan peningkatan yang dibangun ke dalam setiap versi yang mungkin secara otomatis meningkatkan kinerja pekerjaan. Misalnya, AWS Glue 4.0 menyediakan fitur-fitur baru berikut:

- Runtime Apache Spark 3.3.0 baru yang dioptimalkan — AWS Glue 4.0 dibangun di atas runtime Apache Spark 3.3.0, menghadirkan peningkatan kinerja yang sebanding dengan Spark open source. Runtime Spark 3.3.0 dibangun di atas banyak inovasi dari Spark 2.x.

- Konektor Amazon Redshift yang disempurnakan - versi AWS Glue 4.0 dan yang lebih baru menyediakan integrasi Amazon Redshift untuk Apache Spark. Integrasi dibangun di atas konektor open source yang ada dan meningkatkannya untuk kinerja dan keamanan. Integrasi ini membantu aplikasi bekerja hingga 10 kali lebih cepat. Untuk informasi lebih lanjut, lihat posting blog tentang integrasi Amazon Redshift dengan Apache Spark.
- SIMDeksekusi berbasis untuk pembacaan vektor dengan CSV dan JSON data — AWS Glue versi 3.0 dan versi yang lebih baru menambahkan pembaca yang dioptimalkan yang secara signifikan dapat mempercepat kinerja pekerjaan secara keseluruhan dibandingkan dengan pembaca berbasis baris. Untuk informasi selengkapnya tentang CSV data, lihat Mengoptimalkan kinerja baca dengan pembaca vektor SIMD CSV. Untuk informasi selengkapnya tentang JSON data, lihat Menggunakan SIMD JSON pembaca vektor dengan format kolom Apache Arrow.

Setiap AWS Glue versi akan menyertakan upgrade semacam ini, di antara banyak, termasuk konektor, driver dan pembaruan perpustakaan. Untuk informasi selengkapnya, lihat AWS Glue versi dan Memigrasi AWS Glue pekerjaan ke AWS Glue versi 4.0.

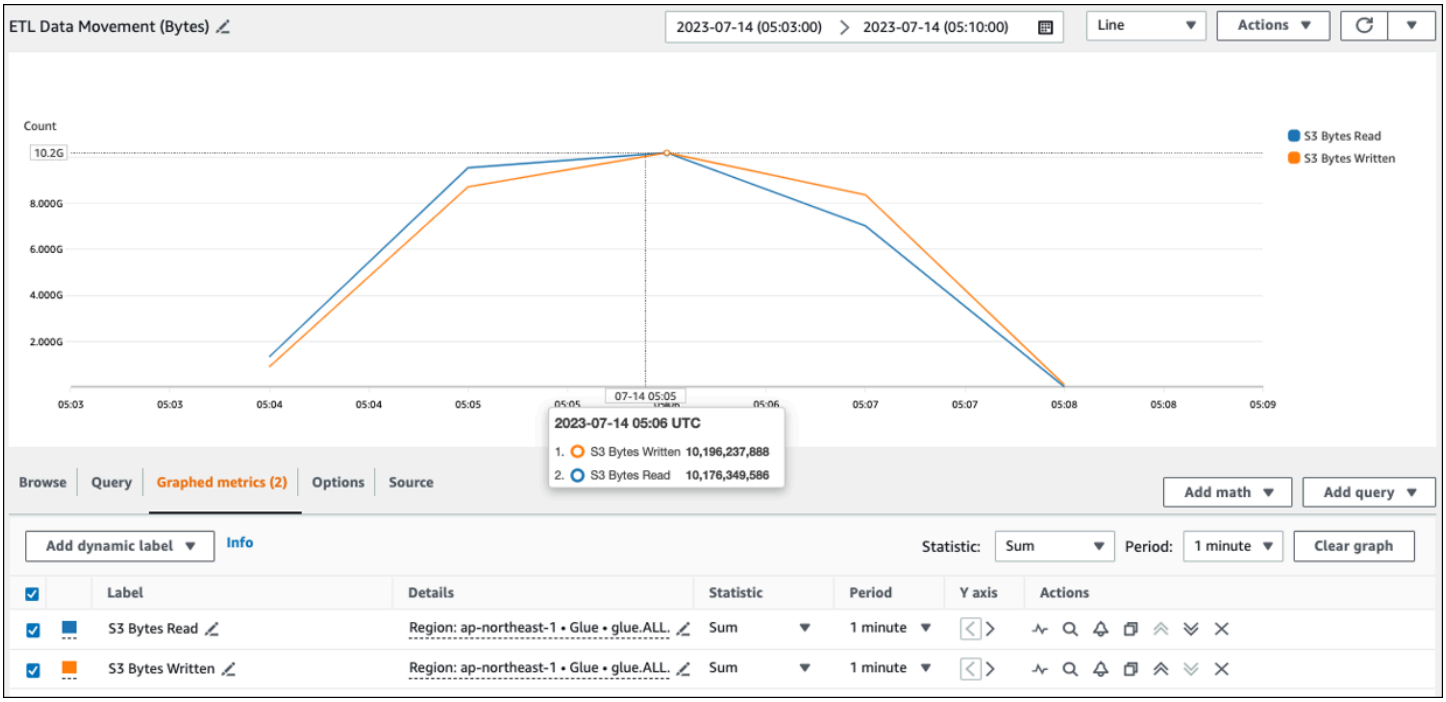
Kurangi jumlah pemindaian data

Untuk memulai, pertimbangkan untuk memuat hanya data yang Anda butuhkan. Anda dapat meningkatkan kinerja hanya dengan mengurangi jumlah data yang dimuat ke cluster Spark Anda untuk setiap sumber data. Untuk menilai apakah pendekatan ini tepat, gunakan metrik berikut.

Anda dapat memeriksa byte baca dari Amazon S3 CloudWatchdalam metrik dan detail selengkapnya di UI Spark seperti yang dijelaskan di bagian UI Spark.

CloudWatch metrik

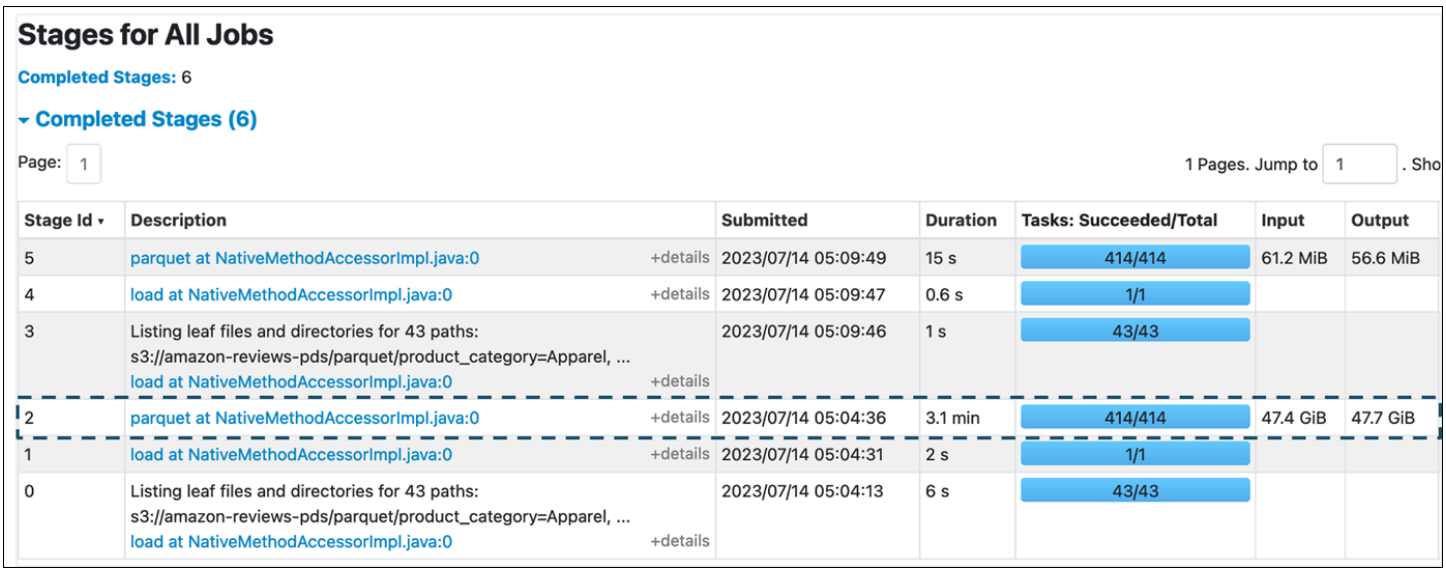
Anda dapat melihat perkiraan ukuran baca dari Amazon S3 di Gerakan ETL Data (Byte). Metrik ini menunjukkan jumlah byte yang dibaca dari Amazon S3 oleh semua pelaksana sejak laporan sebelumnya. Anda dapat menggunakannya untuk memantau pergerakan ETL data dari Amazon S3, dan Anda dapat membandingkan pembacaan dengan tingkat konsumsi dari sumber data eksternal.



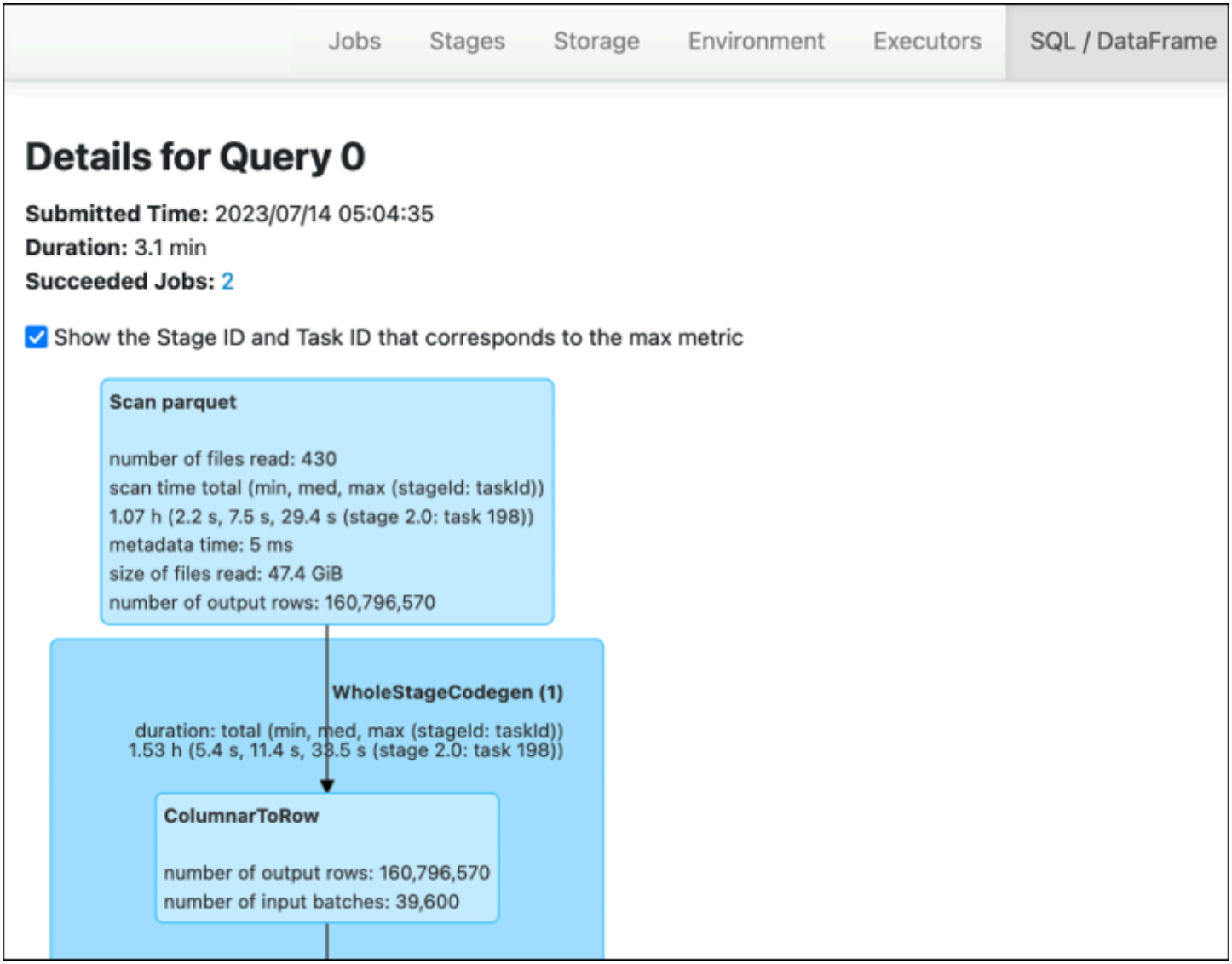
Jika Anda mengamati titik data S3 Bytes Read yang lebih besar dari yang Anda harapkan, pertimbangkan solusi berikut.

Spark UI

Pada tab Stage di AWS Glue for Spark UI, Anda dapat melihat ukuran Input dan Output. Pada contoh berikut, tahap 2 membaca input 47,4 GiB dan output 47,7 GiB, sedangkan tahap 5 membaca input 61,2 MiB dan output 56,6 MiB.



Saat Anda menggunakan Spark SQL atau DataFrame pendekatan dalam AWS Glue pekerjaan Anda, ataFrame tab SQL/D menunjukkan lebih banyak statistik tentang tahapan ini. Dalam hal ini, tahap 2 menunjukkan jumlah file yang dibaca: 430, ukuran file yang dibaca: 47.4 GiB, dan jumlah baris output: 160.796.570.



Jika Anda mengamati bahwa ada perbedaan besar dalam ukuran antara data yang Anda baca dan data yang Anda gunakan, cobalah solusi berikut.

Amazon S3

Untuk mengurangi jumlah data yang dimuat ke pekerjaan Anda saat membaca dari Amazon S3, pertimbangkan ukuran file, kompresi, format file, dan tata letak file (partisi) untuk kumpulan data

Anda. AWS Glue untuk pekerjaan Spark sering digunakan untuk ETL data mentah, tetapi untuk pemrosesan terdistribusi yang efisien, Anda perlu memeriksa fitur format sumber data Anda.

- Ukuran file - Kami merekomendasikan untuk menjaga ukuran file input dan output dalam kisaran moderat (misalnya, 128 MB). File yang terlalu kecil dan file yang terlalu besar dapat menyebabkan masalah.

  Sejumlah besar file kecil menyebabkan masalah berikut:

  - Beban I/O jaringan yang berat di Amazon S3 karena overhead yang diperlukan untuk membuat permintaan (`List`sepertiGet`,, `Head` atau) untuk banyak objek (dibandingkan dengan beberapa objek yang menyimpan jumlah data yang sama).
  - I/O berat dan beban pemrosesan pada driver Spark, yang akan menghasilkan banyak partisi dan tugas dan menyebabkan paralelisme yang berlebihan.

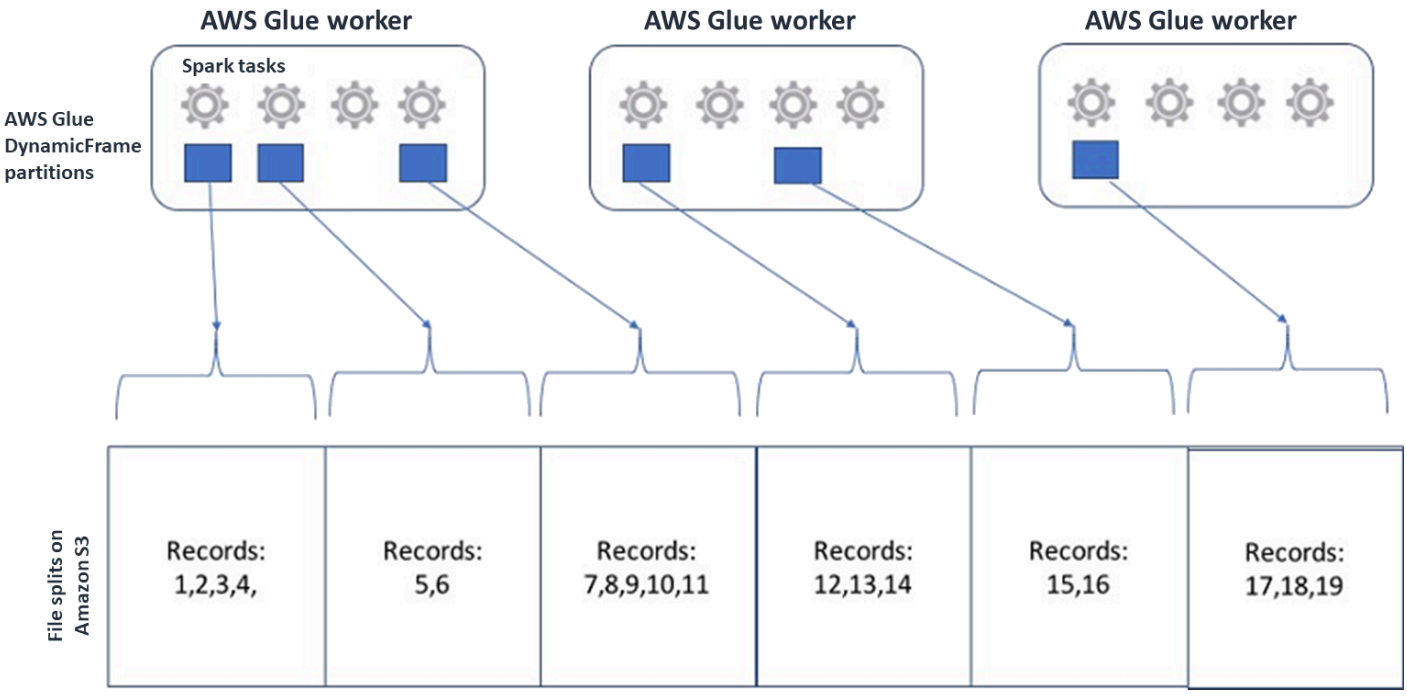
  Di sisi lain, jika jenis file Anda tidak dapat dibagi (seperti gzip) dan file terlalu besar, aplikasi Spark harus menunggu hingga satu tugas selesai membaca seluruh file.

  Untuk mengurangi paralelisme berlebihan yang terjadi ketika tugas Apache Spark dibuat untuk setiap file kecil, gunakan pengelompokan file untuk. DynamicFrames Pendekatan ini mengurangi kemungkinan OOM pengecualian dari driver Spark. Untuk mengkonfigurasi pengelompokan file, atur `groupSize` parameter `groupFiles` dan. Contoh kode berikut menggunakan AWS Glue DynamicFrame API dalam ETL skrip dengan parameter ini.

```
dyf = glueContext.create_dynamic_frame_from_options("s3",
    {'paths': ["s3://input-s3-path/"],
    'recurse':True,
    'groupFiles': 'inPartition',
    'groupSize': '1048576'},
    format="json")
```

- Kompresi — Jika objek S3 Anda berada dalam ratusan megabyte, pertimbangkan untuk mengompresnya. Ada berbagai format kompresi, yang secara luas dapat diklasifikasikan menjadi dua jenis:
  - Format kompresi yang tidak dapat dipisahkan seperti gzip mengharuskan seluruh file didekompresi oleh satu pekerja.
  - Format kompresi yang dapat dipisahkan, seperti bzip2 atau LZO (diindeks), memungkinkan dekompresi sebagian file, yang dapat diparalelkan.

Untuk Spark (dan mesin pemrosesan terdistribusi umum lainnya), Anda akan membagi file data sumber Anda menjadi potongan-potongan yang dapat diproses mesin Anda secara paralel. Unit-unit ini sering disebut sebagai split. Setelah data Anda dalam format yang dapat dibagi, AWS Glue pembaca yang dioptimalkan dapat mengambil split dari objek S3 dengan memberikan `Range` opsi untuk mengambil hanya blok tertentu `GetObject`API. Pertimbangkan diagram berikut untuk melihat bagaimana ini akan bekerja dalam praktik.



Data terkompresi dapat mempercepat aplikasi Anda secara signifikan, selama file berukuran optimal atau file dapat dibagi. Ukuran data yang lebih kecil mengurangi data yang dipindai dari Amazon S3 dan lalu lintas jaringan dari Amazon S3 ke cluster Spark Anda. Di sisi lain, lebih banyak CPU diperlukan untuk mengompres dan mendekompresi data. Jumlah skala komputasi yang diperlukan dengan rasio kompresi algoritma kompresi Anda. Pertimbangkan trade-off ini saat memilih format kompresi yang dapat dibagi.

> **Note**
>
> Meskipun file gzip umumnya tidak dapat dibagi, Anda dapat mengompres blok parket individual dengan gzip, dan blok tersebut dapat diparalelkan.

- Format file - Gunakan format kolumnar. Apache Parquet dan Apache ORC adalah format data kolumnar yang populer. Parket dan ORC menyimpan data secara efisien dengan menggunakan kompresi berbasis kolom, pengkodean dan kompresi setiap kolom berdasarkan tipe datanya. Untuk informasi lebih lanjut tentang pengkodean Parket, lihat Definisi pengkodean parket. File parket juga dapat dibagi.

  Format kolumnar mengelompokkan nilai berdasarkan kolom dan menyimpannya bersama dalam blok. Saat menggunakan format kolumnar, Anda dapat melewati blok data yang sesuai dengan kolom yang tidak Anda rencanakan untuk digunakan. Aplikasi Spark hanya dapat mengambil kolom yang Anda butuhkan. Umumnya, rasio kompresi yang lebih baik atau melewatkan blok data berarti membaca lebih sedikit byte dari Amazon S3, yang mengarah ke kinerja yang lebih baik. Kedua format juga mendukung pendekatan pushdown berikut untuk mengurangi I/O:

  - Proyeksi pushdown - Proyeksi pushdown adalah teknik untuk mengambil hanya kolom yang ditentukan dalam aplikasi Anda. Anda menentukan kolom dalam aplikasi Spark Anda, seperti yang ditunjukkan dalam contoh berikut:
    - DataFrame contoh: `df.select("star_rating")`
    - SQLContoh percikan: `spark.sql("select start_rating from <table>")`
  - Predikat pushdown — Predikat pushdown adalah teknik untuk memproses dan klausa secara efisien. `WHERE GROUP BY` Kedua format memiliki blok data yang mewakili nilai kolom. Setiap blok menyimpan statistik untuk blok, seperti nilai maksimum dan minimum. Spark dapat menggunakan statistik ini untuk menentukan apakah blok harus dibaca atau dilewati tergantung pada nilai filter yang digunakan dalam aplikasi. Untuk menggunakan fitur ini, tambahkan lebih banyak filter dalam kondisi, seperti yang ditunjukkan dalam contoh berikut sebagai berikut:
    - DataFrame contoh: `df.select("star_rating").filter("star_rating < 2")`
    - SQLContoh percikan: `spark.sql("select * from <table> where star_rating < 2")`
- Tata letak file - Dengan menyimpan data S3 Anda ke objek di jalur yang berbeda berdasarkan bagaimana data akan digunakan, Anda dapat secara efisien mengambil data yang relevan. Untuk informasi selengkapnya, lihat Mengatur objek menggunakan awalan dalam dokumentasi Amazon S3. AWS Glue mendukung penyimpanan kunci dan nilai ke awalan Amazon S3 dalam format`key=value`, mempartisi data Anda dengan jalur Amazon S3. Dengan mempartisi data Anda, Anda dapat membatasi jumlah data yang dipindai oleh setiap aplikasi analitik hilir, meningkatkan kinerja dan mengurangi biaya. Untuk informasi selengkapnya, lihat Mengelola partisi untuk ETL output di AWS Glue.

Partisi membagi tabel Anda menjadi beberapa bagian dan menyimpan data terkait dalam file yang dikelompokkan berdasarkan nilai kolom seperti tahun, bulan, dan hari, seperti yang ditunjukkan pada contoh berikut.

```
# Partitioning by /YYYY/MM/DD
s3://<YourBucket>/year=2023/month=03/day=31/0000.gz
s3://<YourBucket>/year=2023/month=03/day=01/0000.gz
s3://<YourBucket>/year=2023/month=03/day=02/0000.gz
s3://<YourBucket>/year=2023/month=03/day=03/0000.gz
...
```

Anda dapat menentukan partisi untuk kumpulan data Anda dengan memodelkannya dengan tabel di. AWS Glue Data Catalog Anda kemudian dapat membatasi jumlah pemindaian data dengan menggunakan pemangkasan partisi sebagai berikut:

- Untuk AWS Glue DynamicFrame, atur `push_down_predicate` (atau`catalogPartitionPredicate`).

  ```
  dyf = Glue_context.create_dynamic_frame.from_catalog(
      database=src_database_name,
      table_name=src_table_name,
      push_down_predicate = "year='2023' and month ='03'",
  )
  ```

- Untuk Spark DataFrame, atur jalur tetap untuk memangkas partisi.

  ```
  df = spark.read.format("json").load("s3://<YourBucket>/year=2023/month=03/*/*.gz")
  ```

- Untuk SparkSQL, Anda dapat mengatur klausa where untuk memangkas partisi dari Katalog Data.

  ```
  df = spark.sql("SELECT * FROM <Table> WHERE year= '2023' and month = '03'")
  ```

- Untuk mempartisi berdasarkan tanggal saat menulis data Anda AWS Glue, Anda mengatur partitionKeys DynamicFrame atau partitionBy() DataFrame dengan informasi tanggal di kolom Anda sebagai berikut.

  - DynamicFrame

    ```
    glue_context.write_dynamic_frame_from_options(
    ```

```
    frame= dyf, connection_type='s3',format='parquet'
    connection_options= {
        'partitionKeys': ["year", "month", "day"],
        'path': 's3://<YourBucket>/<Prefix>/'
    }
)
```

- DataFrame

```
df.write.mode('append')\
    .partitionBy('year','month','day')\
    .parquet('s3://<YourBucket>/<Prefix>/')
```

Hal ini dapat meningkatkan kinerja konsumen dari data output Anda.

Jika Anda tidak memiliki akses untuk mengubah pipeline yang membuat kumpulan data input Anda, partisi bukanlah pilihan. Sebagai gantinya, Anda dapat mengecualikan jalur S3 yang tidak dibutuhkan dengan menggunakan pola glob. Tetapkan pengecualian saat membaca. DynamicFrame Misalnya, kode berikut tidak termasuk hari dalam bulan 01 hingga 09, pada tahun 2023.

```
dyf = glueContext.create_dynamic_frame.from_catalog(
    database=db,
    table_name=table,
    additional_options = { "exclusions":"[\"**year=2023/month=0[1-9]/**\"]" },
    transformation_ctx='dyf'
)
```

Anda juga dapat mengatur pengecualian dalam properti tabel di Katalog Data:

- Kunci: `exclusions`
- Nilai: `["**year=2023/month=0[1-9]/**"]`

- Terlalu banyak partisi Amazon S3 - Hindari mempartisi data Amazon S3 Anda pada kolom yang berisi berbagai nilai, seperti kolom ID dengan ribuan nilai. Ini secara substansional dapat meningkatkan jumlah partisi dalam bucket Anda, karena jumlah partisi yang mungkin adalah produk dari semua bidang yang telah Anda partisi. Terlalu banyak partisi dapat menyebabkan hal berikut:
  - Peningkatan latensi untuk mengambil metadata partisi dari Katalog Data

- Peningkatan jumlah file kecil, yang membutuhkan lebih banyak API permintaan Amazon S3 (`List`,`Get`, dan) `Head`

Misalnya, ketika Anda menetapkan tipe tanggal di `partitionBy` atau`partitionKeys`, partisi tingkat tanggal seperti baik untuk banyak `yyyy/mm/dd` kasus penggunaan. Namun, `yyyy/mm/dd/<ID>` mungkin menghasilkan begitu banyak partisi sehingga akan berdampak negatif pada kinerja secara keseluruhan.

Di sisi lain, beberapa kasus penggunaan, seperti aplikasi pemrosesan waktu nyata, memerlukan banyak partisi sepertiyyyy/mm/dd/hh. Jika kasus penggunaan Anda memerlukan partisi besar, pertimbangkan untuk menggunakan indeks AWS Glue partisi untuk mengurangi latensi untuk mengambil metadata partisi dari Katalog Data.

Database dan JDBC

Untuk mengurangi pemindaian data saat mengambil informasi dari database, Anda dapat menentukan `where` predikat (atau klausa) dalam kueri. SQL Database yang tidak menyediakan SQL antarmuka akan menyediakan mekanisme mereka sendiri untuk query atau penyaringan.

Saat menggunakan koneksi Java Database Connectivity (JDBC), berikan kueri pilih dengan `where` klausa untuk parameter berikut:

- Untuk DynamicFrame, gunakan sampleQueryopsi. Saat menggunakan`create_dynamic_frame.from_catalog`, konfigurasikan `additional_options` argumen sebagai berikut.

```
query = "SELECT * FROM <TableName> where id = 'XX' AND"
datasource0 = glueContext.create_dynamic_frame.from_catalog(
  database = db,
  table_name = table,
  additional_options={
      "sampleQuery": query,
      "hashexpression": key,
      "hashpartitions": 10,
      "enablePartitioningForSampleQuery": True
    },
  transformation_ctx = "datasource0"
)
```

Kapanusing `create_dynamic_frame.from_options`, konfigurasikan `connection_options` argumen sebagai berikut.

```
query = "SELECT * FROM <TableName> where id = 'XX' AND"
datasource0 = glueContext.create_dynamic_frame.from_options(
    connection_type = connection,
    connection_options={
        "url": url,
        "user": user,
        "password": password,
        "dbtable": table,
        "sampleQuery": query,
        "hashexpression": key,
        "hashpartitions": 10,
        "enablePartitioningForSampleQuery": True
    }
)
```

- Untuk DataFrame, gunakan opsi kueri.

```
query = "SELECT * FROM <TableName> where id = 'XX'"
jdbcDF = spark.read \
.format('jdbc') \
.option('url', url) \
.option('user', user) \
.option('password', pwd) \
.option('query', query) \
.load()
```

- Untuk Amazon Redshift, gunakan AWS Glue 4.0 atau yang lebih baru untuk memanfaatkan dukungan pushdown di konektor Amazon Redshift Spark.

```
dyf = glueContext.create_dynamic_frame.from_catalog(
    database = "redshift-dc-database-name",
    table_name = "redshift-table-name",
    redshift_tmp_dir = args["temp-s3-dir"],
    additional_options = {"aws_iam_role": "arn:aws:iam::role-account-id:role/rs-role-name"}
)
```

- Untuk database lain, lihat dokumentasi untuk database tersebut.

AWS Glue pilihan

- Untuk menghindari pemindaian penuh untuk semua pekerjaan yang berjalan terus menerus, dan hanya memproses data yang tidak ada selama pekerjaan terakhir dijalankan, aktifkan bookmark pekerjaan.
- Untuk membatasi jumlah data input yang akan diproses, aktifkan eksekusi terbatas dengan bookmark pekerjaan. Ini membantu mengurangi jumlah data yang dipindai untuk setiap pekerjaan yang dijalankan.

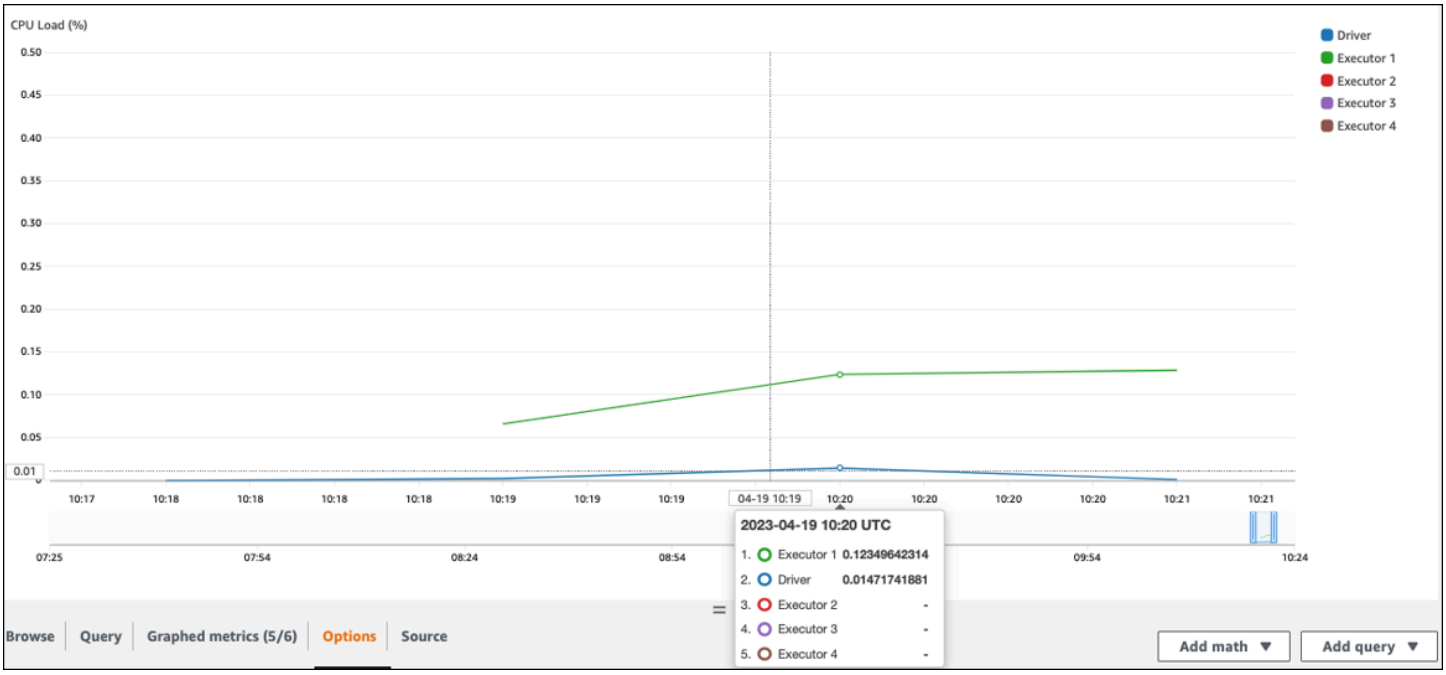
Paralelisasi tugas

Untuk mengoptimalkan kinerja, penting untuk memparalelkan tugas untuk pemuatan dan transformasi data. Seperti yang kita bahas di Topik utama di Apache Spark, jumlah partisi dataset terdistribusi yang tangguh (RDD) penting, karena menentukan tingkat paralelisme. Setiap tugas yang dibuat Spark sesuai dengan RDD partisi berdasarkan 1:1. Untuk mencapai kinerja terbaik, Anda perlu memahami bagaimana jumlah RDD partisi ditentukan dan bagaimana angka itu dioptimalkan.

Jika Anda tidak memiliki cukup paralelisme, gejala berikut akan dicatat dalam CloudWatchmetrik dan UI Spark.

CloudWatch metrik

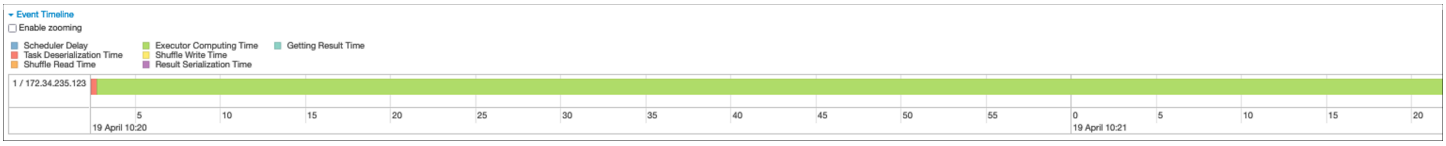
Periksa CPUPemanfaatan Beban dan Memori. Jika beberapa pelaksana tidak memproses selama fase pekerjaan Anda, itu tepat untuk meningkatkan paralelisme. Dalam hal ini, selama jangka waktu yang divisualisasikan, Pelaksana 1 melakukan tugas, tetapi pelaksana yang tersisa (2, 3, dan 4) tidak. Anda dapat menyimpulkan bahwa pelaksana tersebut tidak diberi tugas oleh driver Spark.



Spark UI

Pada tab Stage di UI Spark, Anda dapat melihat jumlah tugas dalam satu tahap. Dalam hal ini, Spark hanya melakukan satu tugas.

Selain itu, timeline acara menunjukkan Executor 1 memproses satu tugas. Ini berarti bahwa pekerjaan pada tahap ini dilakukan sepenuhnya pada satu eksekutor, sementara yang lain menganggur.



Jika Anda mengamati gejala-gejala ini, cobalah solusi berikut untuk setiap sumber data.

Paralelisasi beban data dari Amazon S3

Untuk memparalelkan beban data dari Amazon S3, periksa dulu jumlah partisi default. Anda kemudian dapat secara manual menentukan jumlah target partisi, tetapi pastikan untuk menghindari terlalu banyak partisi.

Tentukan jumlah partisi default

Untuk Amazon S3, jumlah awal RDD partisi Spark (masing-masing sesuai dengan tugas Spark) ditentukan oleh fitur kumpulan data Amazon S3 Anda (misalnya, format, kompresi, dan ukuran). Saat Anda membuat AWS Glue DynamicFrame atau Spark DataFrame dari CSV objek yang disimpan di Amazon S3, jumlah awal RDD partisi `NumPartitions` () dapat dihitung kira-kira sebagai berikut:

- Ukuran objek <= 64 MB: `NumPartitions = Number of Objects`
- Ukuran objek > 64 MB: `NumPartitions = Total Object Size / 64 MB`
- Tidak dapat dipisahkan (gzip): `NumPartitions = Number of Objects`

Seperti yang dibahas di bagian Kurangi jumlah pemindaian data, Spark membagi objek S3 besar menjadi split yang dapat diproses secara paralel. Ketika objek lebih besar dari ukuran split, Spark membagi objek dan membuat RDD partisi (dan tugas) untuk setiap split. Ukuran split Spark didasarkan pada format data dan lingkungan runtime Anda, tetapi ini adalah perkiraan awal yang masuk akal. Beberapa objek dikompresi menggunakan format kompresi yang tidak dapat dipisahkan seperti gzip, sehingga Spark tidak dapat membaginya.

`NumPartitions`Nilai dapat bervariasi tergantung pada format data Anda, kompresi, AWS Glue versi, jumlah AWS Glue pekerja, dan konfigurasi Spark.

Misalnya, ketika Anda memuat satu `csv.gz` objek 10 GB menggunakan Spark DataFrame, driver Spark hanya akan membuat satu RDD Partition (`NumPartitions=1`) karena gzip tidak dapat dipisahkan. Ini menghasilkan beban berat pada satu pelaksana Spark tertentu dan tidak ada tugas yang ditugaskan ke pelaksana yang tersisa, seperti yang dijelaskan pada gambar berikut.

Periksa jumlah tugas (`NumPartitions`) aktual untuk tahap pada tab Spark Web UI Stage, atau jalankan `df.rdd.getNumPartitions()` kode Anda untuk memeriksa paralelisme.

Saat menemukan file gzip 10 GB, periksa apakah sistem yang menghasilkan file itu dapat menghasilkannya dalam format yang dapat dibagi. Jika ini bukan pilihan, Anda mungkin perlu menskalakan kapasitas cluster untuk memproses file. Untuk menjalankan transformasi secara efisien pada data yang Anda muat, Anda perlu menyeimbangkan kembali RDD seluruh pekerja di klaster Anda dengan menggunakan partisi ulang.

Secara manual menentukan jumlah target partisi

Bergantung pada properti data Anda dan implementasi Spark dari fungsionalitas tertentu, Anda mungkin berakhir dengan `NumPartitions` nilai rendah meskipun pekerjaan yang mendasarinya

masih dapat diparalelkan. Jika `NumPartitions` terlalu kecil, jalankan `df.repartition(N)` untuk menambah jumlah partisi sehingga pemrosesan dapat didistribusikan di beberapa pelaksana Spark.

Dalam hal ini, berjalan `df.repartition(100)` akan meningkat `NumPartitions` dari 1 menjadi 100, membuat 100 partisi data Anda, masing-masing dengan tugas yang dapat ditugaskan ke pelaksana lainnya.

Operasi `repartition(N)` membagi seluruh data secara merata (10 GB/100 partisi = 100 MB/partisi), menghindari kemiringan data ke partisi tertentu.

> **Note**
>
> Ketika operasi shuffle seperti `join` dijalankan, jumlah partisi secara dinamis meningkat atau menurun tergantung pada nilai atau. `spark.sql.shuffle.partitions` `spark.default.parallelism` Ini memfasilitasi pertukaran data yang lebih efisien antara pelaksana Spark. Untuk informasi selengkapnya, lihat dokumentasi Spark.

Tujuan Anda saat menentukan jumlah target partisi adalah untuk memaksimalkan penggunaan pekerja yang disediakan AWS Glue . Jumlah AWS Glue pekerja dan jumlah tugas Spark terkait melalui jumlah. vCPUs Spark mendukung satu tugas untuk setiap v CPU core. Dalam AWS Glue versi 3.0 atau yang lebih baru, Anda dapat menghitung jumlah target partisi dengan menggunakan rumus berikut.

```
# Calculate NumPartitions by WorkerType
numExecutors = (NumberOfWorkers - 1)
numSlotsPerExecutor =
  4 if WorkerType is G.1X
  8 if WorkerType is G.2X
  16 if WorkerType is G.4X
  32 if WorkerType is G.8X
NumPartitions = numSlotsPerExecutor * numExecutors

# Example: Glue 4.0 / G.1X / 10 Workers
numExecutors = ( 10 - 1 ) = 9  # 1 Worker reserved on Spark Driver
numSlotsPerExecutor       = 4  # G.1X has 4 vCpu core ( Glue 3.0 or later )
NumPartitions = 9  * 4    = 36
```

Dalam contoh ini, setiap pekerja G.1X menyediakan empat CPU inti v ke eksekutor Spark (). `spark.executor.cores = 4` Spark mendukung satu tugas untuk setiap v CPU Core,

sehingga pelaksana G.1X Spark dapat menjalankan empat tugas secara bersamaan (). `numSlotPerExecutor` Jumlah partisi ini memanfaatkan sepenuhnya cluster jika tugas membutuhkan waktu yang sama. Namun, beberapa tugas akan memakan waktu lebih lama dari yang lain, membuat inti idle. Jika ini terjadi, pertimbangkan untuk mengalikan `numPartitions` dengan 2 atau 3 untuk memecah dan menjadwalkan tugas bottleneck secara efisien.

Terlalu banyak partisi

Jumlah partisi yang berlebihan menciptakan jumlah tugas yang berlebihan. Hal ini menyebabkan beban berat pada driver Spark karena overhead yang terkait dengan pemrosesan terdistribusi, seperti tugas manajemen dan pertukaran data antara pelaksana Spark.

Jika jumlah partisi dalam pekerjaan Anda jauh lebih besar dari jumlah partisi target Anda, pertimbangkan untuk mengurangi jumlah partisi. Anda dapat mengurangi partisi dengan menggunakan opsi berikut:

- Jika ukuran file Anda sangat kecil, gunakan AWS Glue groupFiles. Anda dapat mengurangi paralelisme berlebihan yang dihasilkan dari peluncuran tugas Apache Spark untuk memproses setiap file.
- Gunakan `coalesce(N)` untuk menggabungkan partisi bersama-sama. Ini adalah proses berbiaya rendah. Ketika mengurangi jumlah partisi, lebih disukai daripada`repartition(N)`, `coalesce(N)` karena `repartition(N)` melakukan shuffle untuk mendistribusikan jumlah catatan di setiap partisi secara merata. Itu meningkatkan biaya dan overhead manajemen.
- Gunakan Eksekusi Kueri Adaptif Spark 3.x. Seperti yang dibahas dalam Topik utama di bagian Apache Spark, Adaptive Query Execution menyediakan fungsi untuk secara otomatis menggabungkan jumlah partisi. Anda dapat menggunakan pendekatan ini ketika Anda tidak dapat mengetahui jumlah partisi sampai Anda melakukan eksekusi.

Paralelisasi beban data dari JDBC

Jumlah RDD partisi Spark ditentukan oleh konfigurasi. Perhatikan bahwa secara default hanya satu tugas yang dijalankan untuk memindai seluruh kumpulan data sumber melalui `SELECT` kueri.

Keduanya AWS Glue DynamicFrames dan Spark DataFrames mendukung pemuatan JDBC data paralel di beberapa tugas. Hal ini dilakukan dengan menggunakan `where` predikat untuk membagi satu `SELECT` query menjadi beberapa query. Untuk memparalelkan pembacaan dariJDBC, konfigurasikan opsi berikut:

- Untuk AWS Glue DynamicFrame, atur `hashfield` (atau `hashexpression`) dan`hashpartition`. Untuk mempelajari lebih lanjut, lihat Membaca dari JDBC tabel secara paralel.

```
connection_mysql8_options = {
    "url": "jdbc:mysql://XXXXXXXXXX.XXXXXXX.us-east-1.rds.amazonaws.com:3306/test",
    "dbtable": "medicare_tb",
    "user": "test",
    "password": "XXXXXXXXX",
    "hashexpression":"id",
    "hashpartitions":"10"
}
datasource0 = glueContext.create_dynamic_frame.from_options(
    'mysql',
    connection_options=connection_mysql8_options,
    transformation_ctx= "datasource0"
)
```

- Untuk Spark DataFrame, set`numPartitions`,, `partitionColumn``lowerBound`, dan`upperBound`. Untuk mempelajari lebih lanjut, lihat JDBCKe Database Lain.

```
df = spark.read \
    .format("jdbc") \
    .option("url", "jdbc:mysql://XXXXXXXXXX.XXXXXXX.us-east-1.rds.amazonaws.com:3306/
test") \
    .option("dbtable", "medicare_tb") \
    .option("user", "test") \
    .option("password", "XXXXXXXXXX") \
    .option("partitionColumn", "id") \
    .option("numPartitions", "10") \
    .option("lowerBound", "0") \
    .option("upperBound", "1141455") \
    .load()

df.write.format("json").save("s3://bucket_name/Tests/sparkjdbc/with_parallel/")
```

Paralelisasi beban data dari DynamoDB saat menggunakan konektor ETL

Jumlah RDD partisi Spark ditentukan oleh parameter. `dynamodb.splits` Untuk memparalelkan pembacaan dari Amazon DynamoDB, konfigurasikan opsi berikut:

- Tingkatkan nilai`dynamodb.splits`.
- Optimalkan parameter dengan mengikuti rumus yang dijelaskan dalam Jenis koneksi dan opsi untuk ETL in AWS Glue for Spark.

Paralelisasi beban data dari Kinesis Data Streams

Jumlah RDD partisi Spark ditentukan oleh jumlah pecahan di sumber aliran data Amazon Kinesis Data Streams. Jika Anda hanya memiliki beberapa pecahan dalam aliran data Anda, hanya akan ada beberapa tugas Spark. Hal ini dapat mengakibatkan paralelisme rendah dalam proses hilir. Untuk memparalelkan pembacaan dari Kinesis Data Streams, konfigurasikan opsi berikut:

- Tingkatkan jumlah pecahan untuk mendapatkan lebih banyak paralelisme saat memuat data dari Kinesis Data Streams.
- Jika logika Anda dalam batch mikro cukup kompleks, pertimbangkan untuk mempartisi ulang data di awal batch, setelah menjatuhkan kolom yang tidak dibutuhkan.

Untuk informasi selengkapnya, lihat Praktik terbaik untuk mengoptimalkan biaya dan kinerja untuk ETL pekerjaan AWS Glue streaming.

Paralelisasi tugas setelah pemuatan data

Untuk memparalelkan tugas setelah pemuatan data, tingkatkan jumlah RDD partisi dengan menggunakan opsi berikut:

- Partisi ulang data untuk menghasilkan jumlah partisi yang lebih besar, terutama tepat setelah pemuatan awal jika beban itu sendiri tidak dapat diparalelkan.

  Panggil aktif `repartition()` DynamicFrame atau DataFrame, tentukan jumlah partisi. Aturan praktis yang baik adalah dua atau tiga kali jumlah core yang tersedia.

  Namun, saat menulis tabel yang dipartisi, ini dapat menyebabkan ledakan file (setiap partisi berpotensi menghasilkan file ke setiap partisi tabel). Untuk menghindari hal ini, Anda dapat mempartisi ulang DataFrame dengan kolom Anda. Ini menggunakan kolom partisi tabel sehingga data diatur sebelum menulis. Anda dapat menentukan jumlah partisi yang lebih tinggi tanpa mendapatkan file kecil di partisi tabel. Namun, berhati-hatilah untuk menghindari kemiringan data, di mana beberapa nilai partisi berakhir dengan sebagian besar data dan menunda penyelesaian tugas.

- Ketika ada shuffle, tingkatkan nilainya. `spark.sql.shuffle.partitions` Ini juga dapat membantu mengatasi masalah memori apa pun saat mengocoknya.

  Bila Anda memiliki lebih dari 2.001 partisi shuffle, Spark menggunakan format memori terkompresi. Jika Anda memiliki nomor yang mendekati itu, Anda mungkin ingin menetapkan `spark.sql.shuffle.paritions` nilai di atas batas itu untuk mendapatkan representasi yang lebih efisien.

Optimalkan shuffle

Operasi tertentu, seperti `join()` dan`groupByKey()`, memerlukan Spark untuk melakukan shuffle. Shuffle adalah mekanisme Spark untuk mendistribusikan kembali data sehingga dikelompokkan secara berbeda di seluruh partisi. RDD Pengocokan dapat membantu memulihkan kemacetan kinerja. Namun, karena shuffling biasanya melibatkan penyalinan data antara pelaksana Spark, shuffle adalah operasi yang kompleks dan mahal. Misalnya, shuffle menghasilkan biaya berikut:

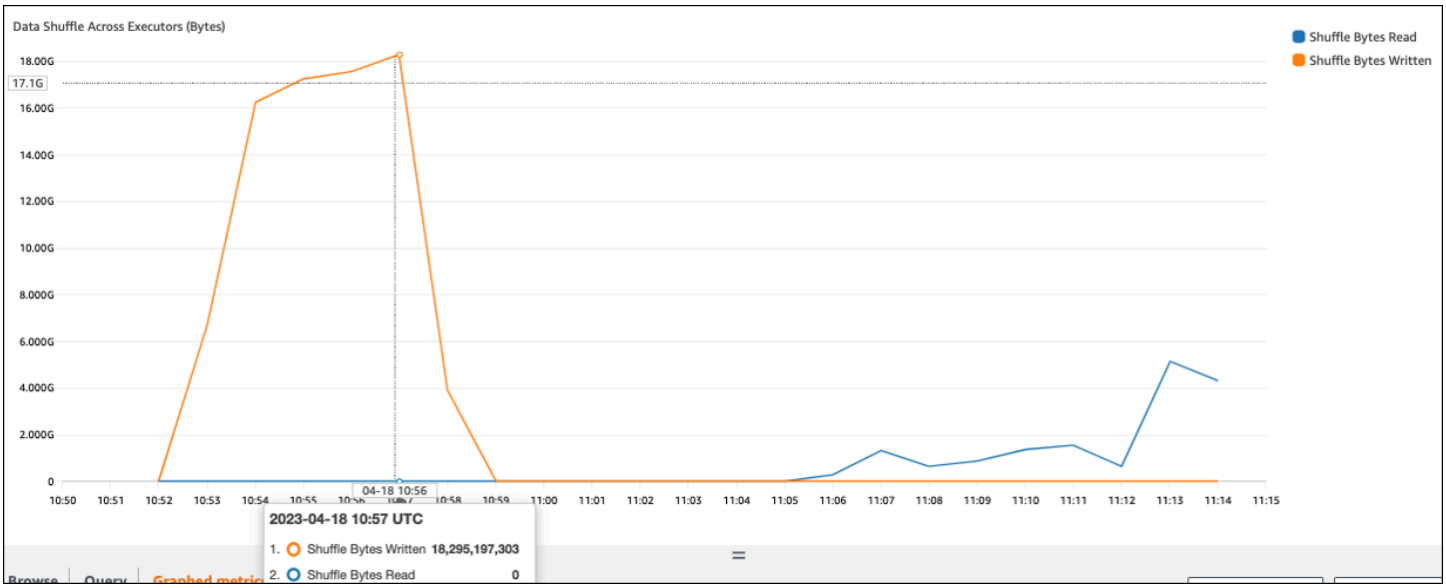
- Disk I/O:
  - Menghasilkan sejumlah besar file perantara pada disk.
- Jaringan I/O:
  - Membutuhkan banyak koneksi jaringan (Jumlah koneksi =`Mapper × Reducer`).
  - Karena catatan digabungkan ke RDD partisi baru yang mungkin di-host pada eksekutor Spark yang berbeda, sebagian besar dataset Anda mungkin berpindah antara pelaksana Spark melalui jaringan.
- CPUdan beban memori:
  - Mengurutkan nilai dan menggabungkan kumpulan data. Operasi ini direncanakan pada pelaksana, menempatkan beban berat pada pelaksana.

Shuffle adalah salah satu faktor terpenting dalam penurunan kinerja aplikasi Spark Anda. Saat menyimpan data perantara, itu dapat menghabiskan ruang pada disk lokal pelaksana, yang menyebabkan pekerjaan Spark gagal.

Anda dapat menilai kinerja shuffle Anda dalam CloudWatch metrik dan di UI Spark.

CloudWatch metrik

Jika nilai Shuffle Bytes Written tinggi dibandingkan dengan Shuffle Bytes Read, pekerjaan Spark Anda mungkin menggunakan operasi shuffle seperti atau. `join()` `groupByKey()`

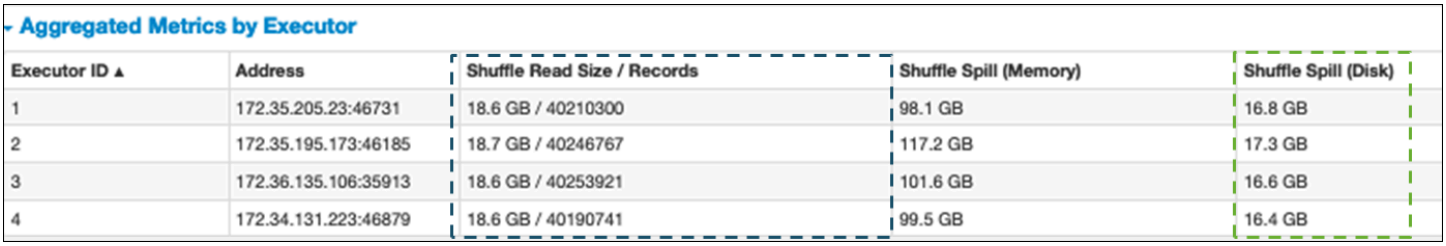


Spark UI

Pada tab Tahap UI Spark, Anda dapat memeriksa nilai Shuffle Read Size /Records. Anda juga dapat melihatnya di tab Executors.

Pada tangkapan layar berikut, setiap eksekutor bertukar sekitar 18.6GB/4020000 catatan dengan proses shuffle, dengan total ukuran baca acak sekitar 75 GB).

Kolom Shuffle Spill (Disk) menunjukkan sejumlah besar memori tumpahan data ke disk, yang dapat menyebabkan disk penuh atau masalah kinerja.



Aggregated Metrics by Executor

Executor ID ▲	Address	Shuffle Read Size / Records	Shuffle Spill (Memory)	Shuffle Spill (Disk)
1	172.35.205.23:46731	18.6 GB / 40210300	98.1 GB	16.8 GB
2	172.35.195.173:46185	18.7 GB / 40246767	117.2 GB	17.3 GB
3	172.36.135.106:35913	18.6 GB / 40253921	101.6 GB	16.6 GB
4	172.34.131.223:46879	18.6 GB / 40190741	99.5 GB	16.4 GB

Jika Anda mengamati gejala-gejala ini dan tahapannya terlalu lama jika dibandingkan dengan tujuan kinerja Anda, atau gagal dengan `Out Of Memory` atau `No space left on device` kesalahan, pertimbangkan solusi berikut.

Optimalkan bergabung

`join()`Operasi, yang bergabung dengan tabel, adalah operasi shuffle yang paling umum digunakan, tetapi sering kali merupakan hambatan kinerja. Karena bergabung adalah operasi yang mahal, kami sarankan untuk tidak menggunakannya kecuali itu penting untuk kebutuhan bisnis Anda. Periksa kembali apakah Anda memanfaatkan pipeline data Anda secara efisien dengan mengajukan pertanyaan berikut:

- Apakah Anda menghitung ulang bergabung yang juga dilakukan di pekerjaan lain yang dapat Anda gunakan kembali?
- Apakah Anda bergabung untuk menyelesaikan kunci asing untuk nilai-nilai yang tidak digunakan oleh konsumen output Anda?

Setelah Anda mengonfirmasi bahwa operasi gabungan Anda sangat penting untuk kebutuhan bisnis Anda, lihat opsi berikut untuk mengoptimalkan bergabung dengan cara yang memenuhi persyaratan Anda.

Gunakan pushdown sebelum bergabung

Saring baris dan kolom yang tidak perlu di DataFrame sebelum melakukan gabungan. Ini memiliki keuntungan sebagai berikut:

- Mengurangi jumlah transfer data selama shuffle
- Mengurangi jumlah pemrosesan di pelaksana Spark
- Mengurangi jumlah pemindaian data

```
# Default
df_joined  = df1.join(df2, ["product_id"])

# Use Pushdown
df1_select =
 df1.select("product_id","product_title","star_rating").filter(col("star_rating")>=4.0)
df2_select = df2.select("product_id","category_id")
df_joined  = df1_select.join(df2_select, ["product_id"])
```

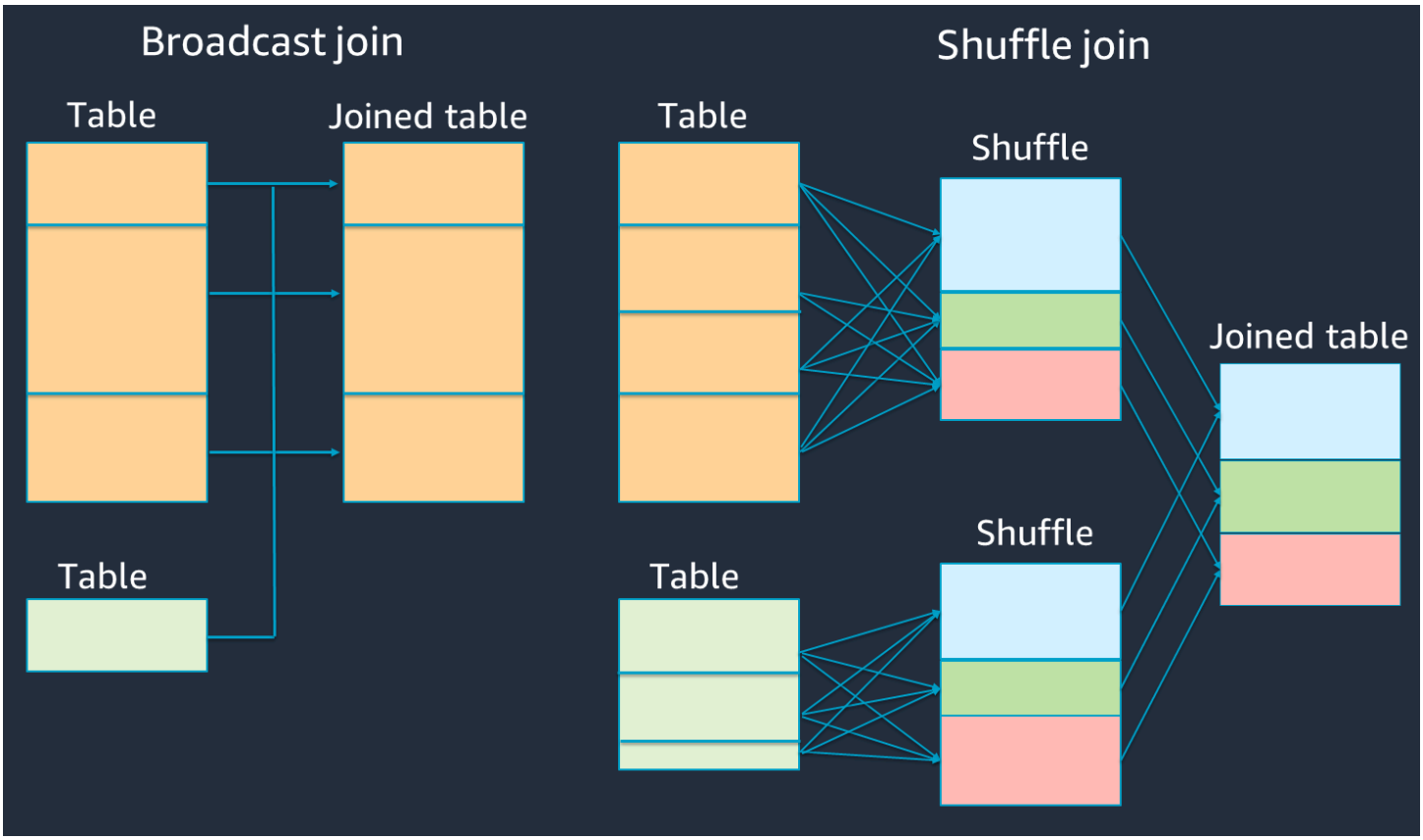
Gunakan DataFrame Gabung

Coba gunakan Spark tingkat tinggi API seperti SparkSQL, DataFrame, dan Datasets alih-alih atau bergabung. RDD API DynamicFrame Anda dapat DynamicFrame mengonversi DataFrame dengan panggilan metode sepertidyf.toDF(). Seperti yang dibahas dalam Topik utama di bagian Apache Spark, operasi gabungan ini secara internal memanfaatkan optimasi kueri oleh pengoptimal Catalyst.

Shuffle dan broadcast hash bergabung dan petunjuk

Spark mendukung dua jenis join: shuffle join dan broadcast hash join. Gabungan hash siaran tidak memerlukan pengocokan, dan dapat memerlukan pemrosesan yang lebih sedikit daripada gabungan acak. Namun, ini hanya berlaku ketika menggabungkan meja kecil ke meja besar. Saat bergabung dengan tabel yang dapat dimasukkan ke dalam memori seorang eksekutor Spark tunggal, pertimbangkan untuk menggunakan gabungan hash siaran.

Diagram berikut menunjukkan struktur tingkat tinggi dan langkah-langkah dari hash broadcast join dan shuffle join.



Rincian masing-masing bergabung adalah sebagai berikut:

- Shuffle bergabung:

  - Hash shuffle bergabung dengan dua tabel tanpa menyortir dan mendistribusikan gabungan antara dua tabel. Ini cocok untuk gabungan tabel kecil yang dapat disimpan dalam memori pelaksana Spark.
  - Gabungan sort-merge mendistribusikan dua tabel untuk digabungkan dengan kunci dan mengurutkannya sebelum bergabung. Ini cocok untuk bergabung dengan meja besar.
- Siaran hash bergabung:
  - Sebuah broadcast hash join mendorong yang lebih kecil RDD atau tabel ke masing-masing node pekerja. Kemudian ia menggabungkan sisi peta dengan setiap partisi yang lebih besar RDD atau tabel.

    Ini cocok untuk bergabung ketika salah satu dari Anda RDDs atau tabel dapat dimasukkan ke dalam memori atau dapat dibuat agar sesuai dengan memori. Sangat bermanfaat untuk melakukan broadcast hash join jika memungkinkan, karena tidak memerlukan shuffle. Anda dapat menggunakan petunjuk bergabung untuk meminta siaran bergabung dari Spark sebagai berikut.

    ```
    # DataFrame
    from pySpark.sql.functions import broadcast
    df_joined= df_big.join(broadcast(df_small), right_df[key] == left_df[key],
     how='inner')

    -- SparkSQL
    SELECT /*+ BROADCAST(t1) /  FROM t1 INNER JOIN t2 ON t1.key = t2.key;
    ```

    Untuk informasi selengkapnya tentang petunjuk bergabung, lihat Petunjuk bergabung.

Di AWS Glue 3.0 dan yang lebih baru, Anda dapat memanfaatkan siaran hash bergabung secara otomatis dengan mengaktifkan Adaptive Query Execution dan parameter tambahan. Eksekusi Kueri Adaptif mengonversi gabungan sort-gabungan menjadi gabungan hash siaran ketika statistik runtime dari kedua sisi gabungan lebih kecil dari ambang batas gabungan hash siaran adaptif.

Di AWS Glue 3.0, Anda dapat mengaktifkan Adaptive Query Execution dengan pengaturan`spark.sql.adaptive.enabled=true`. Eksekusi Kueri Adaptif diaktifkan secara default di AWS Glue 4.0.

Anda dapat mengatur parameter tambahan yang terkait dengan shuffles dan broadcast hash bergabung:

- `spark.sql.adaptive.localShuffleReader.enabled`
- `spark.sql.adaptive.autoBroadcastJoinThreshold`

Untuk informasi selengkapnya tentang parameter terkait, lihat Mengonversi gabungan sort-merge menjadi broadcast join.

Di AWS Glue 3.0 dan atau yang lebih baru, Anda dapat menggunakan petunjuk bergabung lainnya untuk shuffle untuk menyesuaikan perilaku Anda.

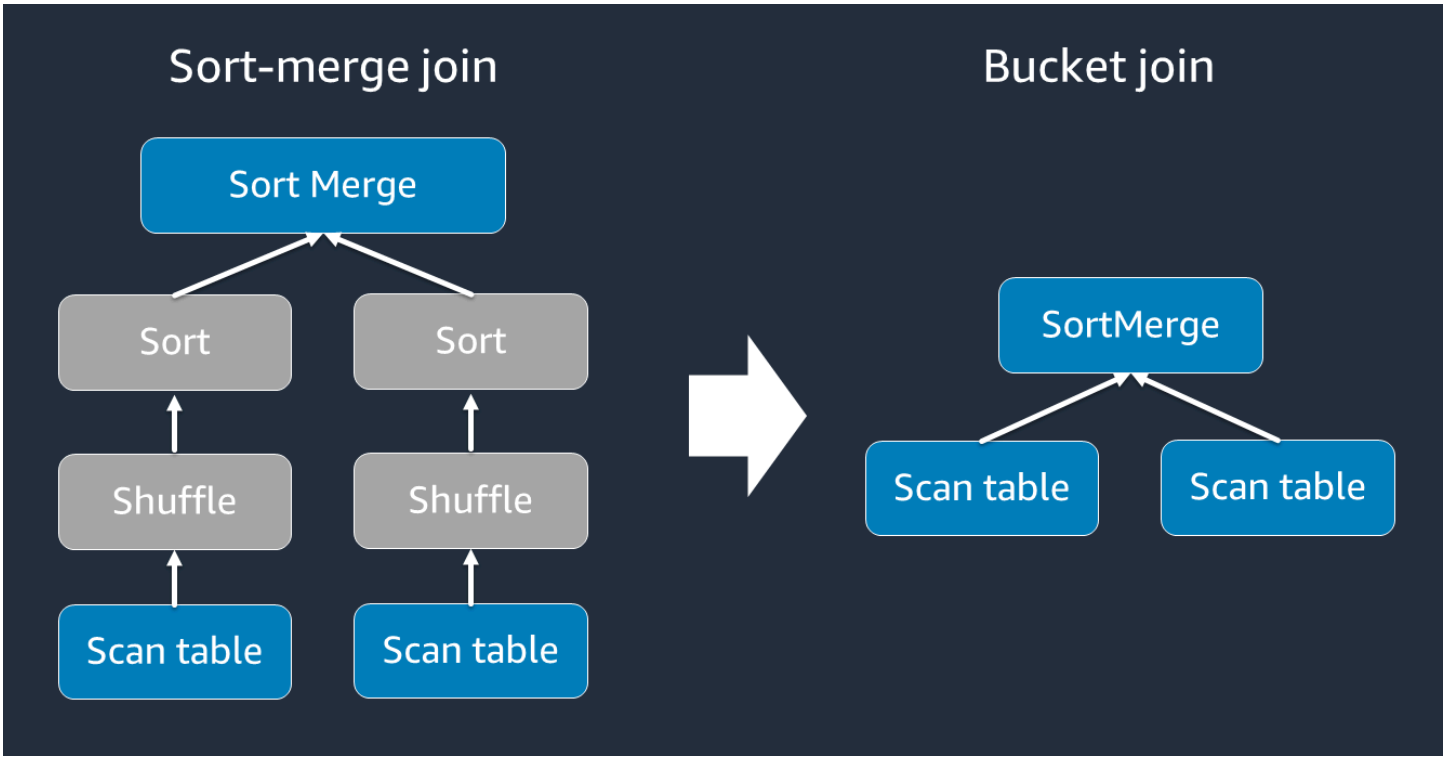
```
-- Join Hints for shuffle sort merge join
SELECT /*+ SHUFFLE_MERGE(t1) /  FROM t1 INNER JOIN t2 ON t1.key = t2.key;
SELECT /*+ MERGEJOIN(t2) /  FROM t1 INNER JOIN t2 ON t1.key = t2.key;
SELECT /*+ MERGE(t1) /  FROM t1 INNER JOIN t2 ON t1.key = t2.key;

-- Join Hints for shuffle hash join
SELECT /*+ SHUFFLE_HASH(t1) /  FROM t1 INNER JOIN t2 ON t1.key = t2.key;

-- Join Hints for shuffle-and-replicate nested loop join
SELECT /*+ SHUFFLE_REPLICATE_NL(t1) /  FROM t1 INNER JOIN t2 ON t1.key = t2.key;
```

Gunakan bucketing

Gabungan sort-merge membutuhkan dua fase, shuffle dan sort, dan kemudian menggabungkan. Kedua fase ini dapat membebani pelaksana Spark OOM dan menyebabkan serta masalah kinerja ketika beberapa pelaksana bergabung dan yang lainnya menyortir secara bersamaan. Dalam kasus seperti itu, dimungkinkan untuk bergabung secara efisien dengan menggunakan bucketing. Bucketing akan melakukan pra-shuffle dan pra-urutkan input Anda pada kunci gabungan, dan kemudian menulis data yang diurutkan itu ke tabel perantara. Biaya langkah pengocokan dan pengurutan dapat dikurangi saat menggabungkan tabel besar dengan mendefinisikan tabel perantara yang diurutkan terlebih dahulu.



Tabel bucketed berguna untuk hal-hal berikut:

- Data sering bergabung melalui kunci yang sama, seperti `account_id`
- Memuat tabel kumulatif harian, seperti tabel dasar dan delta yang dapat diselimuti pada kolom umum

Anda dapat membuat tabel berember dengan menggunakan kode berikut.

```
df.write.bucketBy(50, "account_id").sortBy("age").saveAsTable("bucketed_table")
```

Partisi ulang DataFrames pada kunci gabungan sebelum bergabung

Untuk mempartisi ulang keduanya DataFrames pada kunci gabungan sebelum bergabung, gunakan pernyataan berikut.

```
df1_repartitioned = df1.repartition(N,"join_key")
df2_repartitioned = df2.repartition(N,"join_key")
df_joined  = df1_repartitioned.join(df2_repartitioned,"product_id")
```

Ini akan mempartisi dua (masih terpisah) RDDs pada kunci join sebelum memulai join. Jika keduanya RDDs dipartisi pada kunci yang sama dengan kode partisi yang sama, RDD catatan bahwa rencana

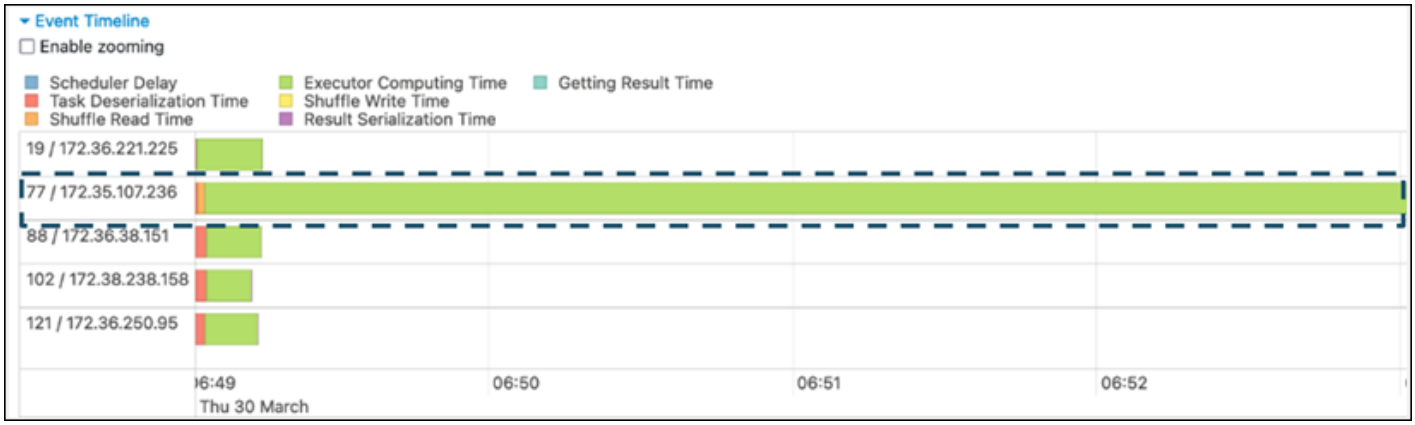
Anda untuk bergabung bersama akan memiliki kemungkinan besar ditempatkan bersama pada pekerja yang sama sebelum mengocoknya untuk bergabung. Ini dapat meningkatkan kinerja dengan mengurangi aktivitas jaringan dan kemiringan data selama bergabung.

Atasi kemiringan data

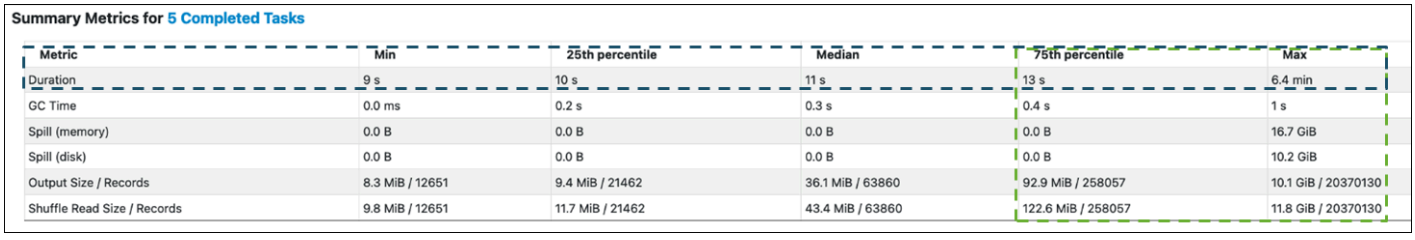
Kemiringan data adalah salah satu penyebab paling umum dari kemacetan untuk pekerjaan Spark. Ini terjadi ketika data tidak terdistribusi secara merata di seluruh RDD partisi. Hal ini menyebabkan tugas untuk partisi itu memakan waktu lebih lama daripada yang lain, menunda waktu pemrosesan aplikasi secara keseluruhan.

Untuk mengidentifikasi kemiringan data, nilai metrik berikut di UI Spark:

- Pada tab Stage di UI Spark, periksa halaman Timeline Acara. Anda dapat melihat distribusi tugas yang tidak merata di tangkapan layar berikut. Tugas yang didistribusikan tidak merata atau terlalu lama untuk dijalankan dapat menunjukkan kemiringan data.



- Halaman penting lainnya adalah Summary Metrics, yang menunjukkan statistik untuk tugas Spark. Tangkapan layar berikut menunjukkan metrik dengan persentil untuk Durasi, Waktu GC, Tumpahan (memori), Tumpahan (disk), dan sebagainya.



Summary Metrics for 5 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	9 s	10 s	11 s	13 s	6.4 min
GC Time	0.0 ms	0.2 s	0.3 s	0.4 s	1 s
Spill (memory)	0.0 B	0.0 B	0.0 B	0.0 B	16.7 GiB
Spill (disk)	0.0 B	0.0 B	0.0 B	0.0 B	10.2 GiB
Output Size / Records	8.3 MiB / 12651	9.4 MiB / 21462	36.1 MiB / 63860	92.9 MiB / 258057	10.1 GiB / 20370130
Shuffle Read Size / Records	9.8 MiB / 12651	11.7 MiB / 21462	43.4 MiB / 63860	122.6 MiB / 258057	11.8 GiB / 20370130

  Ketika tugas didistribusikan secara merata, Anda akan melihat angka yang sama di semua persentil. Ketika ada kemiringan data, Anda akan melihat nilai yang sangat bias di setiap persentil.

Dalam contoh, durasi tugas kurang dari 13 detik dalam persentil Min, 25, Median, dan 75. Sementara tugas Max memproses data 100 kali lebih banyak daripada persentil ke-75, durasinya 6,4 menit sekitar 30 kali lebih lama. Ini berarti bahwa setidaknya satu tugas (atau hingga 25 persen dari tugas) memakan waktu jauh lebih lama daripada tugas lainnya.

Jika Anda melihat data miring, coba yang berikut ini:

- Jika Anda menggunakan AWS Glue 3.0, aktifkan Adaptive Query Execution dengan pengaturan`spark.sql.adaptive.enabled=true`. Eksekusi Kueri Adaptif diaktifkan secara default di AWS Glue 4.0.

  Anda juga dapat menggunakan Eksekusi Kueri Adaptif untuk kemiringan data yang diperkenalkan oleh gabungan dengan menyetel parameter terkait berikut:

  - `spark.sql.adaptive.skewJoin.skewedPartitionFactor`
  - `spark.sql.adaptive.skewJoin.skewedPartitionThresholdInBytes`
  - `spark.sql.adaptive.advisoryPartitionSizeInBytes=128m (128 mebibytes or larger should be good)`
  - `spark.sql.adaptive.coalescePartitions.enabled=true (when you want to coalesce partitions)`

  Untuk informasi selengkapnya, lihat dokumentasi Apache Spark.

- Gunakan kunci dengan berbagai nilai untuk tombol gabungan. Dalam shuffle join, partisi ditentukan untuk setiap nilai hash dari sebuah kunci. Jika kardinalitas kunci gabungan terlalu rendah, fungsi hash lebih cenderung melakukan pekerjaan yang buruk dalam mendistribusikan data Anda di seluruh partisi. Oleh karena itu, jika aplikasi dan logika bisnis Anda mendukungnya, pertimbangkan untuk menggunakan kunci kardinalitas yang lebih tinggi atau kunci komposit.

```
# Use Single Primary Key
df_joined = df1_select.join(df2_select, ["primary_key"])

# Use Composite Key
df_joined = df1_select.join(df2_select, ["primary_key","secondary_key"])
```

Gunakan cache

Saat Anda menggunakan repetitif DataFrames, hindari pengocokan atau komputasi tambahan dengan menggunakan `df.cache()` atau `df.persist()` menyimpan hasil perhitungan di setiap memori pelaksana Spark dan pada disk. Spark juga mendukung bertahan RDDs pada disk atau mereplikasi di beberapa node (tingkat penyimpanan).

Misalnya, Anda dapat bertahan DataFrames dengan menambahkan`df.persist()`. Ketika cache tidak lagi diperlukan, Anda dapat menggunakan `unpersist` untuk membuang data cache.

```
df = spark.read.parquet("s3://<Bucket>/parquet/product_category=Books/")
df_high_rate = df.filter(col("star_rating")>=4.0)
df_high_rate.persist()

df_joined1 = df_high_rate.join(<Table1>, ["key"])
df_joined2 = df_high_rate.join(<Table2>, ["key"])
df_joined3 = df_high_rate.join(<Table3>, ["key"])
...
df_high_rate.unpersist()
```

Hapus tindakan Spark yang tidak dibutuhkan

Hindari menjalankan tindakan yang tidak perlu seperti`count`,`show`, atau`collect`. Seperti yang dibahas dalam Topik utama di bagian Apache Spark, Spark malas. Setiap transformasi RDD dapat dihitung ulang setiap kali Anda menjalankan tindakan di atasnya. Bila Anda menggunakan banyak tindakan Spark, beberapa akses sumber, perhitungan tugas, dan shuffle run untuk setiap tindakan akan dipanggil.

Jika Anda tidak membutuhkan `collect()` atau tindakan lain di lingkungan komersial Anda, pertimbangkan untuk menghapusnya.

> **Note**
>
> Hindari menggunakan Spark `collect()` di lingkungan komersial sebanyak mungkin. `collect()`Tindakan mengembalikan semua hasil perhitungan di pelaksana Spark ke driver Spark, yang dapat menyebabkan driver Spark mengembalikan kesalahan. OOM Untuk menghindari OOM kesalahan, Spark menetapkan secara `spark.driver.maxResultSize = 1GB` default, yang membatasi ukuran data maksimum yang dikembalikan ke driver Spark ke 1 GB.

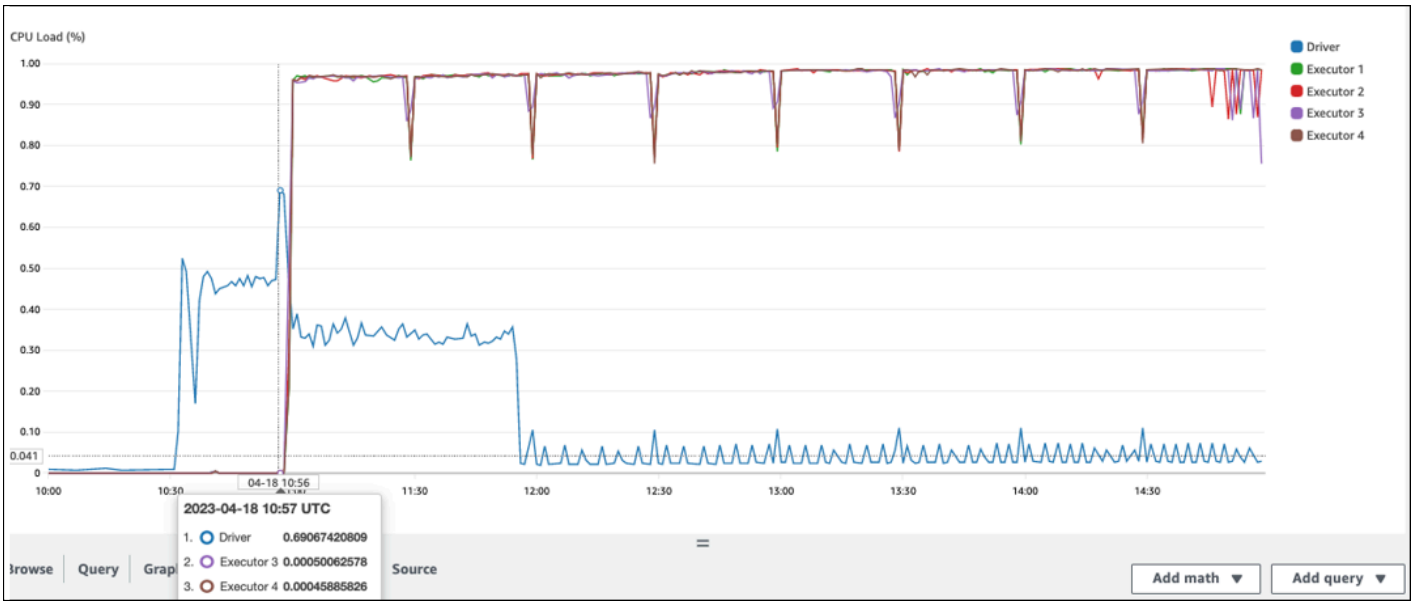
Minimalkan overhead perencanaan

Seperti yang dibahas Topik utama di Apache Spark, driver Spark menghasilkan rencana eksekusi. Berdasarkan rencana itu, tugas ditugaskan ke pelaksana Spark untuk pemrosesan terdistribusi. Namun, driver Spark dapat menjadi hambatan jika ada sejumlah besar file kecil atau jika AWS Glue Data Catalog berisi sejumlah besar partisi. Untuk mengidentifikasi overhead perencanaan yang tinggi, nilai metrik berikut.

CloudWatch metrik

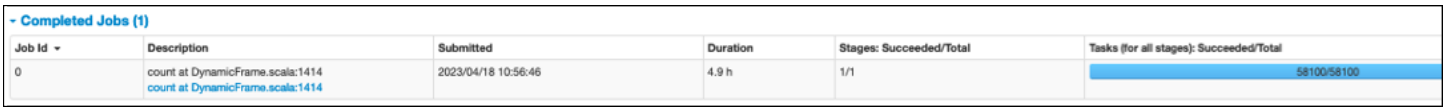
Periksa Pemanfaatan CPUBeban dan Memori untuk situasi berikut:

- CPUBeban driver Spark dan Pemanfaatan Memori dicatat sebagai tinggi. Biasanya, driver Spark tidak memproses data Anda, jadi CPU pemuatan dan pemanfaatan memori tidak melonjak. Namun, jika sumber data Amazon S3 memiliki terlalu banyak file kecil, mencantumkan semua objek S3 dan mengelola sejumlah besar tugas dapat menyebabkan pemanfaatan sumber daya menjadi tinggi.
- Ada celah panjang sebelum pemrosesan dimulai di pelaksana Spark. Dalam contoh screenshot berikut, CPU Beban pelaksana Spark terlalu rendah hingga 10:57, meskipun pekerjaan dimulai pada 10:00. AWS Glue Ini menunjukkan bahwa driver Spark mungkin membutuhkan waktu lama untuk membuat rencana eksekusi. Dalam contoh ini, mengambil sejumlah besar partisi di Katalog Data dan mencantumkan sejumlah besar file kecil di driver Spark membutuhkan waktu lama.



Spark UI

Pada tab Job di UI Spark, Anda dapat melihat waktu yang dikirimkan. Dalam contoh berikut, pengemudi Spark memulai job0 pada 10:56:46, meskipun pekerjaan dimulai pada 10:00:00. AWS Glue



Anda juga dapat melihat Tugas (untuk semua tahapan): Sukses/Total waktu pada tab Job. Dalam hal ini, jumlah tugas dicatat sebagai`58100`. Seperti yang dijelaskan di bagian Amazon S3 dari halaman tugas Parallelize, jumlah tugas kira-kira sesuai dengan jumlah objek S3. Ini berarti ada sekitar 58.100 objek di Amazon S3.

Untuk detail selengkapnya tentang pekerjaan dan timeline ini, tinjau tab Stage. Jika Anda mengamati kemacetan dengan driver Spark, pertimbangkan solusi berikut:

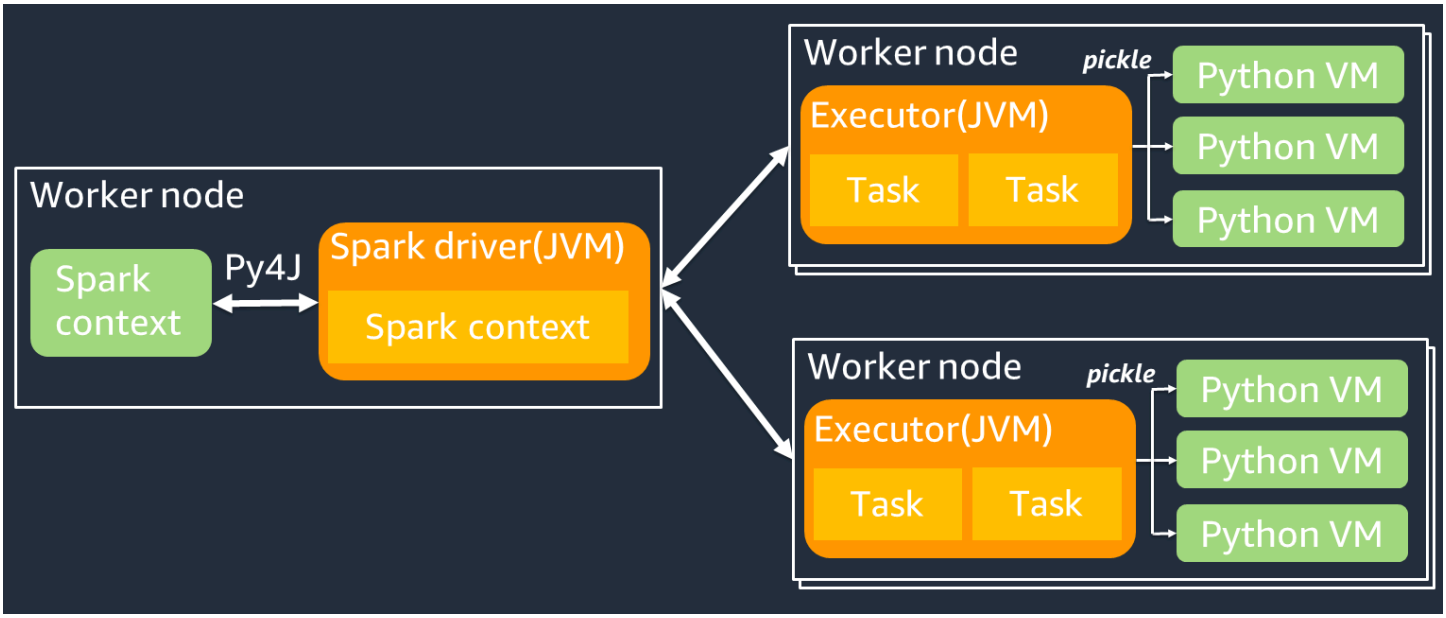
- Ketika Amazon S3 memiliki terlalu banyak file, pertimbangkan panduan tentang paralelisme berlebihan di bagian Terlalu banyak partisi di halaman tugas Parallelize.
- Ketika Amazon S3 memiliki terlalu banyak partisi, pertimbangkan panduan tentang partisi berlebihan di bagian partisi Amazon S3 Terlalu banyak di halaman Kurangi jumlah pemindaian data. Aktifkan indeks AWS Glue partisi jika ada banyak partisi untuk mengurangi latensi untuk mengambil metadata partisi dari Katalog Data. Untuk informasi selengkapnya, lihat Meningkatkan kinerja kueri menggunakan indeks AWS Glue partisi.
- Ketika JDBC memiliki terlalu banyak partisi, turunkan `hashpartition` nilainya.
- Ketika DynamoDB memiliki terlalu banyak partisi, turunkan nilainya. `dynamodb.splits`
- Ketika pekerjaan streaming memiliki terlalu banyak partisi, turunkan jumlah pecahan.

Optimalkan fungsi yang ditentukan pengguna

Fungsi yang ditentukan pengguna (UDFs) dan PySpark sering menurunkan `RDD.map` kinerja secara signifikan. Ini karena overhead yang diperlukan untuk secara akurat mewakili kode Python Anda dalam implementasi Scala yang mendasari Spark.

Diagram berikut menunjukkan arsitektur PySpark pekerjaan. Saat Anda menggunakan PySpark, driver Spark menggunakan pustaka Py4j untuk memanggil metode Java dari Python. Saat memanggil Spark SQL atau fungsi DataFrame bawaan, ada sedikit perbedaan kinerja antara Python dan

Scala karena fungsi berjalan pada setiap pelaksana menggunakan rencana eksekusi yang JVM dioptimalkan.



Jika Anda menggunakan logika Python Anda sendiri, seperti menggunakan`map/ mapPartitions/ udf`, tugas akan berjalan di lingkungan runtime Python. Mengelola dua lingkungan menciptakan biaya overhead. Selain itu, data Anda dalam memori harus diubah untuk digunakan oleh fungsi bawaan lingkungan JVM runtime. Pickle adalah format serialisasi yang digunakan secara default untuk pertukaran antara runtime dan JVM Python. Namun, biaya serialisasi dan biaya deserialisasi ini sangat tinggi, sehingga UDFs ditulis dalam Java atau Scala lebih cepat daripada Python. UDFs

Untuk menghindari overhead serialisasi dan deserialisasi PySpark, pertimbangkan hal berikut:

- Gunakan fungsi Spark bawaan - Pertimbangkan untuk mengganti SQL fungsi Anda sendiri UDF atau peta dengan Spark SQL atau fungsi DataFrame bawaan. Saat menjalankan Spark SQL atau fungsi DataFrame bawaan, ada sedikit perbedaan kinerja antara Python dan Scala karena tugas ditangani pada masing-masing pelaksana. JVM
- Implementasikan UDFs di Scala atau Java - Pertimbangkan untuk menggunakan UDF yang ditulis dalam Java atau Scala, karena mereka berjalan di. JVM
- Gunakan Apache Arrow berbasis UDFs untuk beban kerja vektor — Pertimbangkan untuk menggunakan berbasis Arrow. UDFs Fitur ini juga dikenal sebagai Vectorized UDF (Pandas). UDF Apache Arrow adalah format data dalam memori bahasa agnostik yang AWS Glue dapat digunakan untuk mentransfer data secara efisien antara dan proses Python. JVM Ini saat ini paling bermanfaat bagi pengguna Python yang bekerja dengan Panda atau data. NumPy

Panah adalah format kolumnar (vektor). Penggunaannya tidak otomatis dan mungkin memerlukan beberapa perubahan kecil pada konfigurasi atau kode untuk memanfaatkan sepenuhnya dan memastikan kompatibilitas. Untuk detail dan keterbatasan lebih lanjut, lihat Apache Arrow di PySpark.

Contoh berikut membandingkan inkremental dasar UDF dalam Python standar, sebagai VektorUDF, dan di Spark. SQL

Python Standar UDF

Contoh waktu adalah 3,20 (detik).

Contoh kode

```
# DataSet
df = spark.range(10000000).selectExpr("id AS a","id AS b")

# UDF Example
def plus(a,b):
    return a+b
spark.udf.register("plus",plus)

df.selectExpr("count(plus(a,b))").collect()
```

Rencana eksekusi

```
== Physical Plan ==
AdaptiveSparkPlan isFinalPlan=false
+- HashAggregate(keys=[], functions=[count(pythonUDF0#124)])
+- Exchange SinglePartition, ENSURE_REQUIREMENTS, [id=#580]
+- HashAggregate(keys=[], functions=[partial_count(pythonUDF0#124)])
+- Project [pythonUDF0#124]
+- BatchEvalPython [plus(a#116L, b#117L)], [pythonUDF0#124]
+- Project [id#114L AS a#116L, id#114L AS b#117L]
+- Range (0, 10000000, step=1, splits=16)
```

Divektorkan UDF

Contoh waktu adalah 0,59 (detik).

Vektor UDF 5 kali lebih cepat dari contoh sebelumnya. UDF MemeriksaPhysical Plan, Anda dapat melihatArrowEvalPython, yang menunjukkan aplikasi ini di-vektor oleh Apache Arrow. Untuk mengaktifkan VectorizedUDF, Anda harus menentukan `spark.sql.execution.arrow.pyspark.enabled = true` dalam kode Anda.

Contoh kode

```
# Vectorized UDF
from pyspark.sql.types import LongType
from pyspark.sql.functions import count, pandas_udf

# Enable Apache Arrow Support
spark.conf.set("spark.sql.execution.arrow.pyspark.enabled", "true")

# DataSet
df = spark.range(10000000).selectExpr("id AS a","id AS b")

# Annotate pandas_udf to use Vectorized UDF
@pandas_udf(LongType())
def pandas_plus(a,b):
    return a+b
spark.udf.register("pandas_plus",pandas_plus)

df.selectExpr("count(pandas_plus(a,b))").collect()
```

Rencana eksekusi

```
== Physical Plan ==
AdaptiveSparkPlan isFinalPlan=false
+- HashAggregate(keys=[], functions=[count(pythonUDF0#1082L)],
 output=[count(pandas_plus(a, b))#1080L])
+- Exchange SinglePartition, ENSURE_REQUIREMENTS, [id=#5985]
+- HashAggregate(keys=[], functions=[partial_count(pythonUDF0#1082L)],
 output=[count#1084L])
+- Project [pythonUDF0#1082L]
+- ArrowEvalPython [pandas_plus(a#1074L, b#1075L)], [pythonUDF0#1082L], 200
+- Project [id#1072L AS a#1074L, id#1072L AS b#1075L]
+- Range (0, 10000000, step=1, splits=16)
```

Percikan SQL

Contoh waktu adalah 0,087 (detik).

Spark jauh SQL lebih cepat daripada VectorizedUDF, karena tugas dijalankan pada setiap eksekutor tanpa JVM runtime Python. Jika Anda dapat mengganti fungsi Anda UDF dengan fungsi bawaan, kami sarankan untuk melakukannya.

Contoh kode

```
df.createOrReplaceTempView("test")
spark.sql("select count(a+b) from test").collect()
```

Menggunakan panda untuk data besar

Jika Anda sudah terbiasa dengan panda dan ingin menggunakan Spark untuk big data, Anda dapat menggunakan API panda di Spark. AWS Glue 4.0 dan kemudian mendukungnya. Untuk memulai, Anda dapat menggunakan notebook resmi Quickstart: Pandas API on Spark. Untuk informasi lebih lanjut, lihat PySpark dokumentasi.

Sumber daya

- AWS Glue
- Penyetelan Kinerja (Panduan Spark SQL)
- AWS Glue Lokakarya Optimasi

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	—	Januari 2, 2024

AWS Glosarium Panduan Preskriptif

Berikut ini adalah istilah yang umum digunakan dalam strategi, panduan, dan pola yang disediakan oleh Panduan AWS Preskriptif. Untuk menyarankan entri, silakan gunakan tautan Berikan umpan balik di akhir glosarium.

Nomor

7 Rs

Tujuh strategi migrasi umum untuk memindahkan aplikasi ke cloud. Strategi ini dibangun di atas 5 Rs yang diidentifikasi Gartner pada tahun 2011 dan terdiri dari yang berikut:

- Refactor/Re-Architect — Memindahkan aplikasi dan memodifikasi arsitekturnya dengan memanfaatkan sepenuhnya fitur cloud-native untuk meningkatkan kelincahan, kinerja, dan skalabilitas. Ini biasanya melibatkan porting sistem operasi dan database. Contoh: Migrasikan database Oracle lokal Anda ke Amazon Aurora PostgreSQL Compatible Edition.
- Replatform (angkat dan bentuk ulang) — Pindahkan aplikasi ke cloud, dan perkenalkan beberapa tingkat pengoptimalan untuk memanfaatkan kemampuan cloud. Contoh: Migrasikan database Oracle lokal Anda ke Amazon Relational Database Service (Amazon RDS) untuk Oracle di. AWS Cloud
- Pembelian kembali (drop and shop) - Beralih ke produk yang berbeda, biasanya dengan beralih dari lisensi tradisional ke model SaaS. Contoh: Migrasikan sistem manajemen hubungan pelanggan (CRM) Anda ke Salesforce.com.
- Rehost (lift dan shift) — Pindahkan aplikasi ke cloud tanpa membuat perubahan apa pun untuk memanfaatkan kemampuan cloud. Contoh: Migrasikan database Oracle lokal Anda ke Oracle pada instans EC2 di. AWS Cloud
- Relokasi (hypervisor-level lift and shift) — Pindahkan infrastruktur ke cloud tanpa membeli perangkat keras baru, menulis ulang aplikasi, atau memodifikasi operasi yang ada. Anda memigrasikan server dari platform lokal ke layanan cloud untuk platform yang sama. Contoh: Migrasikan Microsoft Hyper-V aplikasi ke AWS.
- Pertahankan (kunjungi kembali) - Simpan aplikasi di lingkungan sumber Anda. Ini mungkin termasuk aplikasi yang memerlukan refactoring besar, dan Anda ingin menunda pekerjaan itu sampai nanti, dan aplikasi lama yang ingin Anda pertahankan, karena tidak ada pembenaran bisnis untuk memigrasikannya.

- Pensiun — Menonaktifkan atau menghapus aplikasi yang tidak lagi diperlukan di lingkungan sumber Anda.

A

ABAC

Lihat kontrol akses berbasis atribut.

layanan abstrak

Lihat layanan terkelola.

ASAM

Lihat atomisitas, konsistensi, isolasi, daya tahan.

migrasi aktif-aktif

Metode migrasi database di mana database sumber dan target tetap sinkron (dengan menggunakan alat replikasi dua arah atau operasi penulisan ganda), dan kedua database menangani transaksi dari menghubungkan aplikasi selama migrasi. Metode ini mendukung migrasi dalam batch kecil yang terkontrol alih-alih memerlukan pemotongan satu kali. Ini lebih fleksibel tetapi membutuhkan lebih banyak pekerjaan daripada migrasi aktif-pasif.

migrasi aktif-pasif

Metode migrasi database di mana database sumber dan target disimpan dalam sinkron, tetapi hanya database sumber yang menangani transaksi dari menghubungkan aplikasi sementara data direplikasi ke database target. Basis data target tidak menerima transaksi apa pun selama migrasi.

fungsi agregat

Fungsi SQL yang beroperasi pada sekelompok baris dan menghitung nilai pengembalian tunggal untuk grup. Contoh fungsi agregat meliputi `SUM` dan`MAX`.

AI

Lihat kecerdasan buatan.

AIOps

Lihat operasi kecerdasan buatan.

anonimisasi

Proses menghapus informasi pribadi secara permanen dalam kumpulan data. Anonimisasi dapat membantu melindungi privasi pribadi. Data anonim tidak lagi dianggap sebagai data pribadi.

anti-pola

Solusi yang sering digunakan untuk masalah berulang di mana solusinya kontra-produktif, tidak efektif, atau kurang efektif daripada alternatif.

kontrol aplikasi

Pendekatan keamanan yang memungkinkan penggunaan hanya aplikasi yang disetujui untuk membantu melindungi sistem dari malware.

portofolio aplikasi

Kumpulan informasi rinci tentang setiap aplikasi yang digunakan oleh organisasi, termasuk biaya untuk membangun dan memelihara aplikasi, dan nilai bisnisnya. Informasi ini adalah kunci untuk penemuan portofolio dan proses analisis dan membantu mengidentifikasi dan memprioritaskan aplikasi yang akan dimigrasi, dimodernisasi, dan dioptimalkan.

kecerdasan buatan (AI)

Bidang ilmu komputer yang didedikasikan untuk menggunakan teknologi komputasi untuk melakukan fungsi kognitif yang biasanya terkait dengan manusia, seperti belajar, memecahkan masalah, dan mengenali pola. Untuk informasi lebih lanjut, lihat Apa itu Kecerdasan Buatan?

operasi kecerdasan buatan (AIOps)

Proses menggunakan teknik pembelajaran mesin untuk memecahkan masalah operasional, mengurangi insiden operasional dan intervensi manusia, dan meningkatkan kualitas layanan. Untuk informasi selengkapnya tentang cara AIOps digunakan dalam strategi AWS migrasi, lihat panduan integrasi operasi.

enkripsi asimetris

Algoritma enkripsi yang menggunakan sepasang kunci, kunci publik untuk enkripsi dan kunci pribadi untuk dekripsi. Anda dapat berbagi kunci publik karena tidak digunakan untuk dekripsi, tetapi akses ke kunci pribadi harus sangat dibatasi.

atomisitas, konsistensi, isolasi, daya tahan (ACID)

Satu set properti perangkat lunak yang menjamin validitas data dan keandalan operasional database, bahkan dalam kasus kesalahan, kegagalan daya, atau masalah lainnya.

kontrol akses berbasis atribut (ABAC)

Praktik membuat izin berbutir halus berdasarkan atribut pengguna, seperti departemen, peran pekerjaan, dan nama tim. Untuk informasi selengkapnya, lihat ABAC untuk AWS dokumentasi AWS Identity and Access Management (IAM).

sumber data otoritatif

Lokasi di mana Anda menyimpan versi utama data, yang dianggap sebagai sumber informasi yang paling dapat diandalkan. Anda dapat menyalin data dari sumber data otoritatif ke lokasi lain untuk tujuan memproses atau memodifikasi data, seperti menganonimkan, menyunting, atau membuat nama samaran.

Zona Ketersediaan

Lokasi berbeda di dalam Wilayah AWS yang terisolasi dari kegagalan di Availability Zone lainnya dan menyediakan konektivitas jaringan latensi rendah yang murah ke Availability Zone lainnya di Wilayah yang sama.

AWS Kerangka Adopsi Cloud (AWS CAF)

Kerangka pedoman dan praktik terbaik AWS untuk membantu organisasi mengembangkan rencana yang efisien dan efektif untuk bergerak dengan sukses ke cloud. AWS CAF mengatur panduan ke dalam enam area fokus yang disebut perspektif: bisnis, orang, tata kelola, platform, keamanan, dan operasi. Perspektif bisnis, orang, dan tata kelola fokus pada keterampilan dan proses bisnis; perspektif platform, keamanan, dan operasi fokus pada keterampilan dan proses teknis. Misalnya, perspektif masyarakat menargetkan pemangku kepentingan yang menangani sumber daya manusia (SDM), fungsi kepegawaian, dan manajemen orang. Untuk perspektif ini, AWS CAF memberikan panduan untuk pengembangan, pelatihan, dan komunikasi orang untuk membantu mempersiapkan organisasi untuk adopsi cloud yang sukses. Untuk informasi lebih lanjut, lihat situs web AWS CAF dan whitepaper AWS CAF.

AWS Kerangka Kualifikasi Beban Kerja (AWS WQF)

Alat yang mengevaluasi beban kerja migrasi database, merekomendasikan strategi migrasi, dan memberikan perkiraan kerja. AWS WQF disertakan dengan AWS Schema Conversion Tool ()AWS SCT. Ini menganalisis skema database dan objek kode, kode aplikasi, dependensi, dan karakteristik kinerja, dan memberikan laporan penilaian.

B

bot buruk

Bot yang dimaksudkan untuk mengganggu atau menyebabkan kerugian bagi individu atau organisasi.

BCP

Lihat perencanaan kontinuitas bisnis.

grafik perilaku

Pandangan interaktif yang terpadu tentang perilaku dan interaksi sumber daya dari waktu ke waktu. Anda dapat menggunakan grafik perilaku dengan Amazon Detective untuk memeriksa upaya logon yang gagal, panggilan API yang mencurigakan, dan tindakan serupa. Untuk informasi selengkapnya, lihat Data dalam grafik perilaku di dokumentasi Detektif.

sistem big-endian

Sistem yang menyimpan byte paling signifikan terlebih dahulu. Lihat juga endianness.

klasifikasi biner

Sebuah proses yang memprediksi hasil biner (salah satu dari dua kelas yang mungkin). Misalnya, model ML Anda mungkin perlu memprediksi masalah seperti “Apakah email ini spam atau bukan spam?” atau “Apakah produk ini buku atau mobil?”

filter mekar

Struktur data probabilistik dan efisien memori yang digunakan untuk menguji apakah suatu elemen adalah anggota dari suatu himpunan.

deployment biru/hijau

Strategi penyebaran tempat Anda membuat dua lingkungan yang terpisah namun identik. Anda menjalankan versi aplikasi saat ini di satu lingkungan (biru) dan versi aplikasi baru di lingkungan lain (hijau). Strategi ini membantu Anda dengan cepat memutar kembali dengan dampak minimal.

bot

Aplikasi perangkat lunak yang menjalankan tugas otomatis melalui internet dan mensimulasikan aktivitas atau interaksi manusia. Beberapa bot berguna atau bermanfaat, seperti perayap web yang mengindeks informasi di internet. Beberapa bot lain, yang dikenal sebagai bot buruk, dimaksudkan untuk mengganggu atau membahayakan individu atau organisasi.

botnet

Jaringan bot yang terinfeksi oleh malware dan berada di bawah kendali satu pihak, yang dikenal sebagai bot herder atau operator bot. Botnet adalah mekanisme paling terkenal untuk skala bot dan dampaknya.

cabang

Area berisi repositori kode. Cabang pertama yang dibuat dalam repositori adalah cabang utama. Anda dapat membuat cabang baru dari cabang yang ada, dan Anda kemudian dapat mengembangkan fitur atau memperbaiki bug di cabang baru. Cabang yang Anda buat untuk membangun fitur biasanya disebut sebagai cabang fitur. Saat fitur siap dirilis, Anda menggabungkan cabang fitur kembali ke cabang utama. Untuk informasi selengkapnya, lihat Tentang cabang (GitHub dokumentasi).

akses break-glass

Dalam keadaan luar biasa dan melalui proses yang disetujui, cara cepat bagi pengguna untuk mendapatkan akses ke Akun AWS yang biasanya tidak memiliki izin untuk mengaksesnya. Untuk informasi lebih lanjut, lihat indikator Implementasikan prosedur break-glass dalam panduan Well-Architected AWS .

strategi brownfield

Infrastruktur yang ada di lingkungan Anda. Saat mengadopsi strategi brownfield untuk arsitektur sistem, Anda merancang arsitektur di sekitar kendala sistem dan infrastruktur saat ini. Jika Anda memperluas infrastruktur yang ada, Anda dapat memadukan strategi brownfield dan greenfield.

cache penyangga

Area memori tempat data yang paling sering diakses disimpan.

kemampuan bisnis

Apa yang dilakukan bisnis untuk menghasilkan nilai (misalnya, penjualan, layanan pelanggan, atau pemasaran). Arsitektur layanan mikro dan keputusan pengembangan dapat didorong oleh kemampuan bisnis. Untuk informasi selengkapnya, lihat bagian Terorganisir di sekitar kemampuan bisnis dari Menjalankan layanan mikro kontainer di whitepaper. AWS

perencanaan kelangsungan bisnis (BCP)

Rencana yang membahas dampak potensial dari peristiwa yang mengganggu, seperti migrasi skala besar, pada operasi dan memungkinkan bisnis untuk melanjutkan operasi dengan cepat.

C

KAFE

Lihat Kerangka Adopsi AWS Cloud.

penyebaran kenari

Rilis versi yang lambat dan bertahap untuk pengguna akhir. Ketika Anda yakin, Anda menyebarkan versi baru dan mengganti versi saat ini secara keseluruhan.

CCoE

Lihat Cloud Center of Excellence.

CDC

Lihat mengubah pengambilan data.

ubah pengambilan data (CDC)

Proses melacak perubahan ke sumber data, seperti tabel database, dan merekam metadata tentang perubahan tersebut. Anda dapat menggunakan CDC untuk berbagai tujuan, seperti mengaudit atau mereplikasi perubahan dalam sistem target untuk mempertahankan sinkronisasi.

rekayasa kekacauan

Sengaja memperkenalkan kegagalan atau peristiwa yang mengganggu untuk menguji ketahanan sistem. Anda dapat menggunakan AWS Fault Injection Service (AWS FIS) untuk melakukan eksperimen yang menekankan AWS beban kerja Anda dan mengevaluasi responsnya.

CI/CD

Lihat integrasi berkelanjutan dan pengiriman berkelanjutan.

klasifikasi

Proses kategorisasi yang membantu menghasilkan prediksi. Model ML untuk masalah klasifikasi memprediksi nilai diskrit. Nilai diskrit selalu berbeda satu sama lain. Misalnya, model mungkin perlu mengevaluasi apakah ada mobil dalam gambar atau tidak.

Enkripsi sisi klien

Enkripsi data secara lokal, sebelum target layanan AWS menerimanya.

Cloud Center of Excellence (CCoE)

Tim multi-disiplin yang mendorong upaya adopsi cloud di seluruh organisasi, termasuk mengembangkan praktik terbaik cloud, memobilisasi sumber daya, menetapkan jadwal migrasi, dan memimpin organisasi melalui transformasi skala besar. Untuk informasi selengkapnya, lihat posting CCoE di Blog Strategi AWS Cloud Perusahaan.

komputasi cloud

Teknologi cloud yang biasanya digunakan untuk penyimpanan data jarak jauh dan manajemen perangkat IoT. Cloud computing umumnya terhubung ke teknologi edge computing.

model operasi cloud

Dalam organisasi TI, model operasi yang digunakan untuk membangun, mematangkan, dan mengoptimalkan satu atau lebih lingkungan cloud. Untuk informasi selengkapnya, lihat Membangun Model Operasi Cloud Anda.

tahap adopsi cloud

Empat fase yang biasanya dilalui organisasi ketika mereka bermigrasi ke AWS Cloud:

- Proyek — Menjalankan beberapa proyek terkait cloud untuk bukti konsep dan tujuan pembelajaran
- Foundation — Melakukan investasi dasar untuk meningkatkan adopsi cloud Anda (misalnya, membuat landing zone, mendefinisikan CCoE, membuat model operasi)
- Migrasi — Migrasi aplikasi individual
- Re-invention — Mengoptimalkan produk dan layanan, dan berinovasi di cloud

Tahapan ini didefinisikan oleh Stephen Orban dalam posting blog The Journey Toward Cloud-First & the Stages of Adoption di blog Strategi Perusahaan. AWS Cloud Untuk informasi tentang bagaimana kaitannya dengan strategi AWS migrasi, lihat panduan kesiapan migrasi.

CMDB

Lihat database manajemen konfigurasi.

repositori kode

Lokasi di mana kode sumber dan aset lainnya, seperti dokumentasi, sampel, dan skrip, disimpan dan diperbarui melalui proses kontrol versi. Repositori cloud umum termasuk GitHub atau. AWS CodeCommit Setiap versi kode disebut cabang. Dalam struktur layanan mikro, setiap repositori

dikhususkan untuk satu bagian fungsionalitas. Pipa CI/CD tunggal dapat menggunakan beberapa repositori.

cache dingin

Cache buffer yang kosong, tidak terisi dengan baik, atau berisi data basi atau tidak relevan. Ini mempengaruhi kinerja karena instance database harus membaca dari memori utama atau disk, yang lebih lambat daripada membaca dari cache buffer.

data dingin

Data yang jarang diakses dan biasanya historis. Saat menanyakan jenis data ini, kueri lambat biasanya dapat diterima. Memindahkan data ini ke tingkat penyimpanan atau kelas yang berkinerja lebih rendah dan lebih murah dapat mengurangi biaya.

visi komputer (CV)

Bidang AI yang menggunakan pembelajaran mesin untuk menganalisis dan mengekstrak informasi dari format visual seperti gambar dan video digital. Misalnya, AWS Panorama menawarkan perangkat yang menambahkan CV ke jaringan kamera lokal, dan Amazon SageMaker menyediakan algoritme pemrosesan gambar untuk CV.

konfigurasi drift

Untuk beban kerja, konfigurasi berubah dari status yang diharapkan. Ini dapat menyebabkan beban kerja menjadi tidak patuh, dan biasanya bertahap dan tidak disengaja.

database manajemen konfigurasi (CMDB)

Repositori yang menyimpan dan mengelola informasi tentang database dan lingkungan TI, termasuk komponen perangkat keras dan perangkat lunak dan konfigurasinya. Anda biasanya menggunakan data dari CMDB dalam penemuan portofolio dan tahap analisis migrasi.

paket kesesuaian

Kumpulan AWS Config aturan dan tindakan remediasi yang dapat Anda kumpulkan untuk menyesuaikan kepatuhan dan pemeriksaan keamanan Anda. Anda dapat menerapkan paket kesesuaian sebagai entitas tunggal di Akun AWS dan Wilayah, atau di seluruh organisasi, dengan menggunakan templat YAMM. Untuk informasi selengkapnya, lihat Paket kesesuaian dalam dokumentasi. AWS Config

integrasi berkelanjutan dan pengiriman berkelanjutan (CI/CD)

Proses mengotomatiskan sumber, membangun, menguji, pementasan, dan tahap produksi dari proses rilis perangkat lunak. CI/CD umumnya digambarkan sebagai pipa. CI/CD dapat membantu

Anda mengotomatiskan proses, meningkatkan produktivitas, meningkatkan kualitas kode, dan memberikan lebih cepat. Untuk informasi lebih lanjut, lihat Manfaat pengiriman berkelanjutan. CD juga dapat berarti penerapan berkelanjutan. Untuk informasi selengkapnya, lihat Continuous Delivery vs Continuous Deployment.

CV

Lihat visi komputer.

D

data saat istirahat

Data yang stasioner di jaringan Anda, seperti data yang ada di penyimpanan.

klasifikasi data

Proses untuk mengidentifikasi dan mengkategorikan data dalam jaringan Anda berdasarkan kekritisan dan sensitivitasnya. Ini adalah komponen penting dari setiap strategi manajemen risiko keamanan siber karena membantu Anda menentukan perlindungan dan kontrol retensi yang tepat untuk data. Klasifikasi data adalah komponen pilar keamanan dalam AWS Well-Architected Framework. Untuk informasi selengkapnya, lihat Klasifikasi data.

penyimpangan data

Variasi yang berarti antara data produksi dan data yang digunakan untuk melatih model ML, atau perubahan yang berarti dalam data input dari waktu ke waktu. Penyimpangan data dapat mengurangi kualitas, akurasi, dan keadilan keseluruhan dalam prediksi model ML.

data dalam transit

Data yang aktif bergerak melalui jaringan Anda, seperti antara sumber daya jaringan.

jala data

Kerangka arsitektur yang menyediakan kepemilikan data terdistribusi dan terdesentralisasi dengan manajemen dan tata kelola terpusat.

minimalisasi data

Prinsip pengumpulan dan pemrosesan hanya data yang sangat diperlukan. Mempraktikkan minimalisasi data AWS Cloud dapat mengurangi risiko privasi, biaya, dan jejak karbon analitik Anda.

perimeter data

Satu set pagar pembatas pencegahan di AWS lingkungan Anda yang membantu memastikan bahwa hanya identitas tepercaya yang mengakses sumber daya tepercaya dari jaringan yang diharapkan. Untuk informasi selengkapnya, lihat Membangun perimeter data pada AWS.

prapemrosesan data

Untuk mengubah data mentah menjadi format yang mudah diuraikan oleh model ML Anda. Preprocessing data dapat berarti menghapus kolom atau baris tertentu dan menangani nilai yang hilang, tidak konsisten, atau duplikat.

asal data

Proses melacak asal dan riwayat data sepanjang siklus hidupnya, seperti bagaimana data dihasilkan, ditransmisikan, dan disimpan.

subjek data

Individu yang datanya dikumpulkan dan diproses.

gudang data

Sistem manajemen data yang mendukung intelijen bisnis, seperti analitik. Gudang data biasanya berisi sejumlah besar data historis, dan biasanya digunakan untuk kueri dan analisis.

bahasa definisi database (DDL)

Pernyataan atau perintah untuk membuat atau memodifikasi struktur tabel dan objek dalam database.

bahasa manipulasi basis data (DHTML)

Pernyataan atau perintah untuk memodifikasi (memasukkan, memperbarui, dan menghapus) informasi dalam database.

DDL

Lihat bahasa definisi database.

ansambel yang dalam

Untuk menggabungkan beberapa model pembelajaran mendalam untuk prediksi. Anda dapat menggunakan ansambel dalam untuk mendapatkan prediksi yang lebih akurat atau untuk memperkirakan ketidakpastian dalam prediksi.

pembelajaran mendalam

Subbidang ML yang menggunakan beberapa lapisan jaringan saraf tiruan untuk mengidentifikasi pemetaan antara data input dan variabel target yang diinginkan.

defense-in-depth

Pendekatan keamanan informasi di mana serangkaian mekanisme dan kontrol keamanan dilapisi dengan cermat di seluruh jaringan komputer untuk melindungi kerahasiaan, integritas, dan ketersediaan jaringan dan data di dalamnya. Saat Anda mengadopsi strategi ini AWS, Anda menambahkan beberapa kontrol pada lapisan AWS Organizations struktur yang berbeda untuk membantu mengamankan sumber daya. Misalnya, defense-in-depth pendekatan mungkin menggabungkan otentikasi multi-faktor, segmentasi jaringan, dan enkripsi.

administrator yang didelegasikan

Di AWS Organizations, layanan yang kompatibel dapat mendaftarkan akun AWS anggota untuk mengelola akun organisasi dan mengelola izin untuk layanan tersebut. Akun ini disebut administrator yang didelegasikan untuk layanan itu. Untuk informasi selengkapnya dan daftar layanan yang kompatibel, lihat Layanan yang berfungsi dengan AWS Organizations AWS Organizations dokumentasi.

deployment

Proses pembuatan aplikasi, fitur baru, atau perbaikan kode tersedia di lingkungan target. Deployment melibatkan penerapan perubahan dalam basis kode dan kemudian membangun dan menjalankan basis kode itu di lingkungan aplikasi.

lingkungan pengembangan

Lihat lingkungan.

kontrol detektif

Kontrol keamanan yang dirancang untuk mendeteksi, mencatat, dan memperingatkan setelah suatu peristiwa terjadi. Kontrol ini adalah garis pertahanan kedua, memperingatkan Anda tentang peristiwa keamanan yang melewati kontrol pencegahan yang ada. Untuk informasi selengkapnya, lihat Kontrol Detektif dalam Menerapkan kontrol keamanan pada. AWS

pemetaan aliran nilai pengembangan (DVSM)

Sebuah proses yang digunakan untuk mengidentifikasi dan memprioritaskan kendala yang mempengaruhi kecepatan dan kualitas dalam siklus hidup pengembangan perangkat lunak. DVSM memperluas proses pemetaan aliran nilai yang awalnya dirancang untuk praktik

manufaktur ramping. Ini berfokus pada langkah-langkah dan tim yang diperlukan untuk menciptakan dan memindahkan nilai melalui proses pengembangan perangkat lunak.

kembar digital

Representasi virtual dari sistem dunia nyata, seperti bangunan, pabrik, peralatan industri, atau jalur produksi. Kembar digital mendukung pemeliharaan prediktif, pemantauan jarak jauh, dan optimalisasi produksi.

tabel dimensi

Dalam skema bintang, tabel yang lebih kecil yang berisi atribut data tentang data kuantitatif dalam tabel fakta. Atribut tabel dimensi biasanya bidang teks atau angka diskrit yang berperilaku seperti teks. Atribut ini biasanya digunakan untuk pembatasan kueri, pemfilteran, dan pelabelan set hasil.

musibah

Peristiwa yang mencegah beban kerja atau sistem memenuhi tujuan bisnisnya di lokasi utama yang digunakan. Peristiwa ini dapat berupa bencana alam, kegagalan teknis, atau akibat dari tindakan manusia, seperti kesalahan konfigurasi yang tidak disengaja atau serangan malware.

pemulihan bencana (DR)

Strategi dan proses yang Anda gunakan untuk meminimalkan downtime dan kehilangan data yang disebabkan oleh bencana. Untuk informasi selengkapnya, lihat Disaster Recovery of Workloads on AWS: Recovery in the Cloud in the AWS Well-Architected Framework.

DML~

Lihat bahasa manipulasi database.

desain berbasis domain

Pendekatan untuk mengembangkan sistem perangkat lunak yang kompleks dengan menghubungkan komponennya ke domain yang berkembang, atau tujuan bisnis inti, yang dilayani oleh setiap komponen. Konsep ini diperkenalkan oleh Eric Evans dalam bukunya, Domain-Driven Design: Tackling Complexity in the Heart of Software (Boston: Addison-Wesley Professional, 2003). Untuk informasi tentang cara menggunakan desain berbasis domain dengan pola gambar pencekik, lihat Memodernisasi layanan web Microsoft ASP.NET (ASMX) lama secara bertahap menggunakan container dan Amazon API Gateway.

DR

Lihat pemulihan bencana.

deteksi drift

Melacak penyimpangan dari konfigurasi dasar. Misalnya, Anda dapat menggunakan AWS CloudFormation untuk mendeteksi penyimpangan dalam sumber daya sistem, atau Anda dapat menggunakannya AWS Control Tower untuk mendeteksi perubahan di landing zone yang mungkin memengaruhi kepatuhan terhadap persyaratan tata kelola.

DVSM

Lihat pemetaan aliran nilai pengembangan.

E

EDA

Lihat analisis data eksplorasi.

komputasi tepi

Teknologi yang meningkatkan daya komputasi untuk perangkat pintar di tepi jaringan IoT. Jika dibandingkan dengan komputasi awan, komputasi tepi dapat mengurangi latensi komunikasi dan meningkatkan waktu respons.

enkripsi

Proses komputasi yang mengubah data plaintext, yang dapat dibaca manusia, menjadi ciphertext.

kunci enkripsi

String kriptografi dari bit acak yang dihasilkan oleh algoritma enkripsi. Panjang kunci dapat bervariasi, dan setiap kunci dirancang agar tidak dapat diprediksi dan unik.

endianness

Urutan byte disimpan dalam memori komputer. Sistem big-endian menyimpan byte paling signifikan terlebih dahulu. Sistem little-endian menyimpan byte paling tidak signifikan terlebih dahulu.

titik akhir

Lihat titik akhir layanan.

layanan endpoint

Layanan yang dapat Anda host di cloud pribadi virtual (VPC) untuk dibagikan dengan pengguna lain. Anda dapat membuat layanan endpoint dengan AWS PrivateLink dan memberikan izin kepada prinsipal lain Akun AWS atau ke AWS Identity and Access Management (IAM). Akun atau prinsipal ini dapat terhubung ke layanan endpoint Anda secara pribadi dengan membuat titik akhir VPC antarmuka. Untuk informasi selengkapnya, lihat Membuat layanan titik akhir di dokumentasi Amazon Virtual Private Cloud (Amazon VPC).

perencanaan sumber daya perusahaan (ERP)

Sistem yang mengotomatiskan dan mengelola proses bisnis utama (seperti akuntansi, MES, dan manajemen proyek) untuk suatu perusahaan.

enkripsi amplop

Proses mengenkripsi kunci enkripsi dengan kunci enkripsi lain. Untuk informasi selengkapnya, lihat Enkripsi amplop dalam dokumentasi AWS Key Management Service (AWS KMS).

lingkungan

Sebuah contoh dari aplikasi yang sedang berjalan. Berikut ini adalah jenis lingkungan yang umum dalam komputasi awan:

- Development Environment — Sebuah contoh dari aplikasi yang berjalan yang hanya tersedia untuk tim inti yang bertanggung jawab untuk memelihara aplikasi. Lingkungan pengembangan digunakan untuk menguji perubahan sebelum mempromosikannya ke lingkungan atas. Jenis lingkungan ini kadang-kadang disebut sebagai lingkungan pengujian.
- lingkungan yang lebih rendah — Semua lingkungan pengembangan untuk aplikasi, seperti yang digunakan untuk build awal dan pengujian.
- lingkungan produksi — Sebuah contoh dari aplikasi yang berjalan yang pengguna akhir dapat mengakses. Dalam pipa CI/CD, lingkungan produksi adalah lingkungan penyebaran terakhir.
- lingkungan atas — Semua lingkungan yang dapat diakses oleh pengguna selain tim pengembangan inti. Ini dapat mencakup lingkungan produksi, lingkungan praproduksi, dan lingkungan untuk pengujian penerimaan pengguna.

epik

Dalam metodologi tangkas, kategori fungsional yang membantu mengatur dan memprioritaskan pekerjaan Anda. Epik memberikan deskripsi tingkat tinggi tentang persyaratan dan tugas

implementasi. Misalnya, epos keamanan AWS CAF mencakup manajemen identitas dan akses, kontrol detektif, keamanan infrastruktur, perlindungan data, dan respons insiden. Untuk informasi selengkapnya tentang epos dalam strategi AWS migrasi, lihat panduan implementasi program.

ERP

Lihat perencanaan sumber daya perusahaan.

analisis data eksplorasi (EDA)

Proses menganalisis dataset untuk memahami karakteristik utamanya. Anda mengumpulkan atau mengumpulkan data dan kemudian melakukan penyelidikan awal untuk menemukan pola, mendeteksi anomali, dan memeriksa asumsi. EDA dilakukan dengan menghitung statistik ringkasan dan membuat visualisasi data.

F

tabel fakta

Tabel tengah dalam skema bintang. Ini menyimpan data kuantitatif tentang operasi bisnis. Biasanya, tabel fakta berisi dua jenis kolom: kolom yang berisi ukuran dan yang berisi kunci asing ke tabel dimensi.

gagal cepat

Filosofi yang menggunakan pengujian yang sering dan bertahap untuk mengurangi siklus hidup pengembangan. Ini adalah bagian penting dari pendekatan tangkas.

batas isolasi kesalahan

Dalam AWS Cloud, batas seperti Availability Zone, Wilayah AWS, control plane, atau data plane yang membatasi efek kegagalan dan membantu meningkatkan ketahanan beban kerja. Untuk informasi selengkapnya, lihat Batas Isolasi AWS Kesalahan.

cabang fitur

Lihat cabang.

fitur

Data input yang Anda gunakan untuk membuat prediksi. Misalnya, dalam konteks manufaktur, fitur bisa berupa gambar yang diambil secara berkala dari lini manufaktur.

pentingnya fitur

Seberapa signifikan fitur untuk prediksi model. Ini biasanya dinyatakan sebagai skor numerik yang dapat dihitung melalui berbagai teknik, seperti Shapley Additive Explanations (SHAP) dan gradien terintegrasi. Untuk informasi lebih lanjut, lihat Interpretabilitas model pembelajaran mesin dengan:.AWS

transformasi fitur

Untuk mengoptimalkan data untuk proses ML, termasuk memperkaya data dengan sumber tambahan, menskalakan nilai, atau mengekstrak beberapa set informasi dari satu bidang data. Hal ini memungkinkan model ML untuk mendapatkan keuntungan dari data. Misalnya, jika Anda memecah tanggal “2021-05-27 00:15:37" menjadi “2021", “Mei”, “Kamis”, dan “15", Anda dapat membantu algoritme pembelajaran mempelajari pola bernuansa yang terkait dengan komponen data yang berbeda.

FGAC

Lihat kontrol akses berbutir halus.

kontrol akses berbutir halus (FGAC)

Penggunaan beberapa kondisi untuk mengizinkan atau menolak permintaan akses.

migrasi flash-cut

Metode migrasi database yang menggunakan replikasi data berkelanjutan melalui pengambilan data perubahan untuk memigrasikan data dalam waktu sesingkat mungkin, alih-alih menggunakan pendekatan bertahap. Tujuannya adalah untuk menjaga downtime seminimal mungkin.

G

pemblokiran geografis

Lihat pembatasan geografis.

pembatasan geografis (pemblokiran geografis)

Di Amazon CloudFront, opsi untuk mencegah pengguna di negara tertentu mengakses distribusi konten. Anda dapat menggunakan daftar izinkan atau daftar blokir untuk menentukan negara yang disetujui dan dilarang. Untuk informasi selengkapnya, lihat Membatasi distribusi geografis konten Anda dalam dokumentasi. CloudFront

Alur kerja Gitflow

Pendekatan di mana lingkungan bawah dan atas menggunakan cabang yang berbeda dalam repositori kode sumber. Alur kerja Gitflow dianggap warisan, dan alur kerja berbasis batang adalah pendekatan modern yang lebih disukai.

strategi greenfield

Tidak adanya infrastruktur yang ada di lingkungan baru. Saat mengadopsi strategi greenfield untuk arsitektur sistem, Anda dapat memilih semua teknologi baru tanpa batasan kompatibilitas dengan infrastruktur yang ada, juga dikenal sebagai brownfield. Jika Anda memperluas infrastruktur yang ada, Anda dapat memadukan strategi brownfield dan greenfield.

pagar pembatas

Aturan tingkat tinggi yang membantu mengatur sumber daya, kebijakan, dan kepatuhan di seluruh unit organisasi (OU). Pagar pembatas preventif menegakkan kebijakan untuk memastikan keselarasan dengan standar kepatuhan. Mereka diimplementasikan dengan menggunakan kebijakan kontrol layanan dan batas izin IAM. Detective guardrails mendeteksi pelanggaran kebijakan dan masalah kepatuhan, dan menghasilkan peringatan untuk remediasi. Mereka diimplementasikan dengan menggunakan AWS Config, AWS Security Hub, Amazon GuardDuty AWS Trusted Advisor, Amazon Inspector, dan pemeriksaan khusus AWS Lambda .

H

HA

Lihat ketersediaan tinggi.

migrasi database heterogen

Memigrasi database sumber Anda ke database target yang menggunakan mesin database yang berbeda (misalnya, Oracle ke Amazon Aurora). Migrasi heterogen biasanya merupakan bagian dari upaya arsitektur ulang, dan mengubah skema dapat menjadi tugas yang kompleks. AWS menyediakan AWS SCT yang membantu dengan konversi skema.

ketersediaan tinggi (HA)

Kemampuan beban kerja untuk beroperasi terus menerus, tanpa intervensi, jika terjadi tantangan atau bencana. Sistem HA dirancang untuk gagal secara otomatis, secara konsisten memberikan kinerja berkualitas tinggi, dan menangani beban dan kegagalan yang berbeda dengan dampak kinerja minimal.

modernisasi sejarawan

Pendekatan yang digunakan untuk memodernisasi dan meningkatkan sistem teknologi operasional (OT) untuk melayani kebutuhan industri manufaktur dengan lebih baik. Sejarawan adalah jenis database yang digunakan untuk mengumpulkan dan menyimpan data dari berbagai sumber di pabrik.

migrasi database homogen

Memigrasi database sumber Anda ke database target yang berbagi mesin database yang sama (misalnya, Microsoft SQL Server ke Amazon RDS for SQL Server). Migrasi homogen biasanya merupakan bagian dari upaya rehosting atau replatforming. Anda dapat menggunakan utilitas database asli untuk memigrasi skema.

data panas

Data yang sering diakses, seperti data real-time atau data translasi terbaru. Data ini biasanya memerlukan tingkat atau kelas penyimpanan berkinerja tinggi untuk memberikan respons kueri yang cepat.

perbaikan terbaru

Perbaikan mendesak untuk masalah kritis dalam lingkungan produksi. Karena urgensinya, perbaikan terbaru biasanya dibuat di luar alur kerja DevOps rilis biasa.

periode hypercare

Segera setelah cutover, periode waktu ketika tim migrasi mengelola dan memantau aplikasi yang dimigrasi di cloud untuk mengatasi masalah apa pun. Biasanya, periode ini panjangnya 1-4 hari. Pada akhir periode hypercare, tim migrasi biasanya mentransfer tanggung jawab untuk aplikasi ke tim operasi cloud.

I

IAc

Lihat infrastruktur sebagai kode.

kebijakan berbasis identitas

Kebijakan yang dilampirkan pada satu atau beberapa prinsip IAM yang mendefinisikan izin mereka dalam lingkungan. AWS Cloud

aplikasi idle

Aplikasi yang memiliki penggunaan CPU dan memori rata-rata antara 5 dan 20 persen selama periode 90 hari. Dalam proyek migrasi, adalah umum untuk menghentikan aplikasi ini atau mempertahankannya di tempat.

IIoT

Lihat Internet of Things industri.

infrastruktur yang tidak dapat diubah

Model yang menyebarkan infrastruktur baru untuk beban kerja produksi alih-alih memperbarui, menambal, atau memodifikasi infrastruktur yang ada. Infrastruktur yang tidak dapat diubah secara inheren lebih konsisten, andal, dan dapat diprediksi daripada infrastruktur yang dapat berubah. Untuk informasi selengkapnya, lihat praktik terbaik Deploy using immutable infrastructure di AWS Well-Architected Framework.

masuk (masuknya) VPC

Dalam arsitektur AWS multi-akun, VPC yang menerima, memeriksa, dan merutekan koneksi jaringan dari luar aplikasi. Arsitektur Referensi AWS Keamanan merekomendasikan pengaturan akun Jaringan Anda dengan VPC masuk, keluar, dan inspeksi untuk melindungi antarmuka dua arah antara aplikasi Anda dan internet yang lebih luas.

migrasi inkremental

Strategi cutover di mana Anda memigrasikan aplikasi Anda dalam bagian-bagian kecil alih-alih melakukan satu cutover penuh. Misalnya, Anda mungkin hanya memindahkan beberapa layanan mikro atau pengguna ke sistem baru pada awalnya. Setelah Anda memverifikasi bahwa semuanya berfungsi dengan baik, Anda dapat secara bertahap memindahkan layanan mikro atau pengguna tambahan hingga Anda dapat menonaktifkan sistem lama Anda. Strategi ini mengurangi risiko yang terkait dengan migrasi besar.

Industri 4.0

Sebuah istilah yang diperkenalkan oleh Klaus Schwab pada tahun 2016 untuk merujuk pada modernisasi proses manufaktur melalui kemajuan dalam konektivitas, data real-time, otomatisasi, analitik, dan AI/ML.

infrastruktur

Semua sumber daya dan aset yang terkandung dalam lingkungan aplikasi.

infrastruktur sebagai kode (IAc)

Proses penyediaan dan pengelolaan infrastruktur aplikasi melalui satu set file konfigurasi. IAc dirancang untuk membantu Anda memusatkan manajemen infrastruktur, menstandarisasi sumber daya, dan menskalakan dengan cepat sehingga lingkungan baru dapat diulang, andal, dan konsisten.

Internet of Things industri (IIoT)

Penggunaan sensor dan perangkat yang terhubung ke internet di sektor industri, seperti manufaktur, energi, otomotif, perawatan kesehatan, ilmu kehidupan, dan pertanian. Untuk informasi selengkapnya, lihat Membangun strategi transformasi digital Internet of Things (IIoT) industri.

inspeksi VPC

Dalam arsitektur AWS multi-akun, VPC terpusat yang mengelola inspeksi lalu lintas jaringan antara VPC (dalam hal yang sama atau berbeda Wilayah AWS), internet, dan jaringan lokal. Arsitektur Referensi AWS Keamanan merekomendasikan pengaturan akun Jaringan Anda dengan VPC masuk, keluar, dan inspeksi untuk melindungi antarmuka dua arah antara aplikasi Anda dan internet yang lebih luas.

Internet of Things (IoT)

Jaringan objek fisik yang terhubung dengan sensor atau prosesor tertanam yang berkomunikasi dengan perangkat dan sistem lain melalui internet atau melalui jaringan komunikasi lokal. Untuk informasi selengkapnya, lihat Apa itu IoT?

interpretabilitas

Karakteristik model pembelajaran mesin yang menggambarkan sejauh mana manusia dapat memahami bagaimana prediksi model bergantung pada inputnya. Untuk informasi selengkapnya, lihat Interpretabilitas model pembelajaran mesin dengan AWS.

IoT

Lihat Internet of Things.

Perpustakaan informasi TI (ITIL)

Serangkaian praktik terbaik untuk memberikan layanan TI dan menyelaraskan layanan ini dengan persyaratan bisnis. ITIL menyediakan dasar untuk ITSM.

Manajemen layanan TI (ITSM)

Kegiatan yang terkait dengan merancang, menerapkan, mengelola, dan mendukung layanan TI untuk suatu organisasi. Untuk informasi tentang mengintegrasikan operasi cloud dengan alat ITSM, lihat panduan integrasi operasi.

ITIL

Lihat perpustakaan informasi TI.

ITSM

Lihat manajemen layanan TI.

L

kontrol akses berbasis label (LBAC)

Implementasi kontrol akses wajib (MAC) di mana pengguna dan data itu sendiri masing-masing secara eksplisit diberi nilai label keamanan. Persimpangan antara label keamanan pengguna dan label keamanan data menentukan baris dan kolom mana yang dapat dilihat oleh pengguna.

landing zone

Landing zone adalah AWS lingkungan multi-akun yang dirancang dengan baik yang dapat diskalakan dan aman. Ini adalah titik awal dari mana organisasi Anda dapat dengan cepat meluncurkan dan menyebarkan beban kerja dan aplikasi dengan percaya diri dalam lingkungan keamanan dan infrastruktur mereka. Untuk informasi selengkapnya tentang zona pendaratan, lihat Menyiapkan lingkungan multi-akun AWS yang aman dan dapat diskalakan.

migrasi besar

Migrasi 300 atau lebih server.

LBAC

Lihat kontrol akses berbasis label.

hak istimewa paling sedikit

Praktik keamanan terbaik untuk memberikan izin minimum yang diperlukan untuk melakukan tugas. Untuk informasi selengkapnya, lihat Menerapkan izin hak istimewa terkecil dalam dokumentasi IAM.

angkat dan geser

Lihat 7 Rs.

sistem endian kecil

Sebuah sistem yang menyimpan byte paling tidak signifikan terlebih dahulu. Lihat juga endianness.

lingkungan yang lebih rendah

Lihat lingkungan.

M

pembelajaran mesin (ML)

Jenis kecerdasan buatan yang menggunakan algoritma dan teknik untuk pengenalan pola dan pembelajaran. ML menganalisis dan belajar dari data yang direkam, seperti data Internet of Things (IoT), untuk menghasilkan model statistik berdasarkan pola. Untuk informasi selengkapnya, lihat Machine Learning.

cabang utama

Lihat cabang.

malware

Perangkat lunak yang dirancang untuk membahayakan keamanan atau privasi komputer. Malware dapat mengganggu sistem komputer, membocorkan informasi sensitif, atau mendapatkan akses yang tidak sah. Contoh malware termasuk virus, worm, ransomware, Trojan horse, spyware, dan keyloggers.

layanan terkelola

layanan AWS yang AWS mengoperasikan lapisan infrastruktur, sistem operasi, dan platform, dan Anda mengakses titik akhir untuk menyimpan dan mengambil data. Amazon Simple Storage Service (Amazon S3) dan Amazon DynamoDB adalah contoh layanan terkelola. Ini juga dikenal sebagai layanan abstrak.

sistem eksekusi manufaktur (MES)

Sistem perangkat lunak untuk melacak, memantau, mendokumentasikan, dan mengendalikan proses produksi yang mengubah bahan baku menjadi produk jadi di lantai toko.

PETA

Lihat Program Percepatan Migrasi.

mekanisme

Proses lengkap di mana Anda membuat alat, mendorong adopsi alat, dan kemudian memeriksa hasilnya untuk melakukan penyesuaian. Mekanisme adalah siklus yang memperkuat dan meningkatkan dirinya sendiri saat beroperasi. Untuk informasi lebih lanjut, lihat Membangun mekanisme di AWS Well-Architected Framework.

akun anggota

Semua Akun AWS selain akun manajemen yang merupakan bagian dari organisasi di AWS Organizations. Akun dapat menjadi anggota dari hanya satu organisasi pada suatu waktu.

MES

Lihat sistem eksekusi manufaktur.

Transportasi Telemetri Antrian Pesan (MQTT)

Protokol komunikasi ringan machine-to-machine (M2M), berdasarkan pola terbitkan/berlangganan, untuk perangkat IoT yang dibatasi sumber daya.

layanan mikro

Layanan kecil dan independen yang berkomunikasi melalui API yang terdefinisi dengan baik dan biasanya dimiliki oleh tim kecil yang mandiri. Misalnya, sistem asuransi mungkin mencakup layanan mikro yang memetakan kemampuan bisnis, seperti penjualan atau pemasaran, atau subdomain, seperti pembelian, klaim, atau analitik. Manfaat layanan mikro termasuk kelincahan, penskalaan yang fleksibel, penyebaran yang mudah, kode yang dapat digunakan kembali, dan ketahanan. Untuk informasi selengkapnya, lihat Mengintegrasikan layanan mikro dengan menggunakan layanan tanpa AWS server.

arsitektur microservices

Pendekatan untuk membangun aplikasi dengan komponen independen yang menjalankan setiap proses aplikasi sebagai layanan mikro. Layanan mikro ini berkomunikasi melalui antarmuka yang terdefinisi dengan baik dengan menggunakan API ringan. Setiap layanan mikro dalam arsitektur ini dapat diperbarui, digunakan, dan diskalakan untuk memenuhi permintaan fungsi tertentu dari suatu aplikasi. Untuk informasi selengkapnya, lihat Menerapkan layanan mikro di AWS.

Program Percepatan Migrasi (MAP)

AWS Program yang menyediakan dukungan konsultasi, pelatihan, dan layanan untuk membantu organisasi membangun fondasi operasional yang kuat untuk pindah ke cloud, dan untuk membantu mengimbangi biaya awal migrasi. MAP mencakup metodologi migrasi untuk mengeksekusi migrasi lama dengan cara metodis dan seperangkat alat untuk mengotomatisasi dan mempercepat skenario migrasi umum.

migrasi dalam skala

Proses memindahkan sebagian besar portofolio aplikasi ke cloud dalam gelombang, dengan lebih banyak aplikasi bergerak pada tingkat yang lebih cepat di setiap gelombang. Fase ini menggunakan praktik terbaik dan pelajaran yang dipetik dari fase sebelumnya untuk mengimplementasikan pabrik migrasi tim, alat, dan proses untuk merampingkan migrasi beban kerja melalui otomatisasi dan pengiriman tangkas. Ini adalah fase ketiga dari strategi AWS migrasi.

pabrik migrasi

Tim lintas fungsi yang merampingkan migrasi beban kerja melalui pendekatan otomatis dan gesit. Tim pabrik migrasi biasanya mencakup operasi, analis dan pemilik bisnis, insinyur migrasi, pengembang, dan DevOps profesional yang bekerja di sprint. Antara 20 dan 50 persen portofolio aplikasi perusahaan terdiri dari pola berulang yang dapat dioptimalkan dengan pendekatan pabrik. Untuk informasi selengkapnya, lihat diskusi tentang pabrik migrasi dan panduan Pabrik Migrasi Cloud di kumpulan konten ini.

metadata migrasi

Informasi tentang aplikasi dan server yang diperlukan untuk menyelesaikan migrasi. Setiap pola migrasi memerlukan satu set metadata migrasi yang berbeda. Contoh metadata migrasi termasuk subnet target, grup keamanan, dan akun. AWS

pola migrasi

Tugas migrasi berulang yang merinci strategi migrasi, tujuan migrasi, dan aplikasi atau layanan migrasi yang digunakan. Contoh: Rehost migrasi ke Amazon EC2 AWS dengan Layanan Migrasi Aplikasi.

Penilaian Portofolio Migrasi (MPA)

Alat online yang menyediakan informasi untuk memvalidasi kasus bisnis untuk bermigrasi ke. AWS Cloud MPA menyediakan penilaian portofolio terperinci (ukuran kanan server, harga,

perbandingan TCO, analisis biaya migrasi) serta perencanaan migrasi (analisis data aplikasi dan pengumpulan data, pengelompokan aplikasi, prioritas migrasi, dan perencanaan gelombang). Alat MPA (memerlukan login) tersedia gratis untuk semua AWS konsultan dan konsultan APN Partner.

Penilaian Kesiapan Migrasi (MRA)

Proses mendapatkan wawasan tentang status kesiapan cloud organisasi, mengidentifikasi kekuatan dan kelemahan, dan membangun rencana aksi untuk menutup kesenjangan yang diidentifikasi, menggunakan CAF. AWS Untuk informasi selengkapnya, lihat panduan kesiapan migrasi. MRA adalah tahap pertama dari strategi AWS migrasi.

strategi migrasi

Pendekatan yang digunakan untuk memigrasikan beban kerja ke file. AWS Cloud Untuk informasi lebih lanjut, lihat entri 7 Rs di glosarium ini dan lihat Memobilisasi organisasi Anda untuk mempercepat migrasi skala besar.

ML

Lihat pembelajaran mesin.

modernisasi

Mengubah aplikasi usang (warisan atau monolitik) dan infrastrukturnya menjadi sistem yang gesit, elastis, dan sangat tersedia di cloud untuk mengurangi biaya, mendapatkan efisiensi, dan memanfaatkan inovasi. Untuk informasi selengkapnya, lihat Strategi untuk memodernisasi aplikasi di. AWS Cloud

penilaian kesiapan modernisasi

Evaluasi yang membantu menentukan kesiapan modernisasi aplikasi organisasi; mengidentifikasi manfaat, risiko, dan dependensi; dan menentukan seberapa baik organisasi dapat mendukung keadaan masa depan aplikasi tersebut. Hasil penilaian adalah cetak biru arsitektur target, peta jalan yang merinci fase pengembangan dan tonggak untuk proses modernisasi, dan rencana aksi untuk mengatasi kesenjangan yang diidentifikasi. Untuk informasi lebih lanjut, lihat Mengevaluasi kesiapan modernisasi untuk aplikasi di. AWS Cloud

aplikasi monolitik (monolit)

Aplikasi yang berjalan sebagai layanan tunggal dengan proses yang digabungkan secara ketat. Aplikasi monolitik memiliki beberapa kelemahan. Jika satu fitur aplikasi mengalami lonjakan permintaan, seluruh arsitektur harus diskalakan. Menambahkan atau meningkatkan fitur aplikasi monolitik juga menjadi lebih kompleks ketika basis kode tumbuh. Untuk mengatasi masalah ini,

Anda dapat menggunakan arsitektur microservices. Untuk informasi lebih lanjut, lihat Mengurai monolit menjadi layanan mikro.

MPA

Lihat Penilaian Portofolio Migrasi.

MQTT

Lihat Transportasi Telemetri Antrian Pesan.

klasifikasi multiclass

Sebuah proses yang membantu menghasilkan prediksi untuk beberapa kelas (memprediksi satu dari lebih dari dua hasil). Misalnya, model ML mungkin bertanya "Apakah produk ini buku, mobil, atau telepon?" atau "Kategori produk mana yang paling menarik bagi pelanggan ini?"

infrastruktur yang bisa berubah

Model yang memperbarui dan memodifikasi infrastruktur yang ada untuk beban kerja produksi. Untuk meningkatkan konsistensi, keandalan, dan prediktabilitas, AWS Well-Architected Framework merekomendasikan penggunaan infrastruktur yang tidak dapat diubah sebagai praktik terbaik.

O

OAC

Lihat kontrol akses asal.

OAI

Lihat identitas akses asal.

OCM

Lihat manajemen perubahan organisasi.

migrasi offline

Metode migrasi di mana beban kerja sumber diturunkan selama proses migrasi. Metode ini melibatkan waktu henti yang diperpanjang dan biasanya digunakan untuk beban kerja kecil dan tidak kritis.

OI

Lihat integrasi operasi.

OLA

Lihat perjanjian tingkat operasional.

migrasi online

Metode migrasi di mana beban kerja sumber disalin ke sistem target tanpa diambil offline. Aplikasi yang terhubung ke beban kerja dapat terus berfungsi selama migrasi. Metode ini melibatkan waktu henti nol hingga minimal dan biasanya digunakan untuk beban kerja produksi yang kritis.

OPC-UA

Lihat Komunikasi Proses Terbuka - Arsitektur Terpadu.

Komunikasi Proses Terbuka - Arsitektur Terpadu (OPC-UA)

Protokol komunikasi machine-to-machine (M2M) untuk otomasi industri. OPC-UA menyediakan standar interoperabilitas dengan enkripsi data, otentikasi, dan skema otorisasi.

perjanjian tingkat operasional (OLA)

Perjanjian yang menjelaskan apa yang dijanjikan kelompok TI fungsional untuk diberikan satu sama lain, untuk mendukung perjanjian tingkat layanan (SLA).

Tinjauan Kesiapan Operasional (ORR)

Daftar pertanyaan dan praktik terbaik terkait yang membantu Anda memahami, mengevaluasi, mencegah, atau mengurangi ruang lingkup insiden dan kemungkinan kegagalan. Untuk informasi lebih lanjut, lihat Ulasan Kesiapan Operasional (ORR) dalam Kerangka Kerja Well-Architected AWS .

teknologi operasional (OT)

Sistem perangkat keras dan perangkat lunak yang bekerja dengan lingkungan fisik untuk mengendalikan operasi industri, peralatan, dan infrastruktur. Di bidang manufaktur, integrasi sistem OT dan teknologi informasi (TI) adalah fokus utama untuk transformasi Industri 4.0.

integrasi operasi (OI)

Proses modernisasi operasi di cloud, yang melibatkan perencanaan kesiapan, otomatisasi, dan integrasi. Untuk informasi selengkapnya, lihat panduan integrasi operasi.

jejak organisasi

Jejak yang dibuat oleh AWS CloudTrail itu mencatat semua peristiwa untuk semua Akun AWS dalam organisasi di AWS Organizations. Jejak ini dibuat di setiap Akun AWS bagian organisasi dan melacak aktivitas di setiap akun. Untuk informasi selengkapnya, lihat Membuat jejak untuk organisasi dalam CloudTrail dokumentasi.

manajemen perubahan organisasi (OCM)

Kerangka kerja untuk mengelola transformasi bisnis utama yang mengganggu dari perspektif orang, budaya, dan kepemimpinan. OCM membantu organisasi mempersiapkan, dan transisi ke, sistem dan strategi baru dengan mempercepat adopsi perubahan, mengatasi masalah transisi, dan mendorong perubahan budaya dan organisasi. Dalam strategi AWS migrasi, kerangka kerja ini disebut percepatan orang, karena kecepatan perubahan yang diperlukan dalam proyek adopsi cloud. Untuk informasi lebih lanjut, lihat panduan OCM.

kontrol akses asal (OAC)

Di CloudFront, opsi yang disempurnakan untuk membatasi akses untuk mengamankan konten Amazon Simple Storage Service (Amazon S3) Anda. OAC mendukung semua bucket S3 di semua Wilayah AWS, enkripsi sisi server dengan AWS KMS (SSE-KMS), dan dinamis dan permintaan ke bucket S3. `PUT` `DELETE`

identitas akses asal (OAI)

Di CloudFront, opsi untuk membatasi akses untuk mengamankan konten Amazon S3 Anda. Saat Anda menggunakan OAI, CloudFront buat prinsipal yang dapat diautentikasi oleh Amazon S3. Prinsipal yang diautentikasi dapat mengakses konten dalam bucket S3 hanya melalui distribusi tertentu. CloudFront Lihat juga OAC, yang menyediakan kontrol akses yang lebih terperinci dan ditingkatkan.

ORR

Lihat tinjauan kesiapan operasional.

OT

Lihat teknologi operasional.

keluar (jalan keluar) VPC

Dalam arsitektur AWS multi-akun, VPC yang menangani koneksi jaringan yang dimulai dari dalam aplikasi. Arsitektur Referensi AWS Keamanan merekomendasikan pengaturan akun Jaringan

Anda dengan VPC masuk, keluar, dan inspeksi untuk melindungi antarmuka dua arah antara aplikasi Anda dan internet yang lebih luas.

P

batas izin

Kebijakan manajemen IAM yang dilampirkan pada prinsipal IAM untuk menetapkan izin maksimum yang dapat dimiliki pengguna atau peran. Untuk informasi selengkapnya, lihat Batas izin dalam dokumentasi IAM.

Informasi Identifikasi Pribadi (PII)

Informasi yang, jika dilihat secara langsung atau dipasangkan dengan data terkait lainnya, dapat digunakan untuk menyimpulkan identitas individu secara wajar. Contoh PII termasuk nama, alamat, dan informasi kontak.

PII

Lihat informasi yang dapat diidentifikasi secara pribadi.

buku pedoman

Serangkaian langkah yang telah ditentukan sebelumnya yang menangkap pekerjaan yang terkait dengan migrasi, seperti mengirimkan fungsi operasi inti di cloud. Buku pedoman dapat berupa skrip, runbook otomatis, atau ringkasan proses atau langkah-langkah yang diperlukan untuk mengoperasikan lingkungan modern Anda.

PLC

Lihat pengontrol logika yang dapat diprogram.

PLM

Lihat manajemen siklus hidup produk.

kebijakan

Objek yang dapat menentukan izin (lihat kebijakan berbasis identitas), menentukan kondisi akses (lihat kebijakanberbasis sumber daya), atau menentukan izin maksimum untuk semua akun dalam organisasi di (lihat kebijakan kontrol layanan). AWS Organizations

persistensi poliglot

Secara independen memilih teknologi penyimpanan data microservice berdasarkan pola akses data dan persyaratan lainnya. Jika layanan mikro Anda memiliki teknologi penyimpanan data yang sama, mereka dapat menghadapi tantangan implementasi atau mengalami kinerja yang buruk. Layanan mikro lebih mudah diimplementasikan dan mencapai kinerja dan skalabilitas yang lebih baik jika mereka menggunakan penyimpanan data yang paling sesuai dengan kebutuhan mereka. Untuk informasi selengkapnya, lihat Mengaktifkan persistensi data di layanan mikro.

penilaian portofolio

Proses menemukan, menganalisis, dan memprioritaskan portofolio aplikasi untuk merencanakan migrasi. Untuk informasi selengkapnya, lihat Mengevaluasi kesiapan migrasi.

predikat

Kondisi kueri yang mengembalikan `true` atau`false`, biasanya terletak di `WHERE` klausa.

predikat pushdown

Teknik pengoptimalan kueri database yang menyaring data dalam kueri sebelum transfer. Ini mengurangi jumlah data yang harus diambil dan diproses dari database relasional, dan meningkatkan kinerja kueri.

kontrol preventif

Kontrol keamanan yang dirancang untuk mencegah suatu peristiwa terjadi. Kontrol ini adalah garis pertahanan pertama untuk membantu mencegah akses tidak sah atau perubahan yang tidak diinginkan ke jaringan Anda. Untuk informasi selengkapnya, lihat Kontrol pencegahan dalam Menerapkan kontrol keamanan pada. AWS

principal

Entitas AWS yang dapat melakukan tindakan dan mengakses sumber daya. Entitas ini biasanya merupakan pengguna root untuk Akun AWS, peran IAM, atau pengguna. Untuk informasi selengkapnya, lihat Prinsip dalam istilah dan konsep Peran dalam dokumentasi IAM.

Privasi oleh Desain

Pendekatan dalam rekayasa sistem yang memperhitungkan privasi di seluruh proses rekayasa.

zona host pribadi

Container yang menyimpan informasi tentang bagaimana Anda ingin Amazon Route 53 merespons kueri DNS untuk domain dan subdomainnya dalam satu atau beberapa VPC. Untuk

informasi selengkapnya, lihat Bekerja dengan zona yang dihosting pribadi di dokumentasi Route 53.

kontrol proaktif

Kontrol keamanan yang dirancang untuk mencegah penyebaran sumber daya yang tidak sesuai. Kontrol ini memindai sumber daya sebelum disediakan. Jika sumber daya tidak sesuai dengan kontrol, maka itu tidak disediakan. Untuk informasi selengkapnya, lihat panduan referensi Kontrol dalam AWS Control Tower dokumentasi dan lihat Kontrol proaktif dalam Menerapkan kontrol keamanan pada AWS.

manajemen siklus hidup produk (PLM)

Manajemen data dan proses untuk suatu produk di seluruh siklus hidupnya, mulai dari desain, pengembangan, dan peluncuran, melalui pertumbuhan dan kematangan, hingga penurunan dan penghapusan.

lingkungan produksi

Lihat lingkungan.

pengontrol logika yang dapat diprogram (PLC)

Di bidang manufaktur, komputer yang sangat andal dan mudah beradaptasi yang memantau mesin dan mengotomatiskan proses manufaktur.

pseudonimisasi

Proses penggantian pengidentifikasi pribadi dalam kumpulan data dengan nilai placeholder. Pseudonimisasi dapat membantu melindungi privasi pribadi. Data pseudonim masih dianggap sebagai data pribadi.

terbitkan/berlangganan (pub/sub)

Pola yang memungkinkan komunikasi asinkron antara layanan mikro untuk meningkatkan skalabilitas dan daya tanggap. Misalnya, dalam MES berbasis layanan mikro, layanan mikro dapat mempublikasikan pesan peristiwa ke saluran yang dapat berlangganan layanan mikro lainnya. Sistem dapat menambahkan layanan mikro baru tanpa mengubah layanan penerbitan.

Q

rencana kueri

Serangkaian langkah, seperti instruksi, yang digunakan untuk mengakses data dalam sistem database relasional SQL.

regresi rencana kueri

Ketika pengoptimal layanan database memilih rencana yang kurang optimal daripada sebelum perubahan yang diberikan ke lingkungan database. Hal ini dapat disebabkan oleh perubahan statistik, kendala, pengaturan lingkungan, pengikatan parameter kueri, dan pembaruan ke mesin database.

R

Matriks RACI

Lihat bertanggung jawab, akuntabel, dikonsultasikan, diinformasikan (RACI).

ransomware

Perangkat lunak berbahaya yang dirancang untuk memblokir akses ke sistem komputer atau data sampai pembayaran dilakukan.

Matriks RASCI

Lihat bertanggung jawab, akuntabel, dikonsultasikan, diinformasikan (RACI).

RCAC

Lihat kontrol akses baris dan kolom.

replika baca

Salinan database yang digunakan untuk tujuan read-only. Anda dapat merutekan kueri ke replika baca untuk mengurangi beban pada database utama Anda.

arsitek ulang

Lihat 7 Rs.

tujuan titik pemulihan (RPO)

Jumlah waktu maksimum yang dapat diterima sejak titik pemulihan data terakhir. Ini menentukan apa yang dianggap sebagai hilangnya data yang dapat diterima antara titik pemulihan terakhir dan gangguan layanan.

tujuan waktu pemulihan (RTO)

Penundaan maksimum yang dapat diterima antara gangguan layanan dan pemulihan layanan.

refactor

Lihat 7 Rs.

Wilayah

Kumpulan AWS sumber daya di wilayah geografis. Masing-masing Wilayah AWS terisolasi dan independen dari yang lain untuk memberikan toleransi kesalahan, stabilitas, dan ketahanan. Untuk informasi selengkapnya, lihat Menentukan Wilayah AWS akun yang dapat digunakan.

regresi

Teknik ML yang memprediksi nilai numerik. Misalnya, untuk memecahkan masalah “Berapa harga rumah ini akan dijual?” Model ML dapat menggunakan model regresi linier untuk memprediksi harga jual rumah berdasarkan fakta yang diketahui tentang rumah (misalnya, luas persegi).

rehost

Lihat 7 Rs.

melepaskan

Dalam proses penyebaran, tindakan mempromosikan perubahan pada lingkungan produksi.

memindahkan

Lihat 7 Rs.

memplatform ulang

Lihat 7 Rs.

pembelian kembali

Lihat 7 Rs.

ketahanan

Kemampuan aplikasi untuk melawan atau pulih dari gangguan. Ketersediaan tinggi dan pemulihan bencana adalah pertimbangan umum ketika merencanakan ketahanan di. AWS Cloud Untuk informasi lebih lanjut, lihat AWS Cloud Ketahanan.

kebijakan berbasis sumber daya

Kebijakan yang dilampirkan ke sumber daya, seperti bucket Amazon S3, titik akhir, atau kunci enkripsi. Jenis kebijakan ini menentukan prinsip mana yang diizinkan mengakses, tindakan yang didukung, dan kondisi lain yang harus dipenuhi.

matriks yang bertanggung jawab, akuntabel, dikonsultasikan, diinformasikan (RACI)

Matriks yang mendefinisikan peran dan tanggung jawab untuk semua pihak yang terlibat dalam kegiatan migrasi dan operasi cloud. Nama matriks berasal dari jenis tanggung jawab yang didefinisikan dalam matriks: bertanggung jawab (R), akuntabel (A), dikonsultasikan (C), dan diinformasikan (I). Jenis dukungan (S) adalah opsional. Jika Anda menyertakan dukungan, matriks disebut matriks RASCI, dan jika Anda mengecualikannya, itu disebut matriks RACI.

kontrol responsif

Kontrol keamanan yang dirancang untuk mendorong remediasi efek samping atau penyimpangan dari garis dasar keamanan Anda. Untuk informasi selengkapnya, lihat Kontrol responsif dalam Menerapkan kontrol keamanan pada AWS.

melestarikan

Lihat 7 Rs.

pensiun

Lihat 7 Rs.

rotasi

Proses memperbarui rahasia secara berkala untuk membuatnya lebih sulit bagi penyerang untuk mengakses kredensil.

kontrol akses baris dan kolom (RCAC)

Penggunaan ekspresi SQL dasar dan fleksibel yang telah menetapkan aturan akses. RCAC terdiri dari izin baris dan topeng kolom.

RPO

Lihat tujuan titik pemulihan.

RTO

Lihat tujuan waktu pemulihan.

buku runbook

Satu set prosedur manual atau otomatis yang diperlukan untuk melakukan tugas tertentu. Ini biasanya dibangun untuk merampingkan operasi berulang atau prosedur dengan tingkat kesalahan yang tinggi.

D

SAML 2.0

Standar terbuka yang digunakan oleh banyak penyedia identitas (IdPs). Fitur ini memungkinkan sistem masuk tunggal gabungan (SSO), sehingga pengguna dapat masuk ke AWS Management Console atau memanggil operasi AWS API tanpa Anda harus membuat pengguna di IAM untuk semua orang di organisasi Anda. Untuk informasi lebih lanjut tentang federasi berbasis SAMP 2.0, lihat Tentang federasi berbasis SAMP 2.0 dalam dokumentasi IAM.

PENIPUAN

Lihat kontrol pengawasan dan akuisisi data.

SCP

Lihat kebijakan kontrol layanan.

Rahasia

Dalam AWS Secrets Manager, informasi rahasia atau terbatas, seperti kata sandi atau kredensi pengguna, yang Anda simpan dalam bentuk terenkripsi. Ini terdiri dari nilai rahasia dan metadata-nya. Nilai rahasia dapat berupa biner, string tunggal, atau beberapa string. Untuk informasi selengkapnya, lihat Apa yang ada di rahasia Secrets Manager? dalam dokumentasi Secrets Manager.

kontrol keamanan

Pagar pembatas teknis atau administratif yang mencegah, mendeteksi, atau mengurangi kemampuan pelaku ancaman untuk mengeksploitasi kerentanan keamanan. Ada empat jenis kontrol keamanan utama: preventif, detektif, responsif, dan proaktif.

pengerasan keamanan

Proses mengurangi permukaan serangan untuk membuatnya lebih tahan terhadap serangan. Ini dapat mencakup tindakan seperti menghapus sumber daya yang tidak lagi diperlukan, menerapkan praktik keamanan terbaik untuk memberikan hak istimewa paling sedikit, atau menonaktifkan fitur yang tidak perlu dalam file konfigurasi.

sistem informasi keamanan dan manajemen acara (SIEM)

Alat dan layanan yang menggabungkan sistem manajemen informasi keamanan (SIM) dan manajemen acara keamanan (SEM). Sistem SIEM mengumpulkan, memantau, dan menganalisis data dari server, jaringan, perangkat, dan sumber lain untuk mendeteksi ancaman dan pelanggaran keamanan, dan untuk menghasilkan peringatan.

otomatisasi respons keamanan

Tindakan yang telah ditentukan dan diprogram yang dirancang untuk secara otomatis merespons atau memulihkan peristiwa keamanan. Otomatisasi ini berfungsi sebagai kontrol keamanan detektif atau responsif yang membantu Anda menerapkan praktik terbaik AWS keamanan. Contoh tindakan respons otomatis termasuk memodifikasi grup keamanan VPC, menambal instans Amazon EC2, atau memutar kredensil.

enkripsi sisi server

Enkripsi data di tujuannya, oleh layanan AWS yang menerimanya.

kebijakan kontrol layanan (SCP)

Kebijakan yang menyediakan kontrol terpusat atas izin untuk semua akun di organisasi. AWS Organizations SCP menentukan pagar pembatas atau menetapkan batasan pada tindakan yang dapat didelegasikan oleh administrator kepada pengguna atau peran. Anda dapat menggunakan SCP sebagai daftar izin atau daftar penolakan, untuk menentukan layanan atau tindakan mana yang diizinkan atau dilarang. Untuk informasi selengkapnya, lihat Kebijakan kontrol layanan dalam AWS Organizations dokumentasi.

titik akhir layanan

URL titik masuk untuk file layanan AWS. Anda dapat menggunakan endpoint untuk terhubung secara terprogram ke layanan target. Untuk informasi selengkapnya, lihat layanan AWS titik akhir di Referensi Umum AWS.

perjanjian tingkat layanan (SLA)

Perjanjian yang menjelaskan apa yang dijanjikan tim TI untuk diberikan kepada pelanggan mereka, seperti waktu kerja dan kinerja layanan.

indikator tingkat layanan (SLI)

Pengukuran aspek kinerja layanan, seperti tingkat kesalahan, ketersediaan, atau throughputnya.

tujuan tingkat layanan (SLO)

Metrik target yang mewakili kesehatan layanan, yang diukur dengan indikator tingkat layanan.

model tanggung jawab bersama

Model yang menjelaskan tanggung jawab yang Anda bagikan AWS untuk keamanan dan kepatuhan cloud. AWS bertanggung jawab atas keamanan cloud, sedangkan Anda bertanggung jawab atas keamanan di cloud. Untuk informasi selengkapnya, lihat Model tanggung jawab bersama.

SIEM

Lihat informasi keamanan dan sistem manajemen acara.

titik kegagalan tunggal (SPOF)

Kegagalan dalam satu komponen penting dari aplikasi yang dapat mengganggu sistem.

SLA

Lihat perjanjian tingkat layanan.

SLI

Lihat indikator tingkat layanan.

SLO

Lihat tujuan tingkat layanan.

split-and-seed model

Pola untuk menskalakan dan mempercepat proyek modernisasi. Ketika fitur baru dan rilis produk didefinisikan, tim inti berpisah untuk membuat tim produk baru. Ini membantu meningkatkan kemampuan dan layanan organisasi Anda, meningkatkan produktivitas pengembang, dan

mendukung inovasi yang cepat. Untuk informasi lebih lanjut, lihat Pendekatan bertahap untuk memodernisasi aplikasi di. AWS Cloud

SPOF

Lihat satu titik kegagalan.

skema bintang

Struktur organisasi database yang menggunakan satu tabel fakta besar untuk menyimpan data transaksional atau terukur dan menggunakan satu atau lebih tabel dimensi yang lebih kecil untuk menyimpan atribut data. Struktur ini dirancang untuk digunakan dalam gudang data atau untuk tujuan intelijen bisnis.

pola ara pencekik

Pendekatan untuk memodernisasi sistem monolitik dengan menulis ulang secara bertahap dan mengganti fungsionalitas sistem sampai sistem warisan dapat dinonaktifkan. Pola ini menggunakan analogi pohon ara yang tumbuh menjadi pohon yang sudah mapan dan akhirnya mengatasi dan menggantikan inangnya. Pola ini diperkenalkan oleh Martin Fowler sebagai cara untuk mengelola risiko saat menulis ulang sistem monolitik. Untuk contoh cara menerapkan pola ini, lihat Memodernisasi layanan web Microsoft ASP.NET (ASMX) lama secara bertahap menggunakan container dan Amazon API Gateway.

subnet

Rentang alamat IP dalam VPC Anda. Subnet harus berada di Availability Zone tunggal.

kontrol pengawasan dan akuisisi data (SCADA)

Di bidang manufaktur, sistem yang menggunakan perangkat keras dan perangkat lunak untuk memantau aset fisik dan operasi produksi.

enkripsi simetris

Algoritma enkripsi yang menggunakan kunci yang sama untuk mengenkripsi dan mendekripsi data.

pengujian sintetis

Menguji sistem dengan cara yang mensimulasikan interaksi pengguna untuk mendeteksi potensi masalah atau untuk memantau kinerja. Anda dapat menggunakan Amazon CloudWatch Synthetics untuk membuat tes ini.

T

tag

Pasangan nilai kunci yang bertindak sebagai metadata untuk mengatur sumber daya Anda. AWS Tanda dapat membantu Anda mengelola, mengidentifikasi, mengatur, dan memfilter sumber daya. Untuk informasi selengkapnya, lihat Menandai AWS sumber daya Anda.

variabel target

Nilai yang Anda coba prediksi dalam ML yang diawasi. Ini juga disebut sebagai variabel hasil. Misalnya, dalam pengaturan manufaktur, variabel target bisa menjadi cacat produk.

daftar tugas

Alat yang digunakan untuk melacak kemajuan melalui runbook. Daftar tugas berisi ikhtisar runbook dan daftar tugas umum yang harus diselesaikan. Untuk setiap tugas umum, itu termasuk perkiraan jumlah waktu yang dibutuhkan, pemilik, dan kemajuan.

lingkungan uji

Lihat lingkungan.

pelatihan

Untuk menyediakan data bagi model ML Anda untuk dipelajari. Data pelatihan harus berisi jawaban yang benar. Algoritma pembelajaran menemukan pola dalam data pelatihan yang memetakan atribut data input ke target (jawaban yang ingin Anda prediksi). Ini menghasilkan model ML yang menangkap pola-pola ini. Anda kemudian dapat menggunakan model ML untuk membuat prediksi pada data baru yang Anda tidak tahu targetnya.

gerbang transit

Hub transit jaringan yang dapat Anda gunakan untuk menghubungkan VPC dan jaringan lokal Anda. Untuk informasi selengkapnya, lihat Apa itu gateway transit dalam AWS Transit Gateway dokumentasi.

alur kerja berbasis batang

Pendekatan di mana pengembang membangun dan menguji fitur secara lokal di cabang fitur dan kemudian menggabungkan perubahan tersebut ke cabang utama. Cabang utama kemudian dibangun untuk pengembangan, praproduksi, dan lingkungan produksi, secara berurutan.

akses tepercaya

Memberikan izin ke layanan yang Anda tentukan untuk melakukan tugas di organisasi Anda di dalam AWS Organizations dan di akunnya atas nama Anda. Layanan tepercaya menciptakan peran terkait layanan di setiap akun, ketika peran itu diperlukan, untuk melakukan tugas manajemen untuk Anda. Untuk informasi selengkapnya, lihat Menggunakan AWS Organizations dengan AWS layanan lain dalam AWS Organizations dokumentasi.

penyetelan

Untuk mengubah aspek proses pelatihan Anda untuk meningkatkan akurasi model ML. Misalnya, Anda dapat melatih model ML dengan membuat set pelabelan, menambahkan label, dan kemudian mengulangi langkah-langkah ini beberapa kali di bawah pengaturan yang berbeda untuk mengoptimalkan model.

tim dua pizza

Sebuah DevOps tim kecil yang bisa Anda beri makan dengan dua pizza. Ukuran tim dua pizza memastikan peluang terbaik untuk berkolaborasi dalam pengembangan perangkat lunak.

U

waswas

Sebuah konsep yang mengacu pada informasi yang tidak tepat, tidak lengkap, atau tidak diketahui yang dapat merusak keandalan model ML prediktif. Ada dua jenis ketidakpastian: ketidakpastian epistemik disebabkan oleh data yang terbatas dan tidak lengkap, sedangkan ketidakpastian aleatorik disebabkan oleh kebisingan dan keacakan yang melekat dalam data. Untuk informasi lebih lanjut, lihat panduan Mengukur ketidakpastian dalam sistem pembelajaran mendalam.

tugas yang tidak terdiferensiasi

Juga dikenal sebagai angkat berat, pekerjaan yang diperlukan untuk membuat dan mengoperasikan aplikasi tetapi itu tidak memberikan nilai langsung kepada pengguna akhir atau memberikan keunggulan kompetitif. Contoh tugas yang tidak terdiferensiasi termasuk pengadaan, pemeliharaan, dan perencanaan kapasitas.

lingkungan atas

Lihat lingkungan.

V

menyedot debu

Operasi pemeliharaan database yang melibatkan pembersihan setelah pembaruan tambahan untuk merebut kembali penyimpanan dan meningkatkan kinerja.

kendali versi

Proses dan alat yang melacak perubahan, seperti perubahan kode sumber dalam repositori.

Peering VPC

Koneksi antara dua VPC yang memungkinkan Anda merutekan lalu lintas dengan menggunakan alamat IP pribadi. Untuk informasi selengkapnya, lihat Apa itu peering VPC di dokumentasi VPC Amazon.

kerentanan

Kelemahan perangkat lunak atau perangkat keras yang membahayakan keamanan sistem.

W

cache hangat

Cache buffer yang berisi data terkini dan relevan yang sering diakses. Instance database dapat membaca dari cache buffer, yang lebih cepat daripada membaca dari memori utama atau disk.

data hangat

Data yang jarang diakses. Saat menanyakan jenis data ini, kueri yang cukup lambat biasanya dapat diterima.

fungsi jendela

Fungsi SQL yang melakukan perhitungan pada sekelompok baris yang berhubungan dengan catatan saat ini. Fungsi jendela berguna untuk memproses tugas, seperti menghitung rata-rata bergerak atau mengakses nilai baris berdasarkan posisi relatif dari baris saat ini.

beban kerja

Kumpulan sumber daya dan kode yang memberikan nilai bisnis, seperti aplikasi yang dihadapi pelanggan atau proses backend.

aliran kerja

Grup fungsional dalam proyek migrasi yang bertanggung jawab atas serangkaian tugas tertentu. Setiap alur kerja independen tetapi mendukung alur kerja lain dalam proyek. Misalnya, alur kerja portofolio bertanggung jawab untuk memprioritaskan aplikasi, perencanaan gelombang, dan mengumpulkan metadata migrasi. Alur kerja portofolio mengirimkan aset ini ke alur kerja migrasi, yang kemudian memigrasikan server dan aplikasi.

CACING

Lihat menulis sekali, baca banyak.

WQF

Lihat Kerangka Kualifikasi Beban Kerja AWS.

tulis sekali, baca banyak (WORM)

Model penyimpanan yang menulis data satu kali dan mencegah data dihapus atau dimodifikasi. Pengguna yang berwenang dapat membaca data sebanyak yang diperlukan, tetapi mereka tidak dapat mengubahnya. Infrastruktur penyimpanan data ini dianggap tidak dapat diubah.

Z

eksploitasi zero-day

Serangan, biasanya malware, yang memanfaatkan kerentanan zero-day.

kerentanan zero-day

Cacat atau kerentanan yang tak tanggung-tanggung dalam sistem produksi. Aktor ancaman dapat menggunakan jenis kerentanan ini untuk menyerang sistem. Pengembang sering menyadari kerentanan sebagai akibat dari serangan tersebut.

aplikasi zombie

Aplikasi yang memiliki CPU rata-rata dan penggunaan memori di bawah 5 persen. Dalam proyek migrasi, adalah umum untuk menghentikan aplikasi ini.

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