Melhores práticas para ajuste de desempenho AWS Glue para tarefas do Apache Spark

: Melhores práticas para ajuste de desempenho AWS Glue para tarefas do Apache Spark

Table of Contents

Práticas recomendadas para ajuste de desempenho AWS Glue para tarefas do Apache Spark

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

Dezembro de 2023 (histórico do documento)

AWS Glue fornece opções diferentes para ajustar o desempenho. Este guia define os principais tópicos AWS Glue para ajuste do Apache Spark. Em seguida, ele fornece uma estratégia básica para você seguir ao ajustá-las AWS Glue para tarefas do Apache Spark. Use este guia para saber como identificar problemas de performance interpretando as métricas disponíveis no AWS Glue. Depois, incorpore estratégias para resolver esses problemas, maximizando a performance e minimizando os custos.

Este guia aborda as seguintes práticas de ajuste:

- Escalar a capacidade do cluster
- Use a AWS Glue versão mais recente
- Reduzir a quantidade de dados verificados
- Paralelizar tarefas
- Minimizar a sobrecarga de planejamento
- Otimizar as operações de shuffle
- Otimizar as funções definidas pelo usuário

Principais tópicos no Apache Spark

Esta seção explica os conceitos básicos do Apache Spark e os principais tópicos para ajuste da performance do AWS Glue para Apache Spark. É importante entender esses conceitos e tópicos antes de analisar estratégias de ajuste no mundo real.

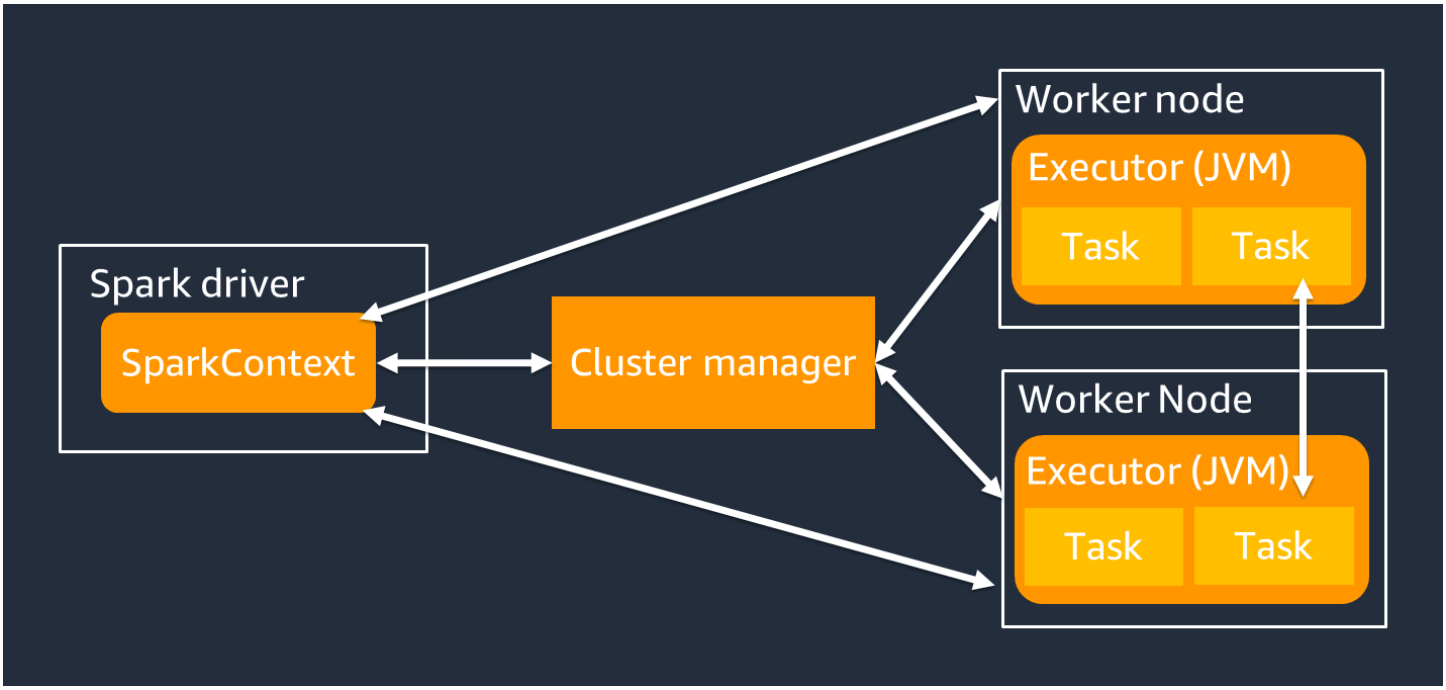
Arquitetura

O driver do Spark é o principal responsável por dividir sua aplicação do Spark em tarefas que podem ser realizadas por operadores individuais. O driver do Spark tem as seguintes responsabilidades:

- Executar `main()` em seu código
- Gerar planos de execução
- Provisionar executores do Spark em conjunto com o gerenciador de clusters, que gerencia os recursos no cluster
- Agendar e solicitar tarefas para os executores do Spark
- Gerenciar o progresso e a recuperação de tarefas

Você usa um objeto `SparkContext` para interagir com o driver do Spark na execução do seu trabalho.

Um executor do Spark é um operador para armazenar dados e executar tarefas que são passadas pelo driver do Spark. O número de executores do Spark aumentará e diminuirá com o tamanho do seu cluster.

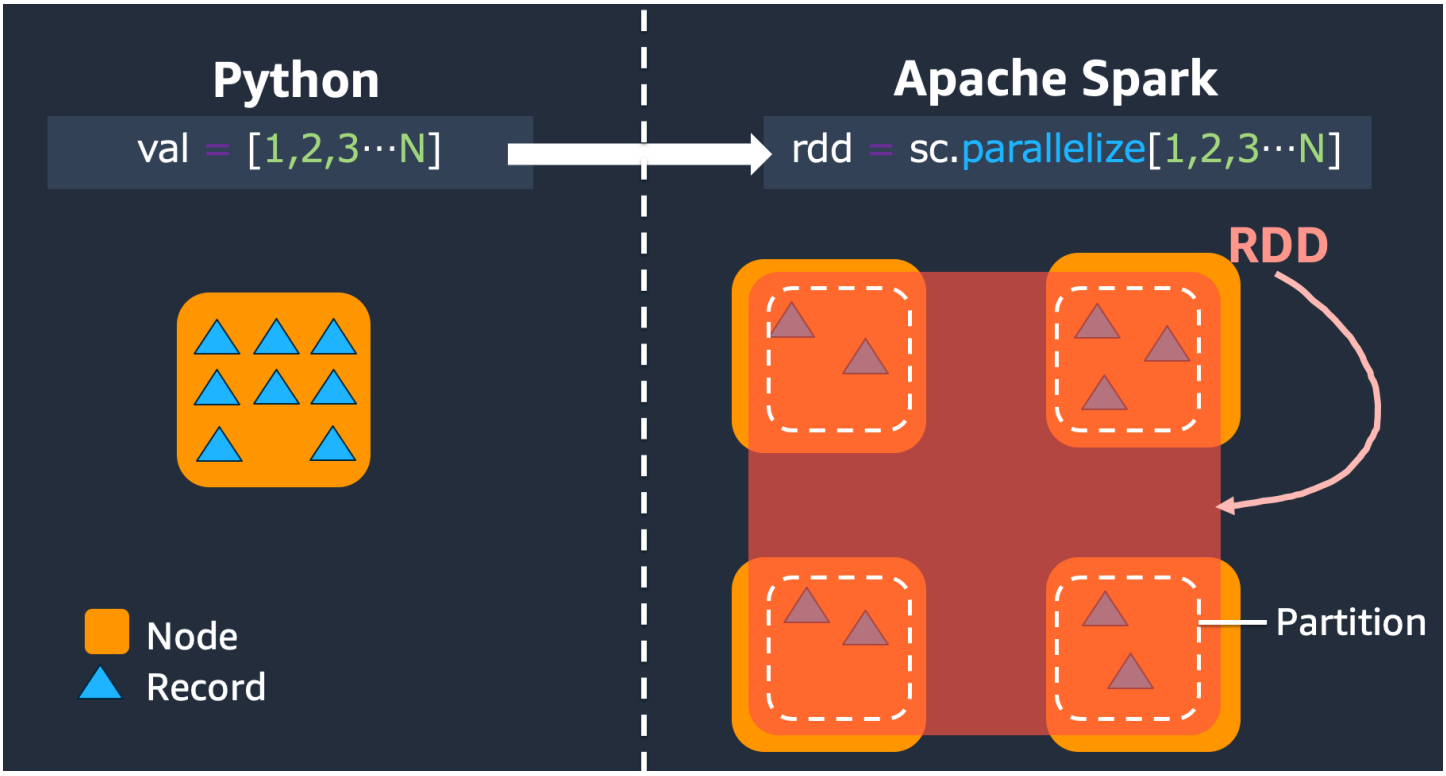


> **Note**
>
> Um executor do Spark tem vários slots para que várias tarefas sejam processadas paralelamente. Por padrão, o Spark é compatível com uma tarefa para cada núcleo de CPU virtual (vCPU). Por exemplo, se um executor tiver quatro núcleos de CPU, ele poderá executar quatro tarefas simultâneas.

Conjunto de dados distribuído resiliente

O Spark realiza o trabalho complexo de armazenar e rastrear grandes conjuntos de dados nos executores do Spark. Ao escrever código para trabalhos do Spark, você não precisa pensar nos detalhes do armazenamento. O Spark fornece a abstração resiliente de conjunto de dados distribuído (RDD), que é uma coleção de elementos que podem ser operados paralelamente e particionados nos executores do Spark do cluster.

A figura a seguir mostra a diferença em como armazenar dados na memória quando um script Python é executado em seu ambiente típico e quando é executado no framework do Spark (PySpark).



- Python: escrever `val = [1,2,3...N]` em um script Python mantém os dados na memória na única máquina em que o código está sendo executado.
- PySpark: o Spark fornece a estrutura de dados RDD para carregar e processar dados distribuídos na memória em vários executores do Spark. Você pode gerar um RDD com código como `rdd = sc.parallelize[1,2,3...N]`, e o Spark pode distribuir e armazenar dados automaticamente na memória em vários executores do Spark.

  Em muitos trabalhos do AWS Glue, você usa RDDs por meio do AWS Glue DynamicFrames e do Spark DataFrames. Estas são as abstrações que permitem definir o esquema de dados em um RDD e realizar tarefas de alto nível com as informações adicionais. Como elas usam RDDs internamente, os dados são distribuídos de forma transparente e carregados em vários nós no seguinte código:

  - 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>")
```

Um RDD tem os seguintes recursos:

- Os RDDs consistem em dados divididos em várias partes chamadas partições. Cada executor do Spark armazena uma ou mais partições na memória, e os dados são distribuídos em vários executores.
- Os RDDs são imutáveis, o que significa que não podem ser alterados após serem criados. Para alterar um DataFrame, você pode usar transformações, que são definidas na seção a seguir.
- Os RDDs replicam dados nos nós disponíveis para que possam se recuperar automaticamente de falhas nos nós.

Avaliação lenta

Os RDDs são compatíveis com dois tipos de operações: transformações, que criam um novo conjunto de dados de um existente, e ações, que retornam um valor ao programa do driver após executar um cálculo no conjunto de dados.

- Transformações: como os RDDs são imutáveis, você só pode alterá-los usando uma transformação.

  Por exemplo, `map` é uma transformação que passa cada elemento do conjunto de dados por meio de uma função e retorna um novo RDD representando os resultados. Observe que o método `map` não retorna uma saída. O Spark armazena a transformação abstrata para o futuro, em vez de permitir que você interaja com o resultado. O Spark não atuará nas transformações até que você chame uma ação.
- Ações: usando transformações, você cria seu plano lógico de transformação. Para iniciar o cálculo, você executa uma ação como `write`, `count`, `show` ou `collect`.

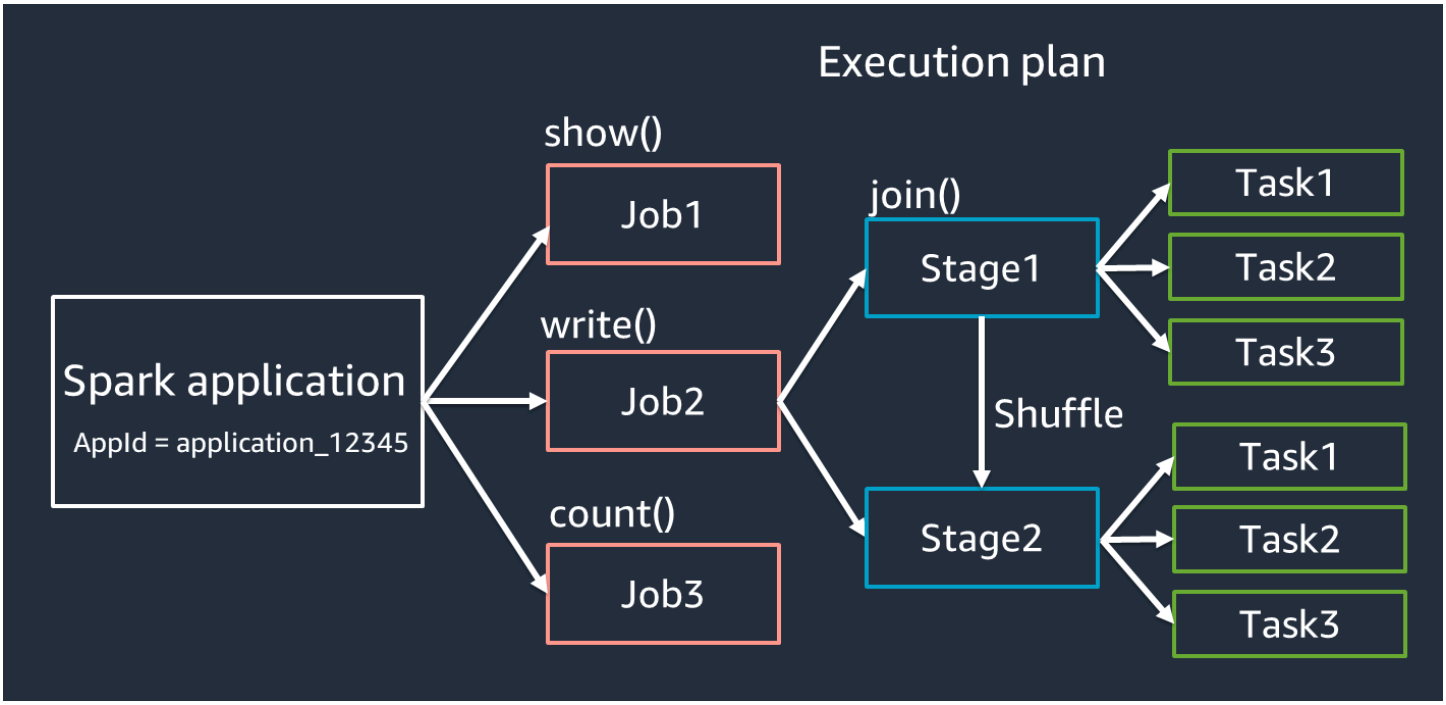
  Todas as transformações no Spark são lentas, pois não computam seus resultados imediatamente. Em vez disso, o Spark se lembra de uma série de transformações aplicadas a alguns conjuntos de dados básicos, como objetos do Amazon Simple Storage Service (Amazon S3). As transformações são calculadas somente quando uma ação exige que um resultado seja

retornado ao driver. Esse design permite que o Spark seja executado com mais eficiência. Por exemplo, considere a situação em que um conjunto de dados criado por meio da transformação `map` é consumido somente por uma transformação que reduz substancialmente o número de linhas, como `reduce`. Você pode então passar o conjunto de dados menor que passou por ambas as transformações para o driver, em vez de passar o conjunto de dados mapeado maior.

Terminologia das aplicações do Spark

Esta seção aborda a terminologia das aplicações do Spark. O driver do Spark cria um plano de execução e controla o comportamento das aplicações em várias abstrações. Os termos a seguir são importantes para o desenvolvimento, a depuração e o ajuste de performance com a interface do usuário do Spark.

- Aplicação: baseada em uma sessão do Spark (contexto do Spark). Identificada por um ID exclusivo, como `<application_XXX>`.
- Trabalhos: baseados nas ações criadas para um RDD. Um trabalho consiste em uma ou mais etapas.
- Etapas: baseadas nos shuffles criados para um RDD. Uma etapa consiste em uma ou mais tarefas. O shuffle é o mecanismo do Spark para redistribuir dados para que sejam agrupados de forma diferente nas partições do RDD. Certas transformações, como `join()`, exigem um shuffle. As operações de shuffle são analisadas mais detalhadamente na prática de ajuste Otimizar as operações de shuffle.
- Tarefas: uma tarefa é a unidade mínima de processamento programada pelo Spark. As tarefas são criadas para cada partição do RDD, e o número de tarefas é o número máximo de execuções simultâneas na etapa.

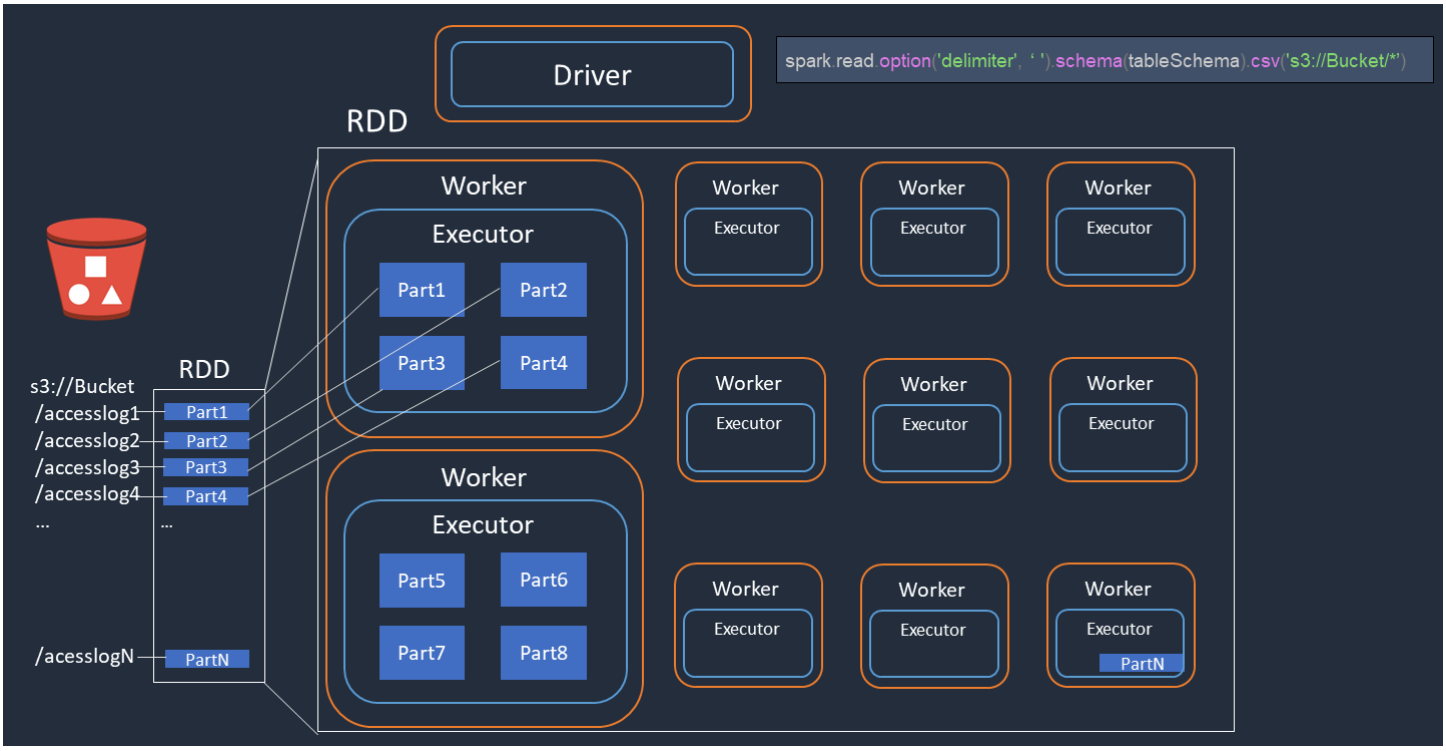


> **Note**
>
> As tarefas são a coisa mais importante a se considerar ao otimizar o paralelismo. O número de tarefas aumenta com o número do RDD

Paralelismo

O Spark paraleliza tarefas para carregar e transformar dados.

Considere um exemplo em que você executa o processamento distribuído de arquivos de logs de acesso (denominados `accesslog1 ... accesslogN`) no Amazon S3. O diagrama a seguir mostra o fluxo de processamento distribuído.

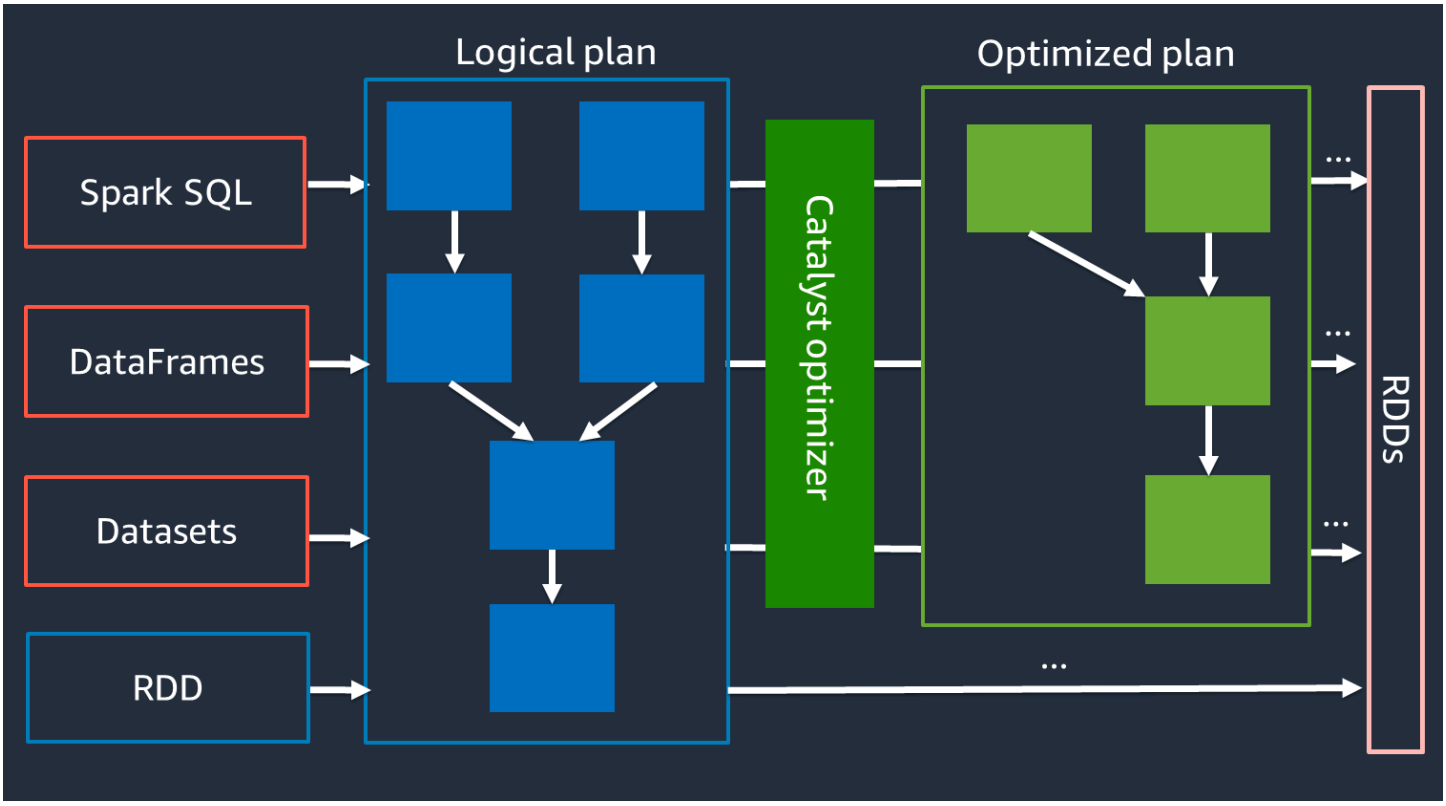


1. O driver do Spark cria um plano de execução para processamento distribuído nos vários executores do Spark.
2. O driver do Spark atribui tarefas a cada executor com base no plano de execução. Por padrão, o driver do Spark cria partições do RDD (cada uma correspondendo a uma tarefa do Spark) para cada objeto do S3 (`Part1 ... N`). Em seguida, o driver do Spark atribui tarefas a cada executor.
3. Cada tarefa do Spark baixa seu objeto do S3 atribuído e o armazena na memória na partição do RDD. Dessa forma, vários executores do Spark baixam e processam paralelamente a tarefa atribuída.

Para obter mais detalhes sobre o número inicial de partições e a otimização, consulte a seção Paralelizar tarefas.

Otimizador Catalyst

Internamente, o Spark usa um mecanismo chamado Otimizador Catalyst para otimizar os planos de execução. O Catalyst tem um otimizador de consultas que você pode usar ao executar APIs de alto nível do Spark, como Spark SQL, DataFrame e Datasets, conforme descrito no diagrama a seguir.



Como o otimizador Catalyst não funciona diretamente com a API RDD, as APIs de alto nível geralmente são mais rápidas do que a API RDD de baixo nível. Para junções complexas, o otimizador Catalyst pode melhorar significativamente a performance otimizando o plano de execução do trabalho. Você pode ver o plano otimizado do seu trabalho do Spark na guia SQL da interface do usuário do Spark.

Execução Adaptativa de Consultas

O otimizador Catalyst executa a otimização do runtime por meio de um processo chamado Execução Adaptativa de Consultas. A Execução Adaptativa de Consultas usa estatísticas de runtime para otimizar novamente o plano de execução das consultas enquanto seu trabalho está em execução. A Execução Adaptativa de Consultas oferece várias soluções para os desafios de performance, incluindo a aglutinação de partições pós-shuffle, a conversão da junção sort-merge em junção broadcast e a otimização da junção skew, conforme descrito nas seções a seguir.

A Execução Adaptativa de Consultas está disponível no AWS Glue versão 3.0 e versões posteriores, e está habilitada por padrão no AWS Glue versão 4.0 (Spark 3.3.0) e versões posteriores. A Execução Adaptativa de Consultas pode ser ativada e desativada usando `spark.conf.set("spark.sql.adaptive.enabled", "true")` em seu código.

Aglutinação de partições pós-shuffle

Esse recurso reduz as partições do RDD (aglutina) após cada shuffle com base nas estatísticas de saída de `map`. Isso simplifica o ajuste do número de partições de shuffle ao executar consultas. Você não precisa definir um número de partições de shuffle para o seu conjunto de dados. O Spark pode escolher o número adequado de partições de shuffle no runtime depois que você tiver uma quantidade inicial adequada de partições de shuffle.

A aglutinação de partições pós-shuffle é habilitada quando `spark.sql.adaptive.enabled` e `spark.sql.adaptive.coalescePartitions.enabled` estão definidas como true. Para obter mais informações, consulte a documentação do Apache Spark.

Conversão da junção soft-merge em junção broadcast

Esse recurso reconhece quando você está juntando dois conjuntos de dados de tamanhos substancialmente diferentes e adota um algoritmo de junção mais eficiente com base nessas informações. Para obter mais detalhes, consulte a documentação do Apache Spark. As estratégias de junção são discutidas na seção Otimizar as operações de shuffle.

Otimização de junções skew

A distorção de dados é um dos gargalos mais comuns dos trabalhos do Spark. É uma situação em que os dados são distorcidos para partições específicas do RDD (e, consequentemente, tarefas específicas), o que atrasa o tempo geral de processamento da aplicação. Isso geralmente pode ocasionar o downgrade da performance das operações de junção. O recurso de otimização de junções skew manipula dinamicamente a distorção nas junções sort-merge dividindo (e replicando, se necessário) tarefas distorcidas em tarefas de tamanho aproximadamente uniforme.

Esse recurso é habilitado quando `spark.sql.adaptive.skewJoin.enabled` está definido como true. Para obter mais detalhes, consulte a documentação do Apache Spark. A distorção de dados é discutida mais detalhadamente na seção Otimizar as operações de shuffle.

Investigar problemas de performance usando a interface do usuário do Spark

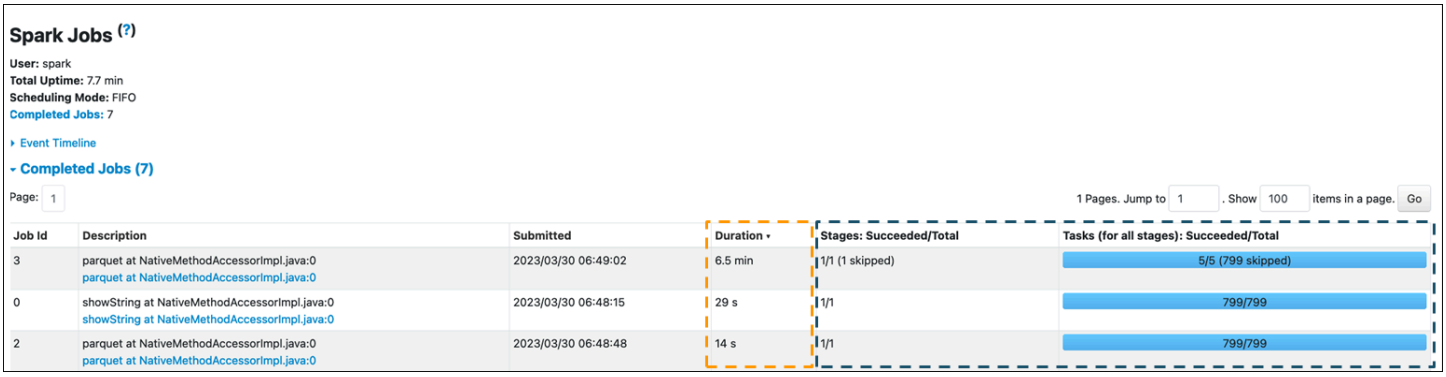
Antes de aplicar as práticas recomendadas para ajustar a performance de seus trabalhos do AWS Glue, é altamente recomendável que você defina o perfil da performance e identifique os gargalos. Isso ajudará você a focar as coisas certas.

Para uma análise rápida, as métricas do Amazon CloudWatch fornecem uma visão básica das métricas do seu trabalho. A interface do usuário do Spark fornece uma visão mais profunda do ajuste de performance. Para usar a interface do usuário do Spark com o AWS Glue, você deve habilitar a interface do usuário do Spark para seus trabalhos do AWS Glue. Depois de se familiarizar com a interface do usuário do Spark, siga as estratégias para ajustar a performance de trabalho do Spark para identificar e reduzir o impacto dos gargalos com base em suas descobertas.

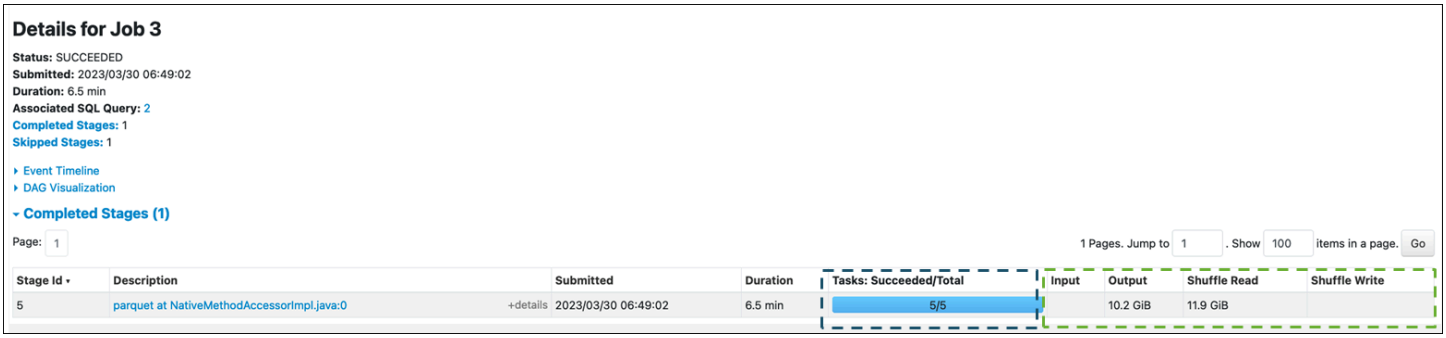
Identificar gargalos usando a interface do usuário do Spark

Quando você abre a interface do usuário do Spark, as aplicações do Spark estão listados em uma tabela. Por padrão, um Nome de aplicação de um trabalho do AWS Glue é `nativespark-<Job Name>-<Job Run ID>`. Escolha a aplicação de destino do Spark com base no ID de execução do trabalho para abrir a guia Trabalhos. Execuções incompletas de trabalhos, como execuções de trabalhos de streaming, estão listadas em Mostrar aplicações incompletas.

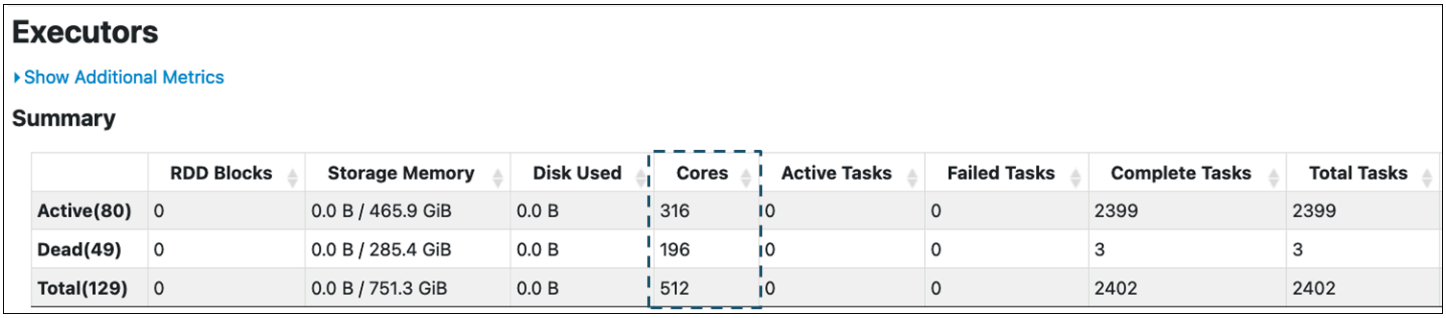
A guia Trabalhos mostra um resumo de todos os trabalhos na aplicação do Spark. Para determinar qualquer falha da etapa ou tarefa, verifique o número total de tarefas. Para encontrar os gargalos, classifique escolhendo Duração. Aprofunde-se nos detalhes de trabalhos de longa duração escolhendo o link mostrado na coluna Descrição.



A página Detalhes do trabalho lista as etapas. Nesta página, você pode ver insights gerais, como a duração, o número de tarefas bem-sucedidas e o total, o número de entradas e saídas e a quantidade de leitura e gravação do shuffle.



A guia Executor mostra a capacidade do cluster do Spark em detalhes. Você pode verificar o número total de núcleos. O cluster mostrado na captura de tela a seguir contém 316 núcleos ativos e 512 núcleos no total. Por padrão, cada núcleo pode processar uma tarefa do Spark ao mesmo tempo.



	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

Com base no valor 5/5 mostrado na página Detalhes do trabalho, a etapa 5 é a etapa mais longa, mas usa apenas 5 núcleos do total de 512. Como o paralelismo dessa etapa é muito baixo, mas leva um tempo significativo, você pode identificá-la como um gargalo. Para melhorar a performance, você precisa entender o porquê. Para saber mais sobre como reconhecer e reduzir o impacto de gargalos comuns de performance, consulte Strategies for tuning Spark job performance.

Estratégias para ajustar a performance de trabalho do Spark

Quando for ajustar os parâmetros, observe as seguintes práticas recomendadas:

- Determine suas metas de performance antes de começar a identificar os problemas.
- Use as métricas para identificar os problemas antes de tentar alterar os parâmetros de ajuste.

Para obter os resultados mais consistentes ao ajustar um trabalho, desenvolva uma estratégia de linha de base para fazer os ajustes.

Estratégia de linha de base para ajuste de performance

Geralmente, o ajuste de performance é feito no seguinte fluxo de trabalho:

1. Determine as metas de performance.
2. Meça as métricas.
3. Identifique os gargalos.
4. Reduza o impacto dos gargalos.
5. Repita as etapas de 2 a 4 até atingir a meta pretendida.

Primeiro, determine suas metas de performance. Por exemplo, uma de suas metas pode ser concluir a execução de um AWS Glue trabalho em 3 horas. Depois de definir suas metas, avalie as métricas de performance do trabalho. Identifique tendências em métricas e gargalos para atingir as metas. Em particular, identificar gargalos é muito importante para solucionar problemas, depurar e ajustar a performance. Durante a execução de uma aplicação do Spark, ele registra o status e as estatísticas de cada tarefa no log de eventos do Spark.

Em AWS Glue, você pode visualizar as métricas do Spark por meio da interface de usuário da Web do Spark, fornecida pelo servidor de histórico do Spark. AWS Glue para trabalhos do Spark, você pode enviar registros de eventos do Spark para um local especificado no Amazon S3. AWS Glue também fornece um AWS CloudFormation modelo de exemplo e um Dockerfile para iniciar o servidor de histórico do Spark em uma instância do Amazon EC2 ou em seu computador local, para que você possa usar a interface do usuário do Spark com registros de eventos.

Depois de determinar suas metas de performance e identificar métricas para avaliá-las, você pode começar a identificar e remediar gargalos usando as estratégias nas seções a seguir.

Práticas de ajuste da performance de trabalho do Spark

Você pode usar as seguintes estratégias AWS Glue para ajustar o desempenho das tarefas do Spark:

- AWS Glue recursos:
  - Escalar a capacidade do cluster
  - Use a AWS Glue versão mais recente
- Aplicações do Spark:
  - Reduzir a quantidade de dados verificados
  - Paralelizar tarefas
  - Otimizar as operações de shuffle
  - Minimizar a sobrecarga de planejamento
  - Otimizar funções definidas pelo usuário

Antes de usar essas estratégias, você deve ter acesso às métricas e à configuração do seu trabalho do Spark. Você pode encontrar essas informações na documentação do AWS Glue.

Do ponto de vista dos AWS Glue recursos, você pode obter melhorias de desempenho adicionando AWS Glue trabalhadores e usando a AWS Glue versão mais recente.

Sob a perspectiva das aplicações do Apache Spark, você tem acesso a várias estratégias que podem melhorar a performance. Se dados desnecessários forem carregados no cluster do Spark, você poderá removê-los para reduzir a quantidade de dados carregados. Se tiver recursos de clusters do Spark subutilizados e pouca E/S de dados, você poderá identificar tarefas para paralelizar. Você também pode querer otimizar as operações pesadas de transferência de dados, como junções, se elas estiverem demorando muito. Você também pode otimizar o plano de consulta do seu trabalho ou reduzir a complexidade computacional de tarefas individuais do Spark.

Para aplicar essas estratégias com eficiência, você deve identificar quando elas são aplicáveis consultando suas métricas. Para obter mais detalhes, consulte todas as seções a seguir. Essas técnicas funcionam não apenas para ajuste de desempenho, mas também para resolver problemas típicos, como erros out-of-memory (OOM).

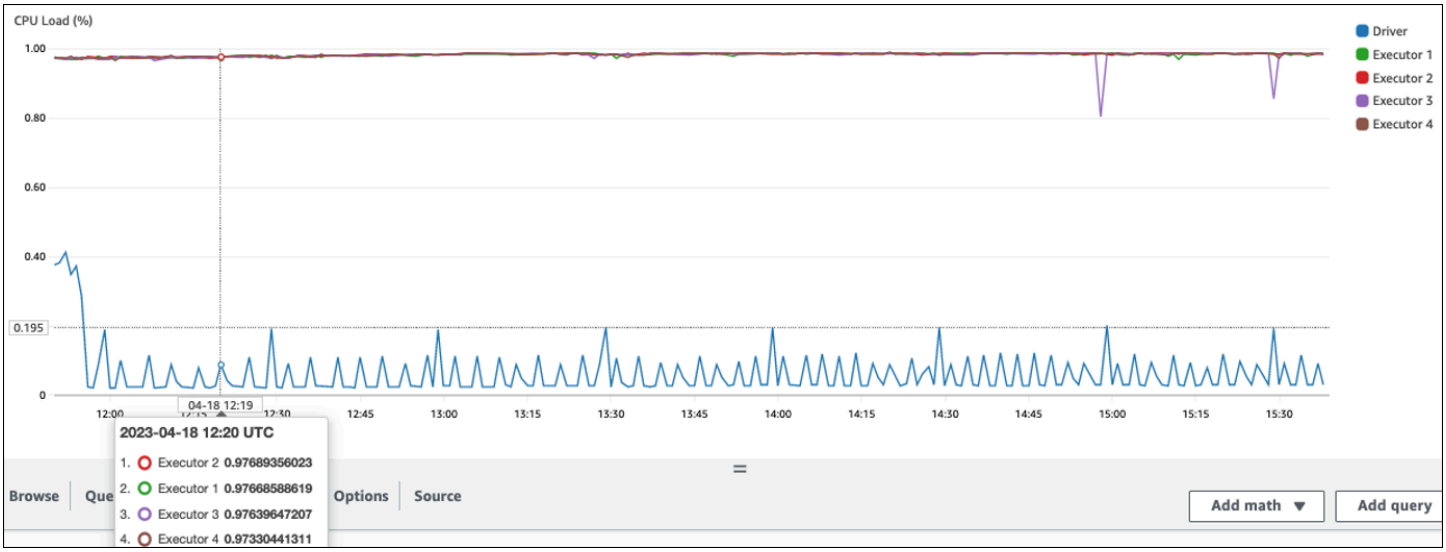
Escalar a capacidade do cluster

Se seu trabalho está demorando muito, mas os executores estão consumindo recursos suficientes e o Spark está criando um grande volume de tarefas em relação aos núcleos disponíveis, considere escalar a capacidade do cluster. Para avaliar se isso é apropriado, use as métricas a seguir.

CloudWatch métricas

- Verifique a carga da CPU e a utilização da memória para determinar se os executores estão consumindo recursos suficientes.
- Verifique há quanto tempo o trabalho foi executado para avaliar se o tempo de processamento é muito longo para atender às suas metas de performance.

No exemplo a seguir, quatro executores estão sendo executados com mais de 97% da carga da CPU, mas o processamento não foi concluído após cerca de três horas.

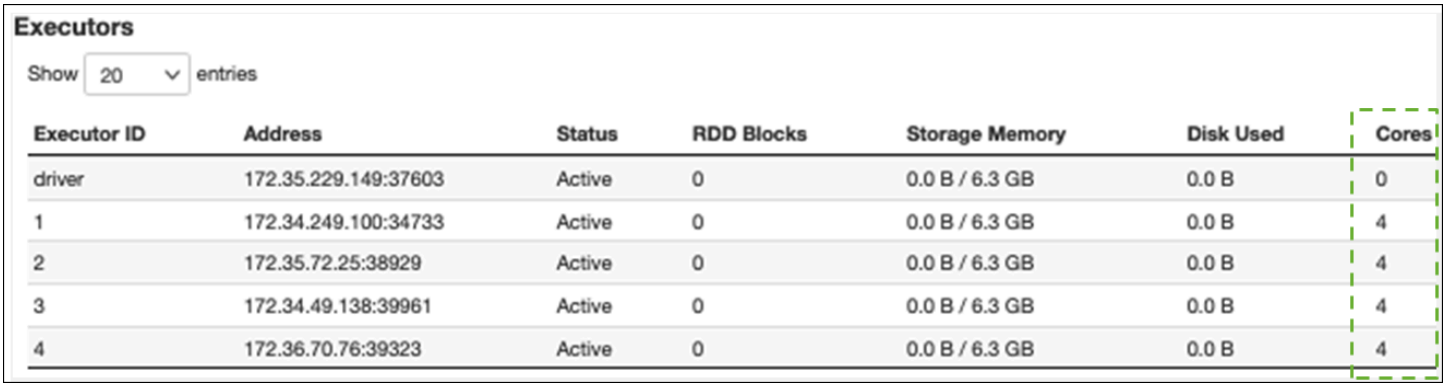


> **Note**
>
> Se a carga da CPU for baixa, você provavelmente não se beneficiará com a escalabilidade da capacidade do cluster.

Interface do usuário do Spark

Na guia Trabalho ou na guia Etapa, você pode ver o número de tarefas para cada trabalho ou etapa. No exemplo a seguir, o Spark criou `58100` tarefas.

Na guia Executor, você pode ver o número total de executores e tarefas. Na captura de tela a seguir, cada executor do Spark tem quatro núcleos e pode realizar quatro tarefas simultaneamente.



Neste exemplo, o número de tarefas do Spark (`58100`)) é muito maior do que as 16 tarefas que os executores podem processar simultaneamente (4 executores × 4 núcleos).

Se você observar esses sintomas, considere escalar o cluster. Você pode escalar a capacidade do cluster usando as seguintes opções:

- Ativar AWS Glue Auto Scaling — O Auto Scaling está disponível para AWS Glue suas tarefas de extração, transformação e carregamento (ETL) e streaming AWS Glue na versão 3.0 ou posterior. AWS Glue adiciona e remove automaticamente os trabalhadores do cluster, dependendo do número de partições em cada estágio ou da taxa na qual os microlotes são gerados na execução do trabalho.

  Se você observar uma situação em que o número de operadores não aumenta, mesmo que o Auto Scaling esteja habilitado, considere adicionar operadores manualmente. No entanto, observe que escalar manualmente para uma etapa pode fazer com que muitos operadores fiquem ociosos durante as etapas posteriores, custando mais sem nenhum ganho de performance.

Depois de ativar o Auto Scaling, você pode ver o número de executores nas métricas do executor. CloudWatch Use as seguintes métricas para monitorar a demanda por executores nas aplicações do Spark:

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

Para obter mais informações sobre métricas, consulte Monitoramento AWS Glue usando CloudWatch métricas da Amazon.

- Aumentar a escala horizontalmente: aumente o número de operadores do AWS Glue : você pode aumentar manualmente o número de operadores do AWS Glue . Adicione operadores somente enquanto observar operadores ociosos. Nesse ponto, adicionar mais operadores aumentará os custos sem melhorar os resultados. Para obter mais informações, consulte Paralelizar tarefas.
- Amplie: use um tipo de trabalhador maior — você pode alterar manualmente o tipo de instância de seus AWS Glue trabalhadores para usar trabalhadores com mais núcleos, memória e armazenamento. Tipos de operadores maiores possibilitam que você aumente a escala verticalmente e execute trabalhos intensivos de integração de dados, como transformações de dados que consomem muita memória, agregações distorcidas e verificações de detecção de entidades envolvendo petabytes de dados.

  Aumentar a escala verticalmente também ajuda nos casos em que o driver do Spark precisa de maior capacidade, por exemplo, porque o plano de consulta do trabalho é muito grande. Para obter mais informações sobre tipos de trabalhadores e desempenho, consulte a postagem do blog AWS Big Data Dimensione suas AWS Glue tarefas do Apache Spark com novos tipos de trabalhadores maiores G.4X e G.8X.

  O uso de operadores maiores também pode reduzir o número total de operadores necessários, o que aumenta a performance ao reduzir o shuffle em operações intensivas, como junção.

Use a AWS Glue versão mais recente

Recomendamos usar a AWS Glue versão mais recente. Há várias otimizações e atualizações incorporadas em cada versão que podem melhorar automaticamente a performance do trabalho. Por exemplo, a AWS Glue versão 4.0 fornece os seguintes novos recursos:

- O novo tempo de execução otimizado do Apache Spark 3.3.0 — AWS Glue 4.0 se baseia no tempo de execução do Apache Spark 3.3.0, trazendo melhorias de desempenho comparáveis

às do Spark de código aberto. O runtime do Spark 3.3.0 é baseado em muitas das inovações do Spark 2.x.

- Conector aprimorado do Amazon Redshift: o AWS Glue 4.0 e versões posteriores oferecem integração do Amazon Redshift para Apache Spark. A integração se baseia em um conector de código aberto existente e o aprimora em termos de performance e segurança. A integração ajuda as aplicações a terem uma performance até dez vezes mais rápida. Para obter mais informações, consulte a publicação do blog Amazon Redshift integration with Apache Spark.
- Execução baseada em SIMD para leituras vetorizadas com dados CSV e JSON — a AWS Glue versão 3.0 e versões posteriores adicionam leitores otimizados que podem acelerar significativamente o desempenho geral do trabalho em comparação com leitores baseados em linhas. Para obter mais informações sobre dados CSV, consulte Otimizar o desempenho de leitura com o leitor de SIMD vetorizado para CSV. Para obter mais informações sobre dados JSON, consulte Usar o leitor vetorizado SIMD para JSON com formato colunar Apache Arrow.

Cada AWS Glue versão incluirá atualizações desse tipo, entre muitas, incluindo conectores, atualizações de drivers e bibliotecas. Para obter mais informações, consulte AWS Glue Versões e Migração de AWS Glue trabalhos para a AWS Glue versão 4.0.

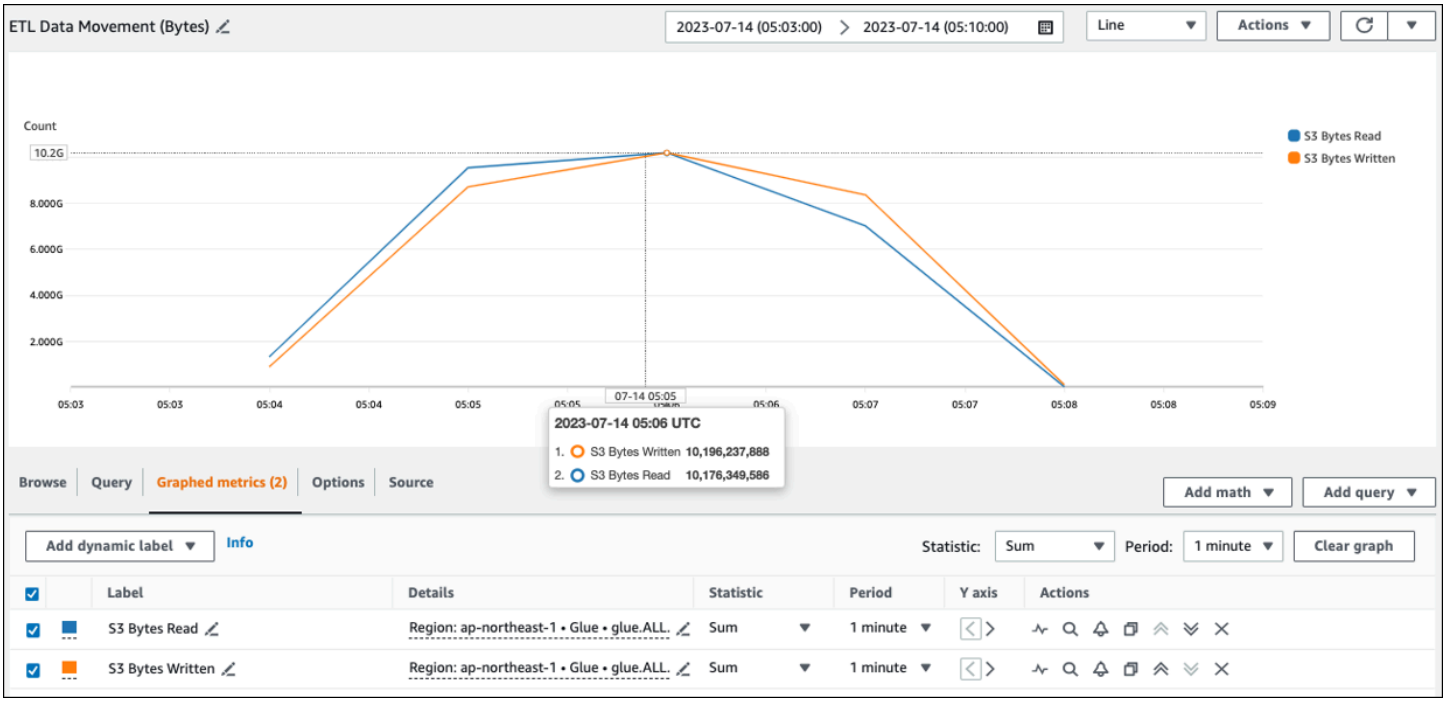
Reduzir a quantidade de dados verificados

Para começar, considere carregar somente os dados de que você precisa. Você pode melhorar a performance apenas reduzindo a quantidade de dados carregados em seu cluster do Spark para cada fonte de dados. Para avaliar se essa abordagem é apropriada, use as métricas a seguir.

Você pode verificar os bytes lidos do Amazon S3 em CloudWatchmétricas e mais detalhes na interface do usuário do Spark, conforme descrito na seção UI do Spark.

CloudWatch métricas

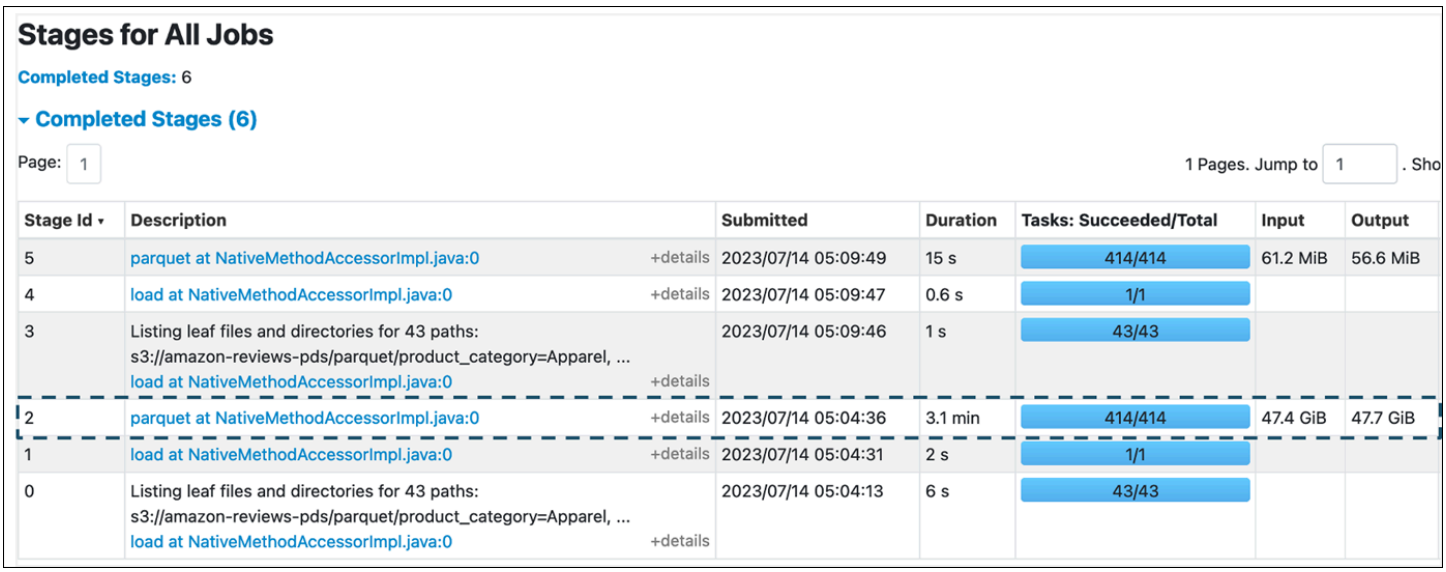
Você pode ver o tamanho aproximado de leitura do Amazon S3 no Movimentação de dados ETL (Bytes). Essa métrica mostra o número de bytes lidos no Amazon S3 por todos os executores desde o relatório anterior. Você pode usá-la para monitorar a movimentação de dados ETL do Amazon S3 e comparar as leituras com as taxas de ingestão de fontes de dados externas.



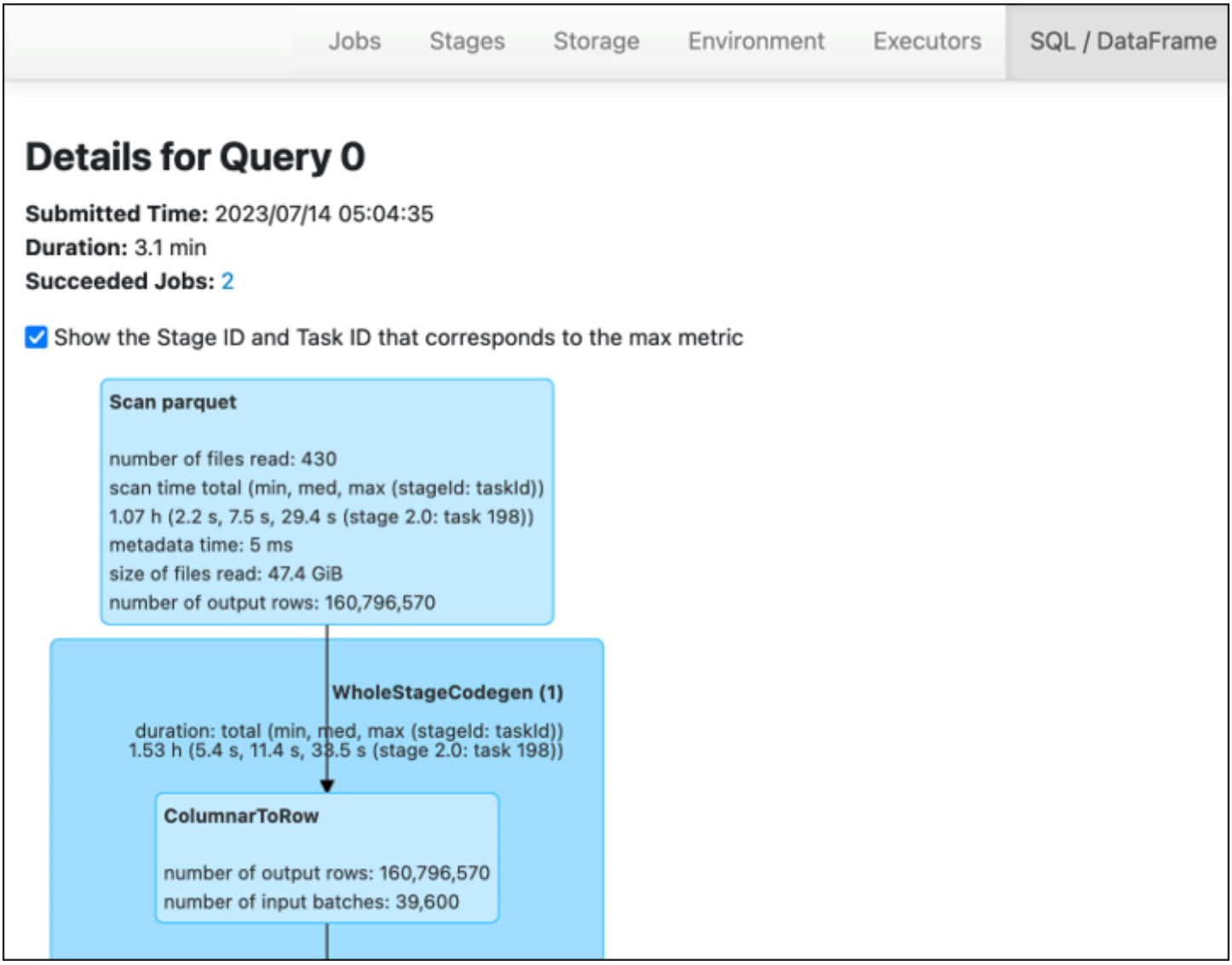
Se você observar um ponto de dados de Bytes lidos do S3 maior do que o esperado, considere as soluções a seguir.

Interface do usuário do Spark

Na guia Stage na interface do AWS Glue usuário do Spark, você pode ver o tamanho de entrada e saída. No exemplo a seguir, a etapa 2 lê a entrada de 47,4 GiB e a saída de 47,7 GiB, enquanto a etapa 5 lê a entrada de 61,2 MiB e a saída de 56,6 MiB.



Quando você usa o Spark SQL ou as DataFrame abordagens em seu AWS Glue trabalho, a guia SQL/D DataFrame mostra mais estatísticas sobre esses estágios. Nesse caso, a etapa 2 mostra o número de arquivos lidos: 430, o tamanho dos arquivos lidos: 47,4 GiB e o número de linhas de saída: 160.796.570.



Se você observar que há uma diferença significativa no tamanho entre os dados que você está lendo e os dados que você está usando, tente as soluções a seguir.

Amazon S3

Para reduzir a quantidade de dados carregados em seu trabalho durante a leitura do Amazon S3, considere o tamanho do arquivo, a compactação, o formato do arquivo e o layout do arquivo (partições) do seu conjunto de dados. AWS Glue As tarefas do For Spark geralmente são usadas

para ETL de dados brutos, mas para um processamento distribuído eficiente, você precisa inspecionar os recursos do formato da fonte de dados.

- Tamanho do arquivo: recomendamos manter o tamanho do arquivo de entradas e saídas dentro de uma faixa moderada (por exemplo, 128 MB). Arquivos muito pequenos e muito grandes podem causar problemas.

  Um grande número de arquivos pequenos causa os seguintes problemas:

  - Carga de rede pesada no Amazon S3 devido à I/O sobrecarga necessária para fazer solicitações (como `ListGet`, ou`Head`) para muitos objetos (em comparação com alguns objetos que armazenam a mesma quantidade de dados).
  - Carga pesada I/O e de processamento no driver Spark, que gerará muitas partições e tarefas e levará a um paralelismo excessivo.

  Por outro lado, se o tipo de arquivo não for divisível (como gzip) e os arquivos forem muito grandes, a aplicação do Spark deverá esperar até que uma única tarefa termine de ler o arquivo todo.

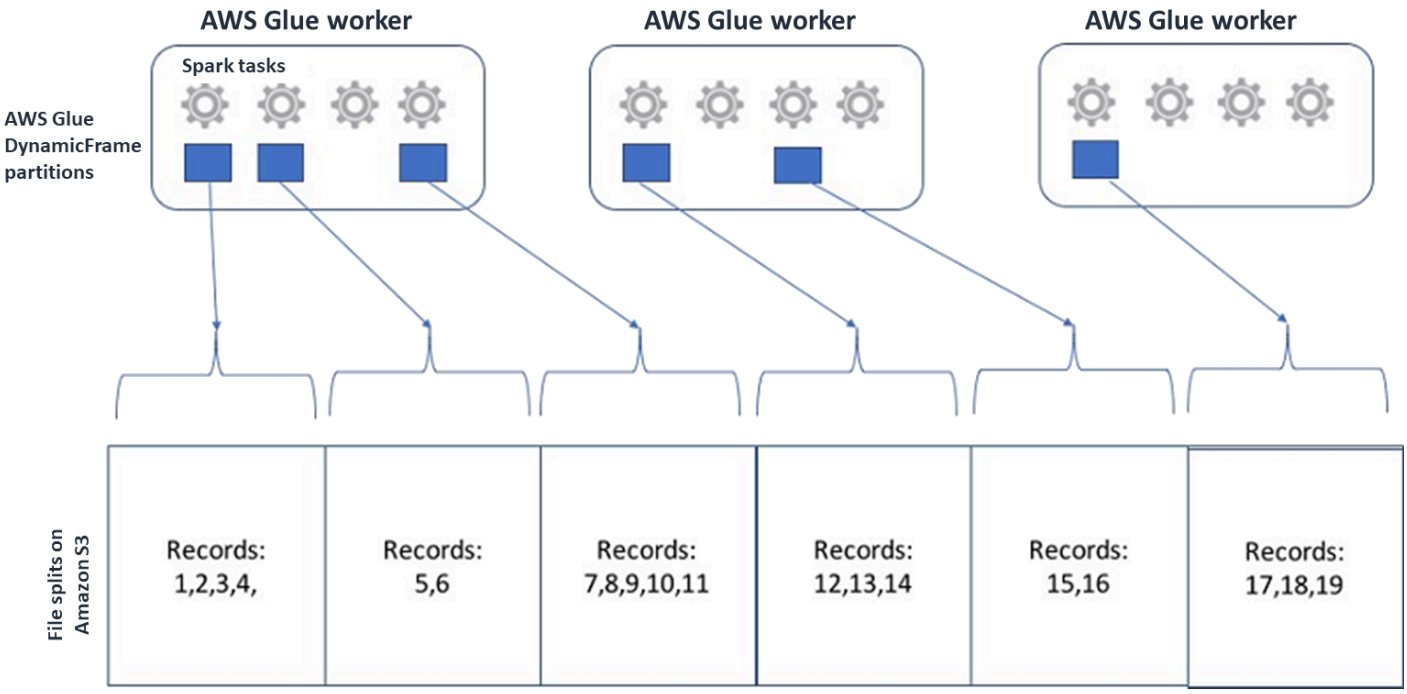
  Para reduzir o paralelismo excessivo que ocorre quando uma tarefa do Apache Spark é criada para cada arquivo pequeno, use o agrupamento de arquivos para. DynamicFrames Essa abordagem reduz a probabilidade de uma exceção de OOM do driver do Spark. Para configurar o agrupamento de arquivos, defina os parâmetros `groupFiles` e `groupSize`. O exemplo de código a seguir usa a AWS Glue DynamicFrame API em um script ETL com esses parâmetros.

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

- Compactação: se seus objetos do S3 estiverem na casa das centenas de megabytes, considere compactá-los. Existem vários formatos de compressão, que podem ser amplamente classificados em dois tipos:

  - Formatos de compactação não divisíveis, como gzip, exigem que o arquivo todo seja descompactado por um operador.
  - Formatos de compactação divisíveis, como bzip2 ou LZO (indexado), permitem a descompactação parcial de um arquivo, que pode ser paralelizado.

Para o Spark (e outros mecanismos comuns de processamento distribuído), você dividirá seu arquivo de dados de origem em fragmentos que seu mecanismo possa processar paralelamente. Essas unidades geralmente são chamadas de splits. Depois que seus dados estiverem em um formato divisível, os AWS Glue leitores otimizados poderão recuperar divisões de um objeto do S3 fornecendo à `GetObject` API a `Range` opção de recuperar somente blocos específicos. Considere o diagrama a seguir para ver como isso funcionaria na prática.



Os dados compactados podem acelerar significativamente sua aplicação, desde que os arquivos tenham um tamanho ideal ou sejam divisíveis. Os tamanhos de dados menores reduzem os dados verificados do Amazon S3 e o tráfego de rede do Amazon S3 para seu cluster do Spark. Por outro lado, é necessária mais CPU para compactar e descompactar dados. A quantidade de computação necessária é escalada com a taxa de compactação do seu algoritmo de compactação. Considere essa compensação ao escolher seu formato de compactação divisível.

> **Note**
>
> Embora os arquivos gzip geralmente não sejam divisíveis, você pode compactar blocos de parquet individuais com gzip, e esses blocos podem ser paralelizados.

- Formato de arquivo: use um formato colunar. O Apache Parquet e o Apache ORC são formatos de dados colunares conhecidos. O Parquet e o ORC armazenam dados de forma eficiente empregando compactação baseada em colunas, codificando e compactando cada coluna com base em seu tipo de dados. Para obter mais informações sobre codificações do Parquet, consulte Parquet encoding definitions. Os arquivos do Parquet também são divisíveis.

  Os formatos colunares agrupam valores por coluna e os armazenam juntos em blocos. Ao usar formatos colunares, você pode ignorar blocos de dados que correspondam às colunas que você não planeja usar. As aplicações do Spark podem recuperar somente as colunas de que você precisa. Geralmente, melhores taxas de compactação ou ignorar blocos de dados significa ler menos bytes do Amazon S3, o que resulta em uma melhor performance. Ambos os formatos também são compatíveis com as seguintes abordagens de pushdown para reduzir a E/S:

  - Pushdown de projeção: o pushdown de projeção é uma técnica para recuperar somente as colunas especificadas em sua aplicação. Você especifica colunas na sua aplicação do Spark, conforme mostrado nos seguintes exemplos:
    - DataFrame exemplo: `df.select("star_rating")`
    - Exemplo do Spark SQL: `spark.sql("select start_rating from <table>")`
  - Pushdown de predicados: o pushdown de predicados é uma técnica para processar de forma eficiente as cláusulas `WHERE` e `GROUP BY`. Ambos os formatos têm blocos de dados que representam os valores das colunas. Cada bloco contém estatísticas do bloco, como valores máximos e mínimos. O Spark pode usar essas estatísticas para determinar se o bloco deve ser lido ou ignorado, dependendo do valor do filtro usado na aplicação. Para usar esse recurso, adicione mais filtros nas condições, conforme mostrado nos seguintes exemplos:
    - DataFrame exemplo: `df.select("star_rating").filter("star_rating < 2")`
    - Exemplo do Spark SQL: `spark.sql("select * from <table> where star_rating < 2")`
- Layout do arquivo: ao armazenar seus dados do S3 em objetos em caminhos diferentes com base em como os dados serão usados, você pode recuperar dados relevantes com eficiência. Para obter mais informações, consulte Organizar objetos usando prefixos na documentação do Amazon S3. O AWS Glue tem suporte para armazenamento de chaves e valores nos prefixos do Amazon S3 no formato `key=value`, particionando seus dados pelo caminho do Amazon S3. Ao particionar seus dados, você pode restringir a quantidade de dados verificados por cada aplicação de analytics downstream, melhorando a performance e reduzindo o custo. Para obter mais informações, consulte Gerenciando partições para saída ETL em. AWS Glue

O particionamento divide sua tabela em partes diferentes e mantém os dados relacionados em arquivos agrupados com base nos valores das colunas, como ano, mês e dia, conforme mostrado no exemplo a seguir.

```
# 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
...
```

Você pode definir partições para seu conjunto de dados modelando-o com uma tabela no AWS Glue Data Catalog. Você pode então restringir a quantidade de dados verificados usando o descarte de partições da seguinte forma:

- Para AWS Glue DynamicFrame, defina `push_down_predicate` (ou`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'",
  )
  ```

- Para o Spark DataFrame, defina um caminho fixo para remover partições.

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

- Para o Spark SQL, você pode definir a cláusula where para descartar partições do Catálogo de Dados.

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

- Para particionar por data ao gravar seus dados com AWS Glue, você define partitionKeys DynamicFrame in ou partitionBy () com as informações de data DataFrame em suas colunas da seguinte forma.

  - 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>/')
```

Isso pode melhorar a performance dos consumidores de seus dados de saída.

Se você não tiver acesso para alterar o pipeline que cria seu conjunto de dados de entrada, o particionamento não é uma opção. Em vez disso, você pode excluir caminhos do S3 desnecessários usando padrões globais. Defina exclusões ao ler. DynamicFrame Por exemplo, o código a seguir exclui dias nos meses 01 a 09, no ano de 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'
)
```

Você também pode definir exclusões nas propriedades da tabela no Catálogo de Dados:

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

- Muitas partições do Amazon S3: evite particionar seus dados do Amazon S3 em colunas que contenham uma grande variedade de valores, como uma coluna de ID com milhares de valores. Isso pode aumentar substancialmente o número de partições em seu bucket, porque o número de partições possíveis é o produto de todos os campos pelos quais você particionou. Muitas partições podem causar o seguinte:
  - Maior latência para recuperar metadados de partição do Catálogo de Dados
  - Maior número de arquivos pequenos, o que exige mais solicitações de API do Amazon S3 (`List`, `Get` e `Head`)

Por exemplo, quando você define um tipo de data em `partitionBy` ou `partitionKeys`, o particionamento em nível de data como `yyyy/mm/dd` é bom para muitos casos de uso. No entanto, `yyyy/mm/dd/<ID>` pode gerar tantas partições que isso afetaria negativamente a performance como um todo.

Por outro lado, alguns casos de uso, como aplicações de processamento em tempo real, exigem muitas partições, como `yyyy/mm/dd/hh`. Se seu caso de uso exigir partições substanciais, considere usar índices de partição do AWS Glue para reduzir a latência na recuperação de metadados de partições do Catálogo de Dados.

Bancos de dados e JDBC

Para reduzir a verificação de dados ao recuperar informações de um banco de dados, você pode especificar um predicado `where` (ou cláusula) em uma consulta SQL. Os bancos de dados que não dispõem de uma interface SQL fornecerão seu próprio mecanismo de consulta ou filtragem.

Ao usar conexões do Java Database Connectivity (JDBC), forneça uma consulta de seleção com a cláusula `where` para os seguintes parâmetros:

- Para DynamicFrame, use a opção SampleQuery. Ao usar `create_dynamic_frame.from_catalog`, configure o argumento `additional_options` como a seguir.

```
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"
)
```

Ao `using create_dynamic_frame.from_options`, configure o argumento `connection_options` como a seguir.

```
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
    }
)
```

- Para DataFrame, use a opção de consulta.

```
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()
```

- Para o Amazon Redshift, use a AWS Glue versão 4.0 ou posterior para aproveitar o suporte a push down no conector 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"}
)
```

- Para outros bancos de dados, consulte a documentação do banco de dados.

AWS Glue opções

- Para evitar uma verificação completa de todas as execuções contínuas de trabalhos e processar somente os dados que não estavam presentes durante a última execução do trabalho, habilite os marcadores de tarefas.
- Para limitar a quantidade de dados de entrada a serem processados, habilite a execução delimitada com marcadores de tarefas. Isso ajuda a reduzir a quantidade de dados verificados para cada execução de trabalho.

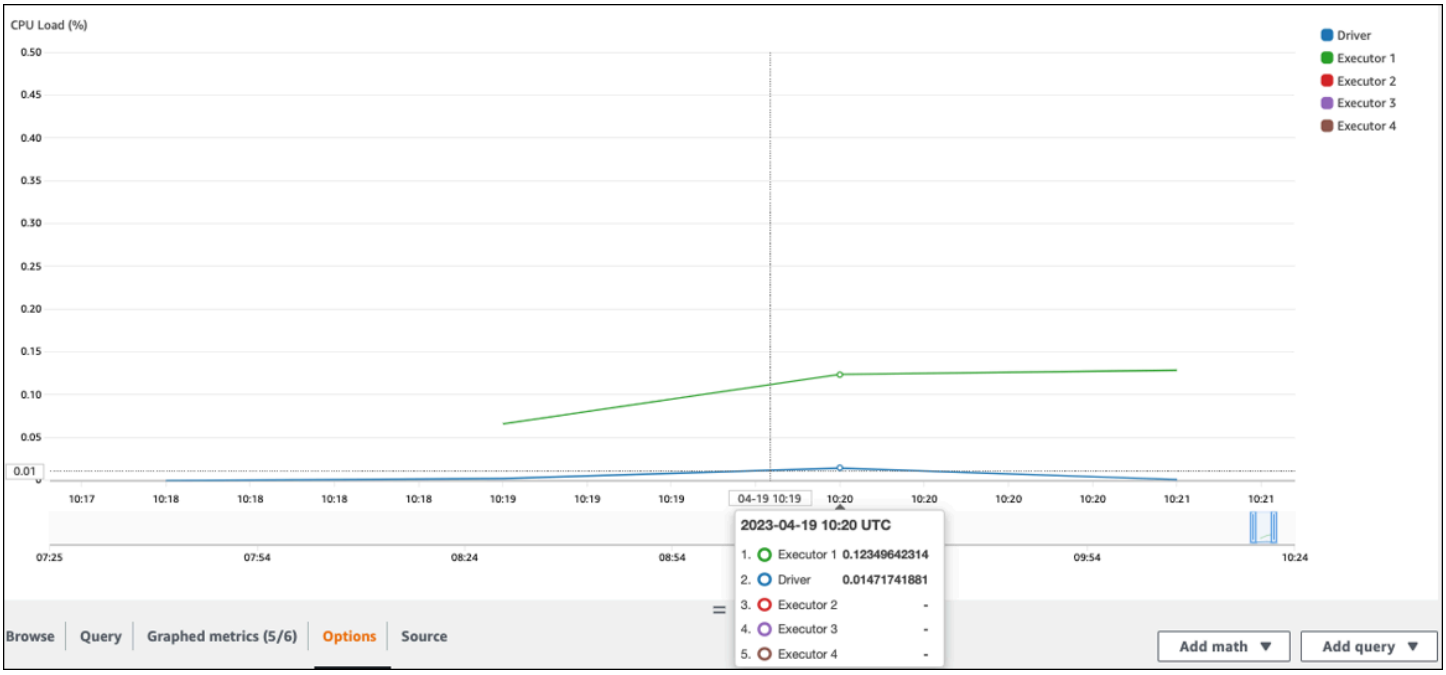
Paralelizar tarefas

Para otimizar a performance, é importante paralelizar tarefas para cargas e transformações de dados. Conforme analisado em Principais tópicos do Apache Spark, o número de partições resilientes do conjunto de dados distribuído (RDD) é importante, pois determina o grau de paralelismo. Cada tarefa que o Spark cria corresponde a uma partição do RDD em uma base 1:1. Para obter a melhor performance, você precisa entender como o número de partições do RDD é determinado e como esse número é otimizado.

Se você não tiver paralelismo suficiente, os seguintes sintomas serão registrados nas CloudWatchmétricas e na interface do usuário do Spark.

CloudWatch métricas

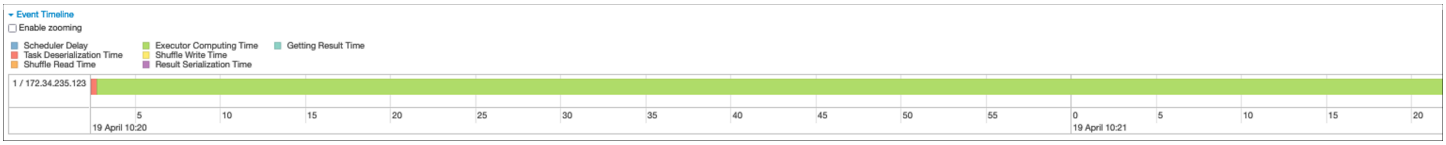
Verifique a carga da CPU e a utilização da memória. Se alguns executores não estiverem processando durante uma fase do seu trabalho, é apropriado melhorar o paralelismo. Nesse caso, durante o período visualizado, o Executor 1 estava executando uma tarefa, mas os demais executores (2, 3 e 4) não. Você pode inferir que esses executores não tiveram tarefas atribuídas pelo driver do Spark.



Interface do usuário do Spark

Na guia Etapa na interface do usuário do Spark, você pode ver o número de tarefas em uma etapa. Nesse caso, o Spark executou apenas uma tarefa.

Além disso, o cronograma de eventos mostra o Executor 1 processando uma tarefa. Isso significa que o trabalho nessa etapa foi executado por completo em um executor, enquanto os outros estavam ociosos.



Se você observar esses sintomas, tente as soluções a seguir para cada fonte de dados.

Paralelizar a carga de dados do Amazon S3

Para paralelizar cargas de dados do Amazon S3, primeiro verifique o número padrão de partições. Você então pode determinar manualmente um número alvo de partições, mas evite ter muitas partições.

Determinar o número padrão de partições

Para o Amazon S3, o número inicial de partições do Spark RDD (cada uma delas corresponde a uma tarefa do Spark) é determinado pelos recursos do seu conjunto de dados do Amazon S3 (por exemplo, formato, compactação e tamanho). Quando você cria um AWS Glue DynamicFrame ou um Spark a DataFrame partir de objetos CSV armazenados no Amazon S3, o número inicial de partições RDD `NumPartitions` () pode ser calculado aproximadamente da seguinte forma:

- Tamanho do objeto <= 64 MB: `NumPartitions = Number of Objects`
- Tamanho do objeto > 64 MB: `NumPartitions = Total Object Size / 64 MB`
- Não divisível (gzip): `NumPartitions = Number of Objects`

Conforme analisado na seção Reduzir a quantidade de dados verificados, o Spark divide objetos grandes do S3 em splits que podem ser processados paralelamente. Quando o objeto é maior que o tamanho do split, o Spark divide o objeto e cria uma partição do RDD (e uma tarefa) para cada split. O tamanho do split do Spark é baseado no formato dos dados e no ambiente de runtime, mas esta é uma aproximação inicial razoável. Alguns objetos são compactados usando formatos de compactação não divisíveis, como gzip, para que o Spark não possa dividi-los.

O `NumPartitions` valor pode variar dependendo do formato dos dados, da compactação, da AWS Glue versão, do número de AWS Glue trabalhadores e da configuração do Spark.

Por exemplo, quando você carrega um único `csv.gz` objeto de 10 GB usando um Spark DataFrame, o driver do Spark cria somente uma partição RDD (`NumPartitions=1`) porque o gzip não pode ser dividido. Isso resulta em uma carga pesada em um executor específico do Spark e nenhuma tarefa é atribuída aos demais executores, conforme descrito na figura a seguir.

Verifique o número real de tarefas (`NumPartitions`) da etapa na guia Etapa da Interface do usuário web do Spark, ou execute `df.rdd.getNumPartitions()` em seu código para verificar o paralelismo.

Ao encontrar um arquivo gzip de 10 GB, examine se o sistema que está gerando esse arquivo pode gerá-lo em um formato divisível. Se isso não for uma opção, talvez seja necessário escalar a capacidade do cluster para processar o arquivo. Para executar transformações com eficiência nos dados que você carregou, você precisará rebalancear seu RDD nos operadores do seu cluster usando a repartição.

Determinar manualmente um número alvo de partições

Dependendo das propriedades de seus dados e da implementação de determinadas funcionalidades pelo Spark, você pode acabar com um valor de `NumPartitions` baixo, mesmo que o trabalho subjacente ainda possa ser paralelizado. Se `NumPartitions` for muito pequeno, execute `df.repartition(N)` para aumentar o número de partições para que o processamento possa ser distribuído nos vários executores do Spark.

Nesse caso, a execução `df.repartition(100)` aumentará o `NumPartitions` de 1 para 100, criando 100 partições de seus dados, cada uma com uma tarefa que pode ser atribuída aos outros executores.

A operação `repartition(N)` divide os dados inteiros igualmente (10 GB/100 partições = 100 MB/partição), evitando a distorção de dados em determinadas partições.

> **Note**
>
> Quando uma operação de shuffle, como `join`, é executada, o número de partições aumenta ou diminui dinamicamente, dependendo do valor de `spark.sql.shuffle.partitions` ou `spark.default.parallelism`. Isso facilita uma troca mais eficiente de dados entre os executores do Spark. Para obter mais informações, consulte a documentação do Spark.

Sua meta ao determinar o número alvo de partições é maximizar o uso dos trabalhadores provisionados AWS Glue . O número de AWS Glue trabalhadores e o número de tarefas do Spark estão relacionados por meio do número de v. CPUs O Spark oferece suporte a uma tarefa para cada núcleo de vCPU. Na AWS Glue versão 3.0 ou posterior, você pode calcular um número alvo de partições usando a fórmula a seguir.

```
# 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
```

Neste exemplo, cada operador G.1X fornece quatro núcleos de vCPU para um executor do Spark (`spark.executor.cores = 4`). O Spark oferece suporte a uma tarefa para cada núcleo de vCPU, então os executores G.1X do Spark podem executar quatro tarefas simultaneamente (`numSlotPerExecutor`). Esse número de partições fará uso total do cluster se as tarefas levarem o mesmo tempo. No entanto, algumas tarefas demorarão mais do que outras, criando núcleos ociosos. Se isso acontecer, considere multiplicar `numPartitions` por 2 ou 3 para dividir e programar com eficiência as tarefas de gargalo.

Muitas partições

Um número excessivo de partições cria um número excessivo de tarefas. Isso causa uma carga pesada no driver do Spark devido à sobrecarga relacionada ao processamento distribuído, como tarefas de gerenciamento e troca de dados entre os executores do Spark.

Se o número de partições em seu trabalho for substancialmente maior do que o número alvo de partições, considere reduzir o número de partições. É possível reduzir partições usando as seguintes opções:

- Se o tamanho do arquivo for muito pequeno, use AWS Glue GroupFiles. Você pode reduzir o paralelismo excessivo resultante do lançamento de uma tarefa do Apache Spark para processar cada arquivo.
- Use `coalesce(N)` para mesclar as partições. Este é um processo de baixo custo. Ao reduzir o número de partições, `coalesce(N)` é preferível em vez de `repartition(N)`, pois `repartition(N)` executa o shuffle para distribuir igualmente a quantidade de registros em cada partição. Isso aumenta os custos e a sobrecarga de gerenciamento.
- Use a Execução Adaptativa de Consultas do Spark 3.x. Conforme analisado na seção Principais tópicos do Apache Spark, a Execução Adaptativa de Consultas fornece uma função para aglutinar automaticamente o número de partições. Você pode usar essa abordagem quando não consegue saber o número de partições até realizar a execução.

Paralelizar a carga de dados do JDBC

O número de partições do Spark RDD é determinado pela configuração. Observe que, por padrão, somente uma única tarefa é executada para verificar todo o conjunto de dados de origem por meio de uma consulta `SELECT`.

Tanto o Spark AWS Glue DynamicFrames quanto o Spark DataFrames oferecem suporte ao carregamento de dados JDBC paralelizado em várias tarefas. Isso é feito usando predicados `where` para dividir uma consulta `SELECT` em várias consultas. Para paralelizar as leituras do JDBC, configure as seguintes opções:

- Para AWS Glue DynamicFrame, defina `hashfield` (ou `hashexpression`) `hashpartition` e. Para saber mais, consulte Leitura de tabelas JDBC em paralelo.

```
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"
)
```

- Para Spark DataFrame, set `numPartitionspartitionColumn`,`lowerBound`, e. `upperBound` Para saber mais, consulte JDBC To Other Databases.

```
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/")
```

Paralelizar a carga de dados do DynamoDB ao usar o conector ETL

O número de partições do Spark RDD é determinado pelo parâmetro `dynamodb.splits`. Para paralelizar as leituras do Amazon DynamoDB, configure as seguintes opções:

- Aumente o valor de `dynamodb.splits`.
- Otimize o parâmetro seguindo a fórmula explicada em Tipos e opções de conexão para ETL no AWS Glue Spark.

Paralelizar a carga de dados do Kinesis Data Streams

O número de partições do Spark RDD é determinado pelo número de fragmentos no fluxo de dados de origem do Amazon Kinesis Data Streams. Se você tiver apenas alguns fragmentos em seu fluxo de dados, haverá apenas algumas tarefas do Spark. Isso pode resultar em baixo paralelismo nos processos downstream. Para paralelizar as leituras do Kinesis Data Streams, configure as seguintes opções:

- Aumente o número de fragmentos para obter mais paralelismo ao carregar dados do Kinesis Data Streams.
- Se sua lógica no microlote for complexa o suficiente, considere reparticionar os dados no início do lote, depois de remover as colunas desnecessárias.

Para obter mais informações, consulte Melhores práticas para otimizar o custo e o desempenho de trabalhos AWS Glue de ETL de streaming.

Paralelizar tarefas após o carregamento dos dados

Para paralelizar tarefas após o carregamento dos dados, aumente o número de partições do RDD usando as seguintes opções:

- Reparticione os dados para gerar um número maior de partições, especialmente logo após o carregamento inicial, se a carga em si não puder ser paralelizada.

  Ligue `repartition()` DynamicFrame ou DataFrame especifique o número de partições. Uma boa regra é duas ou três vezes o número de núcleos disponíveis.

  No entanto, ao gravar uma tabela particionada, isso pode levar a uma explosão de arquivos (cada partição pode potencialmente gerar um arquivo em cada partição da tabela). Para evitar isso, você

pode reparticionar sua coluna DataFrame por. Isso usa as colunas de partição da tabela para que os dados sejam organizados antes da gravação. Você pode especificar um número maior de partições sem colocar arquivos pequenos nas partições da tabela. No entanto, tenha cuidado para evitar a distorção de dados, em que alguns valores de partição acabam com a maioria dos dados e atrasam a conclusão da tarefa.

- Quando houver shuffles, aumente o valor de `spark.sql.shuffle.partitions`. Isso também pode ajudar com qualquer problema de memória ao realizar a operação de shuffle.

  Quando você tem mais de 2.001 partições de shuffle, o Spark usa um formato de memória compactada. Se você tiver um número próximo a esse, talvez queira definir o valor de `spark.sql.shuffle.partitions` acima desse limite para obter uma representação mais eficiente.

Otimizar as operações de shuffle

Certas operações, como `join()` `egroupByKey()`, exigem que o Spark execute uma operação de shuffle. O shuffle é o mecanismo do Spark para redistribuir dados para que sejam agrupados de forma diferente nas partições do RDD. Realizar a operação de shuffle pode ajudar a remediar gargalos de performance. No entanto, como o a operação de shuffle geralmente envolve a cópia de dados entre os executores do Spark, o shuffle é uma operação complexa e cara. Por exemplo, os shuffles geram os seguintes custos:

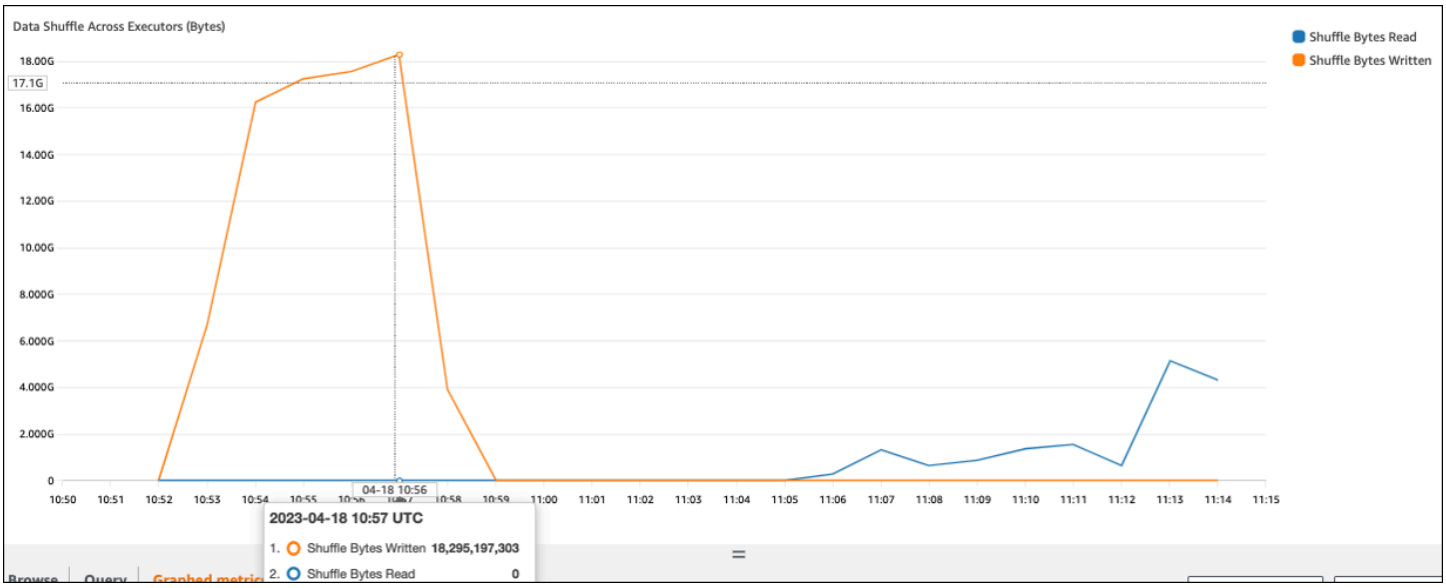
- E/S de disco:
  - Gera um grande número de arquivos intermediários no disco.
- E/S de rede:
  - Precisa de muitas conexões de rede (número de conexões = `Mapper × Reducer`).
  - Como os registros são agregados a novas partições do RDD que podem estar hospedadas em um executor diferente do Spark, uma fração substancial do seu conjunto de dados pode se movimentar entre os executores do Spark pela rede.
- Carga de CPU e memória:
  - Classifica valores e mescla conjuntos de dados. Essas operações são planejadas no executor, sobrecarregando o executor.

O shuffle é um dos fatores mais importantes na degradação da performance da sua aplicação do Spark. Ao armazenar os dados intermediários, ele pode esgotar espaço no disco local do executor, o que faz com que a tarefa do Spark falhe.

Você pode avaliar seu desempenho aleatório nas CloudWatch métricas e na interface do usuário do Spark.

CloudWatch métricas

Se o valor de Bytes gravados de shuffle for alto em comparação com o valor de Bytes lidos de shuffle, seu trabalho do Spark poderá usar operações de shuffle, como ou `join()` ou `groupByKey()`.



Interface do usuário do Spark

Na guia Etapa da interface do usuário do Spark, você pode verificar os valores de Registros/Tamanho de gravações de shuffle. Você também pode verificá-los na guia Executores.

Na captura de tela a seguir, cada executor troca aproximadamente 18,6 GB/4.020.000 registros com o processo shuffle, para um tamanho total de leitura de shuffle de cerca de 75 GB.

A coluna Vazamento de shuffle (Disco) mostra um grande volume de derramamento de dados da memória para o disco, o que pode causar um disco cheio ou um problema de performance.

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

Se você observar esses sintomas e a etapa demorar muito em comparação com suas metas de performance, ou se ela falhar com os erros `Out Of Memory` ou `No space left on device`, considere as soluções a seguir.

Otimizar a junção

A operação `join()`, que faz a junção de tabelas, é a operação de shuffle mais usada, mas geralmente é um gargalo de performance. Como a junção é uma operação onerosa, recomendamos não usá-la, a menos que seja essencial para seus requisitos de negócios. Certifique-se de que você está utilizando o seu pipeline de dados de forma eficiente, questionando-se sobre o seguinte:

- Você está recalculando uma junção que também é executada em outros trabalhos que você pode reutilizar?
- Você está fazendo uma junção para resolver chaves estrangeiras para valores que não são usados pelos consumidores do seu resultado?

Depois de confirmar que suas operações de junção são essenciais para seus requisitos de negócios, consulte as opções a seguir para otimizar sua junção de forma que atenda às suas necessidades.

Usar o pushdown antes realizar uma junção

Filtre linhas e colunas desnecessárias no DataFrame antes de realizar uma junção. Essa abordagem tem as seguintes vantagens:

- Reduz o volume de transferência de dados durante o shuffle
- Reduz o volume de processamento no executor do Spark
- Reduz o volume de verificação de dados

```
# 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"])
```

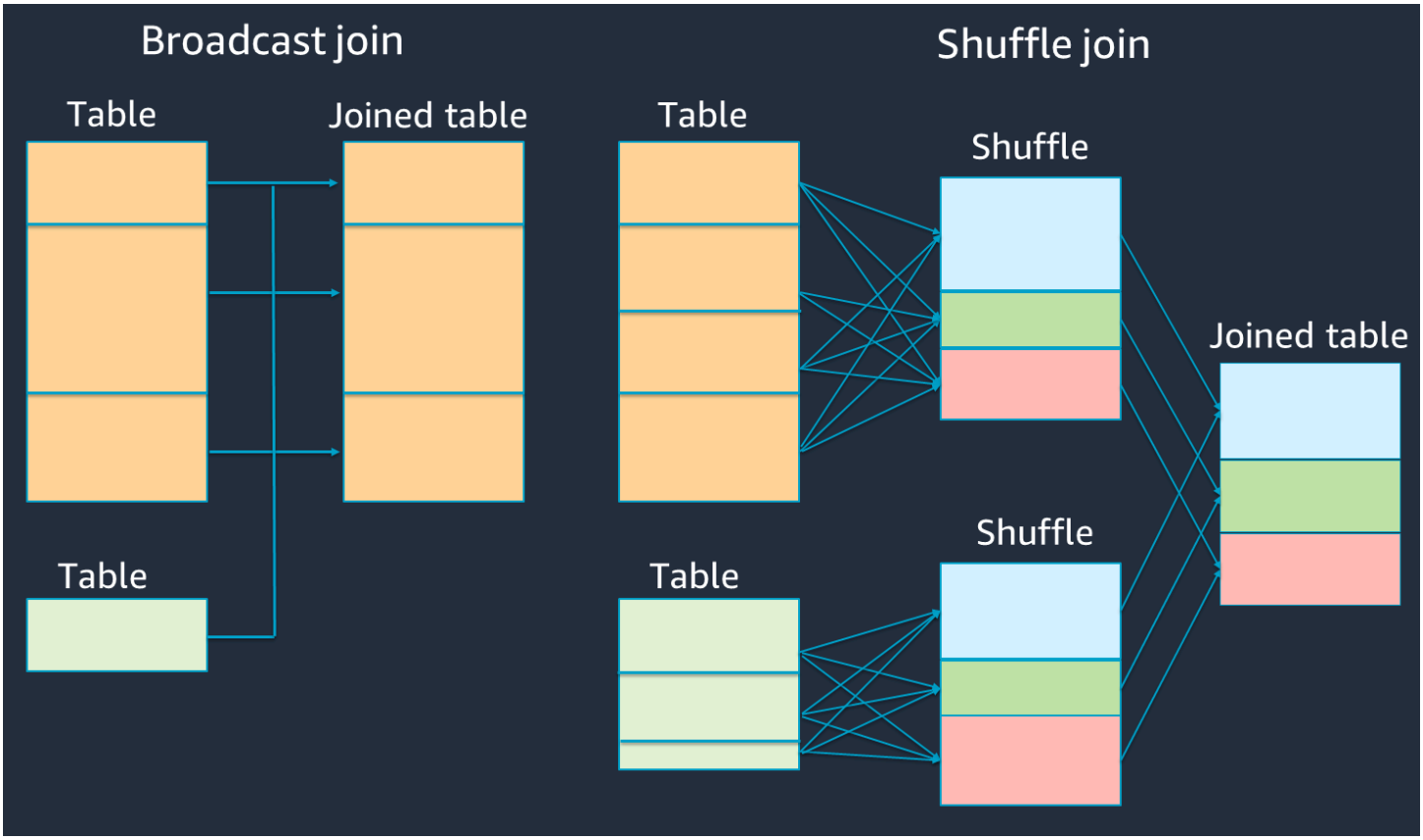
Use DataFrame Join

Tente usar uma API de alto nível do Spark, como SparkSQL e conjuntos de dados DataFrame, em vez da API RDD ou join. DynamicFrame Você pode converter DynamicFrame para DataFrame com uma chamada de método, como`df.toDF()`. Conforme analisado na seção Principais tópicos do Apache Spark, essas operações de junção aproveitam internamente a otimização de consultas pelo otimizador Catalyst.

Dicas e junções shuffle e broadcast hash

O Spark é compatível com dois tipos de junção: junção shuffle e junção broadcast hash. Uma junção broadcast hash não requer operação de shuffle e pode exigir menos processamento do que uma junção shuffle. No entanto, é aplicável somente ao realizar a junção de uma tabela pequena com uma grande. Ao realizar a junção de uma tabela que cabe na memória de um único executor do Spark, considere usar uma junção broadcast hash.

O diagrama a seguir mostra a estrutura de alto nível e as etapas de uma junção broadcast hash e uma junção shuffle.



Os detalhes de cada junção são os seguintes:

- Junção shuffle:
  - A junção shuffle hash combina duas tabelas sem classificação e distribui a junção entre as duas tabelas. É adequado para junções de pequenas tabelas que podem ser armazenadas na memória do executor do Spark.
  - A junção sort-merge distribui as duas tabelas a serem combinadas por chave e as classifica antes juntá-las. É adequado para junções de tabelas grandes.
- Junção broadcast hash:
  - Uma junção broadcast hash envia o RDD ou a tabela menor para cada um dos nós de processamento. Em seguida, ele realiza uma combinação do lado do map com cada partição do RDD ou tabela de maior volume.

    É adequado para junções quando uma de suas RDDs tabelas cabe na memória ou pode ser feita para caber na memória. É vantajoso fazer uma junção broadcast hash sempre que possível, porque não requer um shuffle. Você pode usar uma dica de junção para solicitar uma junção broadcast do Spark, como a seguir.

```
# 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;
```

Para obter mais informações sobre dicas de junção, consulte Join hints.

Na AWS Glue versão 3.0 e versões posteriores, você pode aproveitar automaticamente as junções de hash de transmissão ativando a execução adaptativa de consultas e parâmetros adicionais. A Execução Adaptativa de Consultas converte uma junção sort-merge em uma junção broadcast hash quando as estatísticas de runtime de um dos lados da junção são menores do que o limite de junção broadcast hash adaptativa.

Na AWS Glue versão 3.0, você pode ativar a Execução Adaptativa de Consultas `spark.sql.adaptive.enabled=true` definindo. A Execução Adaptativa de Consultas está habilitada por padrão no AWS Glue 4.0.

Você pode definir parâmetros adicionais relacionados aos shuffles e às junções broadcast hash:

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

Para obter mais informações sobre parâmetros relacionados, consulte Converting sort-merge join to broadcast join.

Na AWS Glue versão 3.0 ou posterior, você pode usar outras dicas de junção para o shuffle para ajustar seu comportamento.

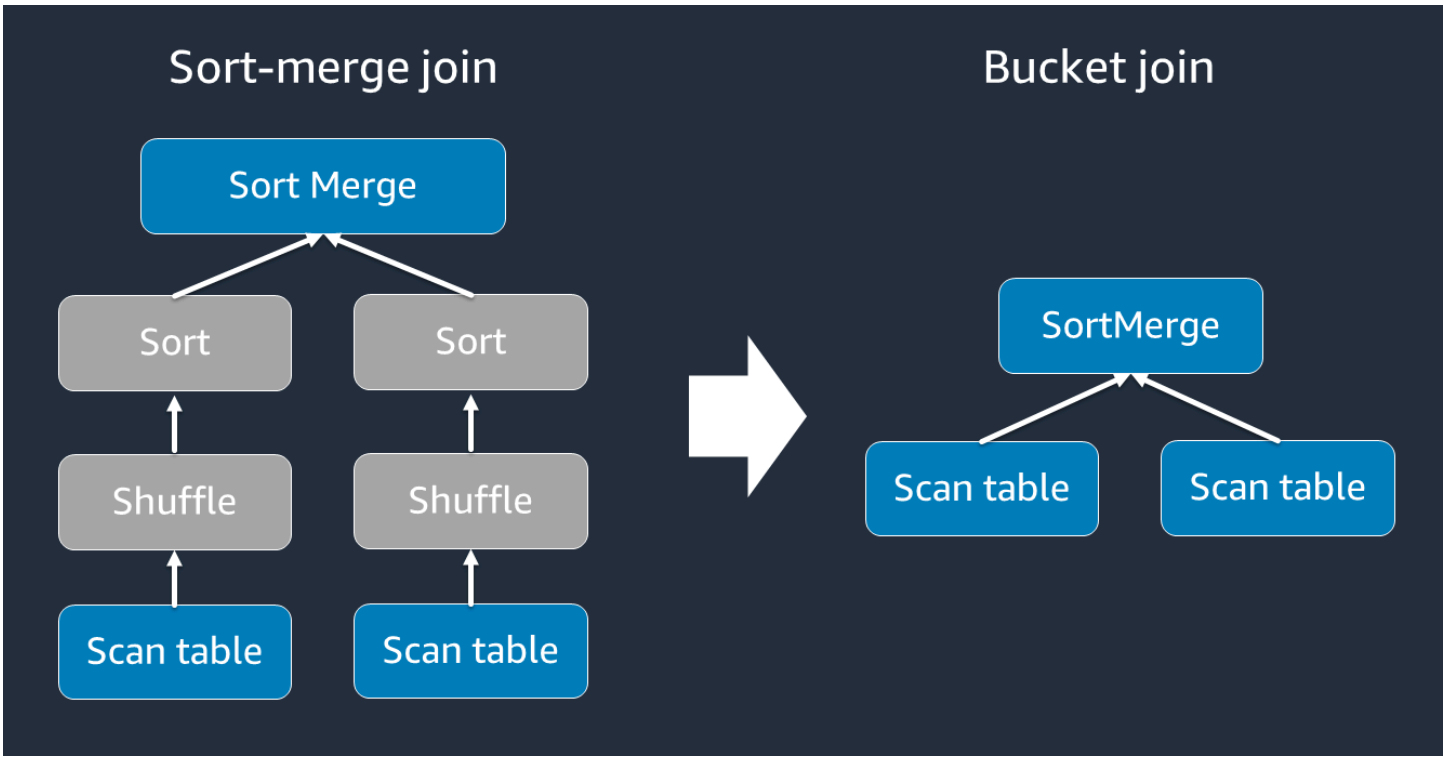
```
-- Join Hints for shuffle sort merge join
SELECT /*+ SHUFFLE_MERGE(t1) /  FROM t1 INNER JOIN t2 ON t1.key = t2.key;
SELECT /*+ MERGEJOIN(t2) /  FROM t1 INNER JOIN t2 ON t1.key = t2.key;
SELECT /*+ MERGE(t1) /  FROM t1 INNER JOIN t2 ON t1.key = t2.key;

-- 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;
```

Usar bucketing

A junção sort-merge requer duas fases: shuffle e classificar, e depois mesclar. Essas duas fases podem sobrecarregar o executor do Spark e causar problemas de OOM e performance quando alguns dos executores estão sendo mesclados e outros estão sendo classificados simultaneamente. Nesses casos, talvez seja possível juntar com eficiência usando bucketing. O bucketing realizará o pré-shuffle e pré-classificará sua entrada nas chaves de junção. Em seguida, gravará esses dados classificados em uma tabela intermediária. O custo das etapas de shuffle e classificação pode ser reduzido ao juntar tabelas grandes definindo as tabelas intermediárias classificadas com antecedência.



As tabelas em buckets são úteis para o seguinte:

- Dados combinados com frequência pela mesma chave, como `account_id`
- Carregar tabelas cumulativas diárias, como tabelas base e delta, que podem ser agrupadas em buckets em uma coluna comum

Você pode criar uma tabela em buckets usando o código a seguir.

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

Repartição DataFrames nas chaves de junção antes da junção

Para reparticionar os dois DataFrames nas chaves de junção antes da junção, use as instruções a seguir.

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

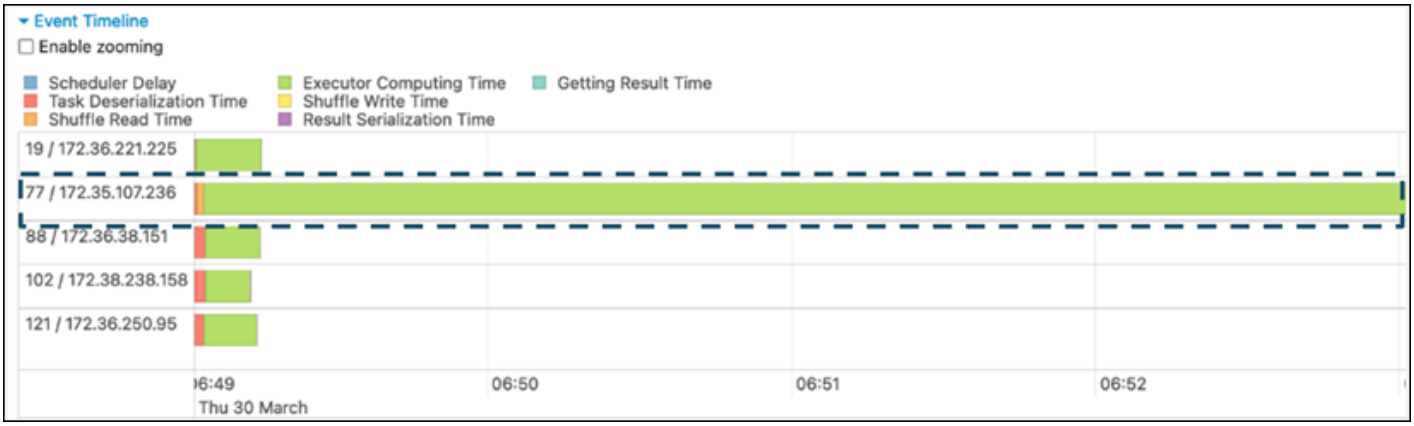
Isso particionará dois (ainda separados) RDDs na chave de junção antes de iniciar a junção. Se os dois RDDs estiverem particionados na mesma chave com o mesmo código de particionamento, o RDD registra que seu plano de união terá uma grande probabilidade de ser colocado no mesmo trabalhador antes de embaralhar para a junção. Isso pode melhorar a performance ao reduzir a atividade da rede e a distorção de dados durante a junção.

Superar distorções de dados

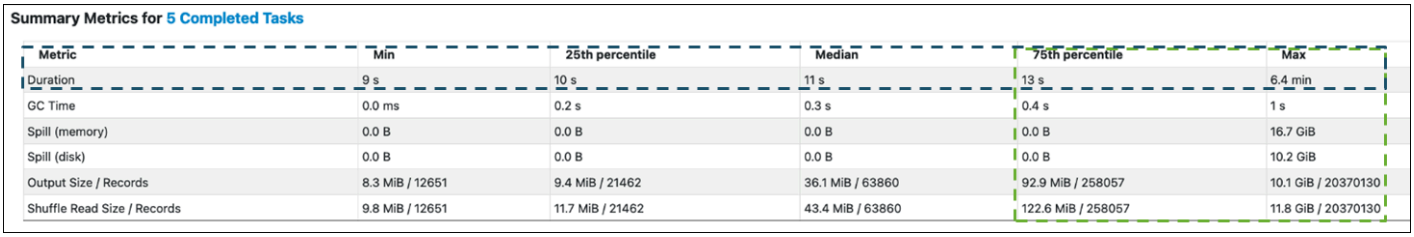
A distorção de dados é uma das causas mais comuns de um gargalo nos trabalhos do Spark. Ela ocorre quando os dados não são distribuídos uniformemente nas partições do RDD. Isso faz com que as tarefas dessa partição demorem muito mais do que outras, atrasando o tempo geral de processamento da aplicação.

Para identificar a distorção de dados, avalie as seguintes métricas na interface do usuário do Spark:

- Na guia Etapa da interface do usuário do Spark, examine a página Cronograma de eventos. Você pode ver uma distribuição desigual de tarefas na captura de tela a seguir. Tarefas distribuídas de forma desigual ou demorando muito para serem executadas podem indicar distorção de dados.



- Outra página importante é o Resumo de métricas, que mostra as estatísticas das tarefas do Spark. A captura de tela a seguir mostra métricas com percentis para Duração, Tempo de GC, Derramamento (memória), Derramamento (disco) e assim por diante.



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

  Quando as tarefas forem distribuídas uniformemente, você verá números semelhantes em todos os percentis. Quando houver distorção de dados, você verá valores muito enviesados em cada percentil. No exemplo, a duração da tarefa é inferior a 13 segundos em Mín., 25.º percentil, Mediana e 75.º percentil. Embora a tarefa Máx. tenha processado 100 vezes mais dados do que o 75.º percentil, sua duração de 6,4 minutos é cerca de 30 vezes maior. Isso significa que pelo menos uma tarefa (ou até 25% das tarefas) demorou muito mais do que o resto das tarefas.

Se você vir distorções de dados, tente o seguinte:

- Se você usa AWS Glue 3.0, ative a Execução de Consulta Adaptativa por meio da configuração`spark.sql.adaptive.enabled=true`. A execução adaptativa de consultas está ativada por padrão na AWS Glue versão 4.0.

  Você também pode usar a Execução Adaptativa de Consultas para a distorção de dados introduzida pelas junções, definindo os seguintes parâmetros relacionados:

  - `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)`

Para obter mais informações, consulte a documentação do Apache Spark.

- Use chaves com uma vasta gama de valores para as chaves de junção. Em uma junção shuffle, as partições são determinadas para cada valor de hash de uma chave. Se a cardinalidade de uma chave de junção for muito baixa, é mais provável que a função hash faça um trabalho ruim ao distribuir seus dados nas partições. Portanto, se sua aplicação e sua lógica de negócios forem compatíveis, considere usar uma chave de cardinalidade mais alta ou uma chave composta.

```
# 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"])
```

Usar cache

Ao usar repetitivo DataFrames, evite operações aleatórias ou computacionais adicionais usando `df.cache()` ou `df.persist()` armazenando em cache os resultados do cálculo na memória e no disco de cada executor do Spark. O Spark também suporta a persistência RDDs em disco ou a replicação em vários nós (nível de armazenamento).

Por exemplo, você pode persistir DataFrames adicionando`df.persist()`. Quando o cache não for mais necessário, você poderá usar `unpersist` para descartar os dados em 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()
```

Remover ações desnecessárias do Spark

Evite executar ações desnecessárias, como `count`, `show` ou `collect`. Conforme analisado na seção Principais tópicos do Apache Spark, o Spark tem carregamento lento. Cada RDD transformado pode ser recalculado toda vez que você executa uma ação nele. Quando você usa várias ações do Spark, vários acessos à origem, cálculos de tarefas e operações de shuffle são executados para cada ação sendo chamada.

Se você não precisar de `collect()` ou de outras ações em seu ambiente comercial, considere removê-las.

> **Note**
>
> Evitar usar a ação `collect()` do Spark em ambientes comerciais o máximo possível. A ação `collect()` retorna todos os resultados de um cálculo no executor do Spark para o driver do Spark, o que pode fazer com que este retorne um erro de OOM. Para evitar um erro de OOM, o Spark define `spark.driver.maxResultSize = 1GB` por padrão, o que limita o tamanho máximo dos dados retornados ao driver do Spark a 1 GB.

Minimizar a sobrecarga de planejamento

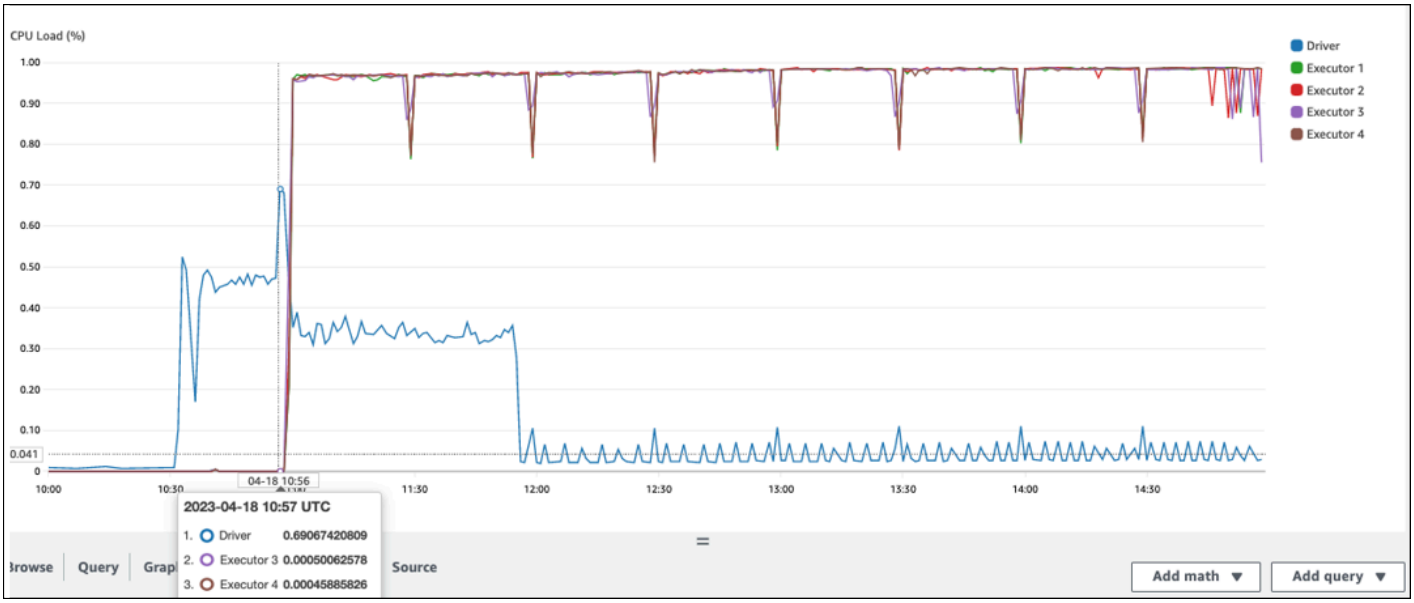
Conforme analisado em Principais tópicos do Apache Spark, o driver do Spark gera o plano de execução. Com base nesse plano, as tarefas são atribuídas ao executor do Spark para processamento distribuído. No entanto, o driver do Spark pode se tornar um gargalo se houver um grande número de arquivos pequenos ou se o AWS Glue Data Catalog contiver um grande número de partições. Para identificar a alta sobrecarga de planejamento, avalie as métricas a seguir.

CloudWatch métricas

Verifique a Carga da CPU e a Utilização da memória nas seguintes situações:

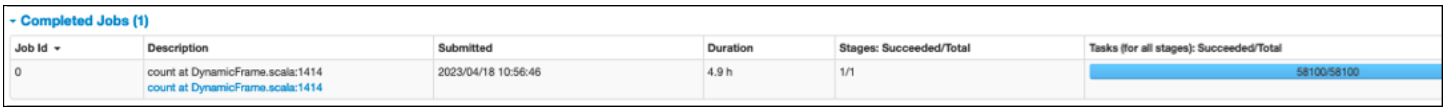
- A Carga da CPU e a Utilização da memória do driver do Spark estão registradas como altas. Normalmente, o driver do Spark não processa seus dados, portanto, a carga da CPU e a utilização da memória não têm picos. No entanto, se a fonte de dados do Amazon S3 tiver muitos arquivos pequenos, listar todos os objetos do S3 e gerenciar um grande número de tarefas pode fazer com que a utilização dos recursos seja alta.

- Há uma grande lacuna antes do início do processamento no executor do Spark. No exemplo de captura de tela a seguir, a carga da CPU do executor do Spark está muito baixa até 10:57, mesmo que o trabalho tenha começado às 10:00. AWS Glue Isso indica que o driver do Spark pode estar demorando muito para gerar um plano de execução. Neste exemplo, recuperar o grande número de partições no Catálogo de Dados e listar o grande número de arquivos pequenos no driver do Spark está demorando muito.



Interface do usuário do Spark

Na guia Trabalho na interface do usuário do Spark, você pode ver o horário de Envio. No exemplo a seguir, o driver do Spark iniciou o job0 às 10:56:46, embora o trabalho tenha começado às 10:00:00. AWS Glue



Você também pode ver o horário das Tarefas (para todas as etapas): Com êxito/Total na guia Trabalho. Nesse caso, o número de tarefas é registrado como `58100`. Conforme explicado na seção do Amazon S3 da página Paralelizar tarefas, o número de tarefas corresponde aproximadamente ao número de objetos do S3. Isso significa que há cerca de 58.100 objetos no Amazon S3.

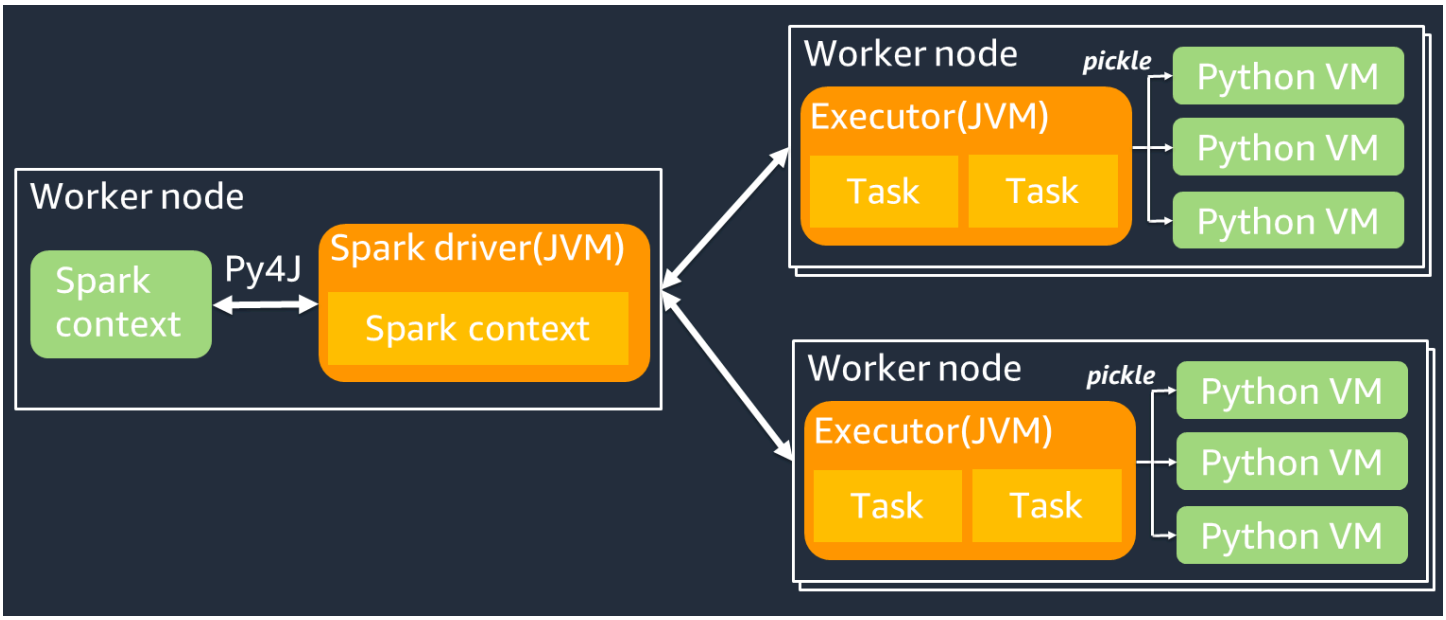
Para obter mais detalhes sobre esse trabalho e o cronograma, revise a guia Etapa. Se você observar um gargalo com o driver do Spark, considere as seguintes soluções:

- Quando o Amazon S3 tem muitos arquivos, considere a orientação sobre paralelismo excessivo na seção Muitas partições da página Paralelizar tarefas.
- Quando o Amazon S3 tem muitas partições, considere a orientação sobre particionamento excessivo na seção Muitas partições do Amazon S3 da página Reduzir a quantidade de dados verificados. Habilite os índices de partição do AWS Glue se houver muitas partições para reduzir a latência na recuperação de metadados de partições do Catálogo de Dados. Para obter mais informações, consulte Melhorar o desempenho da consulta usando índices de AWS Glue partição.
- Quando o JDBC tiver muitas partições, diminua o valor de `hashpartition`.
- Quando o DynamoDB tiver muitas partições, diminua o valor de `dynamodb.splits`.
- Quando os trabalhos de streaming tiverem muitas partições, diminua o número de fragmentos.

Otimizar funções definidas pelo usuário

Funções definidas pelo usuário (UDFs) e `RDD.map` PySpark geralmente degradam significativamente o desempenho. Isso se deve à sobrecarga necessária para representar com precisão seu código Python na implementação subjacente do Scala do Spark.

O diagrama a seguir mostra a arquitetura dos PySpark trabalhos. Quando você usa PySpark, o driver do Spark usa a biblioteca Py4j para chamar métodos Java do Python. Ao chamar o Spark SQL ou funções DataFrame integradas, há pouca diferença de desempenho entre Python e Scala porque as funções são executadas na JVM de cada executor usando um plano de execução otimizado.



Se você usar sua própria lógica do Python, como usar `map/ mapPartitions/ udf`, a tarefa será executada em um ambiente de runtime do Python. O gerenciamento de dois ambientes gera um custo de sobrecarga. Além disso, seus dados na memória devem ser transformados para serem usados pelas funções integradas do ambiente de runtime da JVM. O Pickle é um formato de serialização usado por padrão para a troca entre os runtimes da JVM e do Python. No entanto, o custo desse custo de serialização e desserialização é muito alto, portanto, UDFs escritos em Java ou Scala são mais rápidos que Python. UDFs

Para evitar a sobrecarga de serialização e desserialização PySpark, considere o seguinte:

- Use as funções integradas do Spark SQL — Considere substituir sua própria UDF ou função de mapa pelo Spark SQL ou DataFrame por funções integradas. Ao executar o Spark SQL ou funções DataFrame integradas, há pouca diferença de desempenho entre o Python e o Scala porque as tarefas são gerenciadas na JVM de cada executor.
- Implemente UDFs em Scala ou Java — Considere usar uma UDF escrita em Java ou Scala, porque elas são executadas na JVM.
- Use o Apache Arrow UDFs para cargas de trabalho vetorizadas — Considere usar o baseado em Arrow. UDFs Esse recurso também é conhecido como UDF Vetorizada (Pandas UDF). O Apache Arrow é um formato de dados em memória independente de linguagem que AWS Glue pode ser usado para transferir dados com eficiência entre processos JVM e Python. Atualmente, isso é mais benéfico para usuários de Python que trabalham com Pandas ou dados. NumPy

  A seta é um formato colunar (vetorizado). Seu uso não é automático e pode exigir algumas pequenas alterações na configuração ou no código para aproveitar ao máximo e garantir a compatibilidade. Para obter mais detalhes e limitações, consulte Apache Arrow em PySpark.

  O exemplo a seguir compara uma UDF incremental básica no Python padrão, como uma UDF Vetorizada, e no Spark SQL.

UDF Python padrão

O tempo do exemplo é 3,20 (seg.).

Código de exemplo

```
# 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()
```

Plano de execução

```
== 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)
```

UDF Vetorizada

O tempo do exemplo é 0,59 (seg.).

A UDF Vetorizada é cinco vezes mais rápida do que o exemplo anterior da UDF. Ao verificar `Physical Plan`, você pode ver `ArrowEvalPython`, o que mostra que essa aplicação é vetorizada pelo Apache Arrow. Para habilitar a UDF Vetorizada, você deve especificar `spark.sql.execution.arrow.pyspark.enabled = true` em seu código.

Código de exemplo

```
# 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()
```

Plano de execução

```
== 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)
```

Spark SQL

O tempo do exemplo é 0,087 (seg.).

O Spark SQL é muito mais rápido do que a UDF Vetorizada, porque as tarefas são executadas na JVM de cada executor sem um runtime do Python. Se você puder substituir sua UDF por uma função integrada, recomendamos que o faça.

Código de exemplo

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

Uso de pandas para big data

Se você já está familiarizado com pandas e quer usar o Spark para big data, pode usar a API pandas no Spark. AWS Glue 4.0 e versões posteriores o suportam. Para começar, você pode usar o caderno oficial Quickstart: Pandas API on Spark. Para obter mais informações, consulte a documentação do PySpark.

Recursos

- AWS Glue
- Performance Tuning (Guia do Spark SQL)
- AWS Glue Optimization Workshop

Histórico do documento

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

Alteração	Descrição	Data
Publicação inicial	—	2 de janeiro de 2024

AWS Glossário de orientação prescritiva

A seguir estão os termos comumente usados em estratégias, guias e padrões fornecidos pela Orientação AWS Prescritiva. Para sugerir entradas, use o link Fornecer feedback no final do glossário.

Números

7 Rs

Sete estratégias comuns de migração para mover aplicações para a nuvem. Essas estratégias baseiam-se nos 5 Rs identificados pela Gartner em 2011 e consistem em:

- Refatorar/rearquitetar: mova uma aplicação e modifique sua arquitetura aproveitando ao máximo os recursos nativos de nuvem para melhorar a agilidade, a performance e a escalabilidade. Isso normalmente envolve a portabilidade do sistema operacional e do banco de dados. Exemplo: migrar seu banco de dados Oracle on-premises para o Amazon Aurora Edição Compatível com PostgreSQL.
- Redefinir a plataforma (mover e redefinir [mover e redefinir (lift-and-reshape]): mova uma aplicação para a nuvem e introduza algum nível de otimização a fim de aproveitar os recursos da nuvem. Exemplo: migrar seu banco de dados Oracle on-premises para o Amazon Relational Database Service (Amazon RDS) para Oracle na Nuvem AWS.
- Recomprar (drop and shop): mude para um produto diferente, normalmente migrando de uma licença tradicional para um modelo SaaS. Exemplo: migrar seu sistema de gerenciamento de relacionamento com o cliente (CRM) para o Salesforce.com.
- Redefinir a hospedagem (mover sem alterações [lift-and-shift])mover uma aplicação para a nuvem sem fazer nenhuma alteração a fim de aproveitar os recursos da nuvem. Exemplo: migrar seu banco de dados Oracle on-premises para o Oracle em uma instância do EC2 na Nuvem AWS.
- Realocar (mover o hipervisor sem alterações [hypervisor-level lift-and-shift]): mover a infraestrutura para a nuvem sem comprar novo hardware, reescrever aplicações ou modificar suas operações existentes. Você migra servidores de uma plataforma on-premises para um serviço de nuvem para a mesma plataforma. Exemplo: migrar um Microsoft Hyper-V aplicativo para o. AWS
- Reter (revisitar): mantenha as aplicações em seu ambiente de origem. Isso pode incluir aplicações que exigem grande refatoração, e você deseja adiar esse trabalho para um

momento posterior, e aplicações antigas que você deseja manter porque não há justificativa comercial para migrá-las.

- Retirar: desative ou remova aplicações que não são mais necessárias em seu ambiente de origem.

A

ABAC

Consulte controle de acesso baseado em atributo.

serviços abstraídos

Veja serviços gerenciados.

ACID

Veja atomicidade, consistência, isolamento, durabilidade.

migração ativa-ativa

Um método de migração de banco de dados no qual os bancos de dados de origem e de destino são mantidos em sincronia (por meio de uma ferramenta de replicação bidirecional ou operações de gravação dupla), e ambos os bancos de dados lidam com transações de aplicações conectadas durante a migração. Esse método oferece suporte à migração em lotes pequenos e controlados, em vez de exigir uma substituição única. É mais flexível, mas exige mais trabalho do que a migração ativa-passiva.

migração ativa-passiva

Um método de migração de banco de dados em que os bancos de dados de origem e de destino são mantidos em sincronia, mas somente o banco de dados de origem manipula as transações das aplicações conectadas, enquanto os dados são replicados no banco de dados de destino. O banco de dados de destino não aceita nenhuma transação durante a migração.

AGGREGATE FUNCTION

Uma função SQL que opera em um grupo de linhas e calcula um único valor de retorno para o grupo. Exemplos de funções agregadas incluem `SUM` e `MAX`.

AI

Veja inteligência artificial.

AIOps

Veja operações de inteligência artificial.

anonimização

O processo de excluir permanentemente informações pessoais em um conjunto de dados. A anonimização pode ajudar a proteger a privacidade pessoal. Dados anônimos não são mais considerados dados pessoais.

antipadrões

Uma solução frequentemente usada para um problema recorrente em que a solução é contraproducente, ineficaz ou menos eficaz do que uma alternativa.

controle de aplicações

Uma abordagem de segurança que permite o uso somente de aplicações aprovadas para ajudar a proteger um sistema contra malware.

portfólio de aplicações

Uma coleção de informações detalhadas sobre cada aplicação usada por uma organização, incluindo o custo para criar e manter a aplicação e seu valor comercial. Essas informações são fundamentais para o processo de descoberta e análise de portfólio e ajudam a identificar e priorizar as aplicações a serem migradas, modernizadas e otimizadas.

inteligência artificial (IA)

O campo da ciência da computação que se dedica ao uso de tecnologias de computação para desempenhar funções cognitivas normalmente associadas aos humanos, como aprender, resolver problemas e reconhecer padrões. Para obter mais informações, consulte O que é inteligência artificial?

operações de inteligência artificial (AIOps)

O processo de usar técnicas de machine learning para resolver problemas operacionais, reduzir incidentes operacionais e intervenção humana e aumentar a qualidade do serviço. Para obter mais informações sobre como AIOps é usado na estratégia de AWS migração, consulte o guia de integração de operações.

criptografia assimétrica

Um algoritmo de criptografia que usa um par de chaves, uma chave pública para criptografia e uma chave privada para descriptografia. É possível compartilhar a chave pública porque ela não é usada na descriptografia, mas o acesso à chave privada deve ser altamente restrito.

atomicidade, consistência, isolamento, durabilidade (ACID)

Um conjunto de propriedades de software que garantem a validade dos dados e a confiabilidade operacional de um banco de dados, mesmo no caso de erros, falhas de energia ou outros problemas.

controle de acesso por atributo (ABAC)

A prática de criar permissões minuciosas com base nos atributos do usuário, como departamento, cargo e nome da equipe. Para obter mais informações, consulte ABAC AWS na documentação AWS Identity and Access Management (IAM).

fonte de dados autorizada

Um local onde você armazena a versão principal dos dados, que é considerada a fonte de informações mais confiável. Você pode copiar dados da fonte de dados autorizada para outros locais com o objetivo de processar ou modificar os dados, como anonimizá-los, redigi-los ou pseudonimizá-los.

Zona de disponibilidade

Um local distinto dentro de um Região da AWS que está isolado de falhas em outras zonas de disponibilidade e fornece conectividade de rede barata e de baixa latência a outras zonas de disponibilidade na mesma região.

AWS Estrutura de adoção da nuvem (AWS CAF)

Uma estrutura de diretrizes e melhores práticas AWS para ajudar as organizações a desenvolver um plano eficiente e eficaz para migrar com sucesso para a nuvem. AWS O CAF organiza a orientação em seis áreas de foco chamadas perspectivas: negócios, pessoas, governança, plataforma, segurança e operações. As perspectivas de negócios, pessoas e governança têm como foco habilidades e processos de negócios; as perspectivas de plataforma, segurança e operações concentram-se em habilidades e processos técnicos. Por exemplo, a perspectiva das pessoas tem como alvo as partes interessadas que lidam com recursos humanos (RH), funções de pessoal e gerenciamento de pessoal. Nessa perspectiva, o AWS CAF fornece orientação para desenvolvimento, treinamento e comunicação de pessoas para ajudar a preparar a organização

para a adoção bem-sucedida da nuvem. Para obter mais informações, consulte o site da AWS CAF e o whitepaper da AWS CAF.

AWS Estrutura de qualificação da carga de trabalho (AWS WQF)

Uma ferramenta que avalia as cargas de trabalho de migração do banco de dados, recomenda estratégias de migração e fornece estimativas de trabalho. AWS O WQF está incluído com AWS Schema Conversion Tool ()AWS SCT. Ela analisa esquemas de banco de dados e objetos de código, código de aplicações, dependências e características de performance, além de fornecer relatórios de avaliação.

B

bot malicioso

Um bot destinado a causar disrupção ou danos a indivíduos ou organizações.

BCP

Veja planejamento de continuidade de negócios

gráfico de comportamento

Uma visualização unificada e interativa do comportamento e das interações de recursos ao longo do tempo. É possível usar um gráfico de comportamento com o Amazon Detective para examinar tentativas de login malsucedidas, chamadas de API suspeitas e ações similares. Para obter mais informações, consulte Dados em um gráfico de comportamento na documentação do Detective.

sistema big-endian

Um sistema que armazena o byte mais significativo antes. Veja também endianness.

classificação binária

Um processo que prevê um resultado binário (uma de duas classes possíveis). Por exemplo, seu modelo de ML pode precisar prever problemas como “Este e-mail é ou não é spam?” ou “Este produto é um livro ou um carro?”

filtro de bloom

Uma estrutura de dados probabilística e eficiente em termos de memória que é usada para testar se um elemento é membro de um conjunto.

blue/green deployment (implantação azul/verde)

Uma estratégia de implantação em que você cria dois ambientes separados, mas idênticos. Você executa a versão atual da aplicação em um ambiente (azul) e a nova versão da aplicação no outro ambiente (verde). Essa estratégia ajuda você a reverter rapidamente com o mínimo de impacto.

bot

Uma aplicação de software que executa tarefas automatizadas na internet e simula a atividade ou interação humana. Alguns bots são úteis ou benéficos, como crawlers da web que indexam informações na internet. Outros bots, conhecidos como bots maliciosos, têm como objetivo causar disrupção ou danos a indivíduos ou organizações.

botnet

Redes de bots infectadas por malware e sob o controle de uma única parte, conhecidas como bot herder ou operador de bots. Os botnets são o mecanismo mais conhecido para escalar bots e seu impacto.

ramo

Uma área contida de um repositório de código. A primeira ramificação criada em um repositório é a ramificação principal. Você pode criar uma nova ramificação a partir de uma ramificação existente e, em seguida, desenvolver recursos ou corrigir bugs na nova ramificação. Uma ramificação que você cria para gerar um recurso é comumente chamada de ramificação de recurso. Quando o recurso estiver pronto para lançamento, você mesclará a ramificação do recurso de volta com a ramificação principal. Para obter mais informações, consulte Sobre filiais (GitHub documentação).

Acesso de emergência

Em circunstâncias excepcionais e por meio de um processo aprovado, um meio rápido para um usuário obter acesso a um Conta da AWS que ele normalmente não tem permissão para acessar. Para obter mais informações, consulte o indicador Implement break-glass procedures nas orientações do AWS Well-Architected.

estratégia brownfield

A infraestrutura existente em seu ambiente. Ao adotar uma estratégia brownfield para uma arquitetura de sistema, você desenvolve a arquitetura de acordo com as restrições dos sistemas e da infraestrutura atuais. Se estiver expandindo a infraestrutura existente, poderá combinar as estratégias brownfield e greenfield.

cache do buffer

A área da memória em que os dados acessados com mais frequência são armazenados.

capacidade de negócios

O que uma empresa faz para gerar valor (por exemplo, vendas, atendimento ao cliente ou marketing). As arquiteturas de microsserviços e as decisões de desenvolvimento podem ser orientadas por recursos de negócios. Para obter mais informações, consulte a seção Organizados de acordo com as capacidades de negócios do whitepaper Executar microsserviços conteinerizados na AWS.

planejamento de continuidade de negócios (BCP)

Um plano que aborda o impacto potencial de um evento disruptivo, como uma migração em grande escala, nas operações e permite que uma empresa retome as operações rapidamente.

C

CAF

Veja AWS Cloud Adoption Framework.

implantação canário

O lançamento lento e incremental de uma versão para usuários finais. Quando estiver confiante, você implanta a nova versão e substitui a versão atual por completo.

CCoE

Veja Centro de Excelência da Nuvem.

CDC

Veja captura de dados de alteração.

captura de dados de alterações (CDC)

O processo de rastrear alterações em uma fonte de dados, como uma tabela de banco de dados, e registrar metadados sobre a alteração. É possível usar o CDC para várias finalidades, como auditar ou replicar alterações em um sistema de destino para manter a sincronização.

engenharia do caos

Introduzir intencionalmente falhas ou eventos disruptivos para testar a resiliência de um sistema. Você pode usar AWS Fault Injection Service (AWS FIS) para realizar experimentos que estressam suas AWS cargas de trabalho e avaliar sua resposta.

CI/CD

Veja integração e entrega contínuas.

classificação

Um processo de categorização que ajuda a gerar previsões. Os modelos de ML para problemas de classificação predizem um valor discreto. Os valores discretos são sempre diferentes uns dos outros. Por exemplo, um modelo pode precisar avaliar se há ou não um carro em uma imagem.

criptografia no lado do cliente

Criptografia de dados localmente, antes que o alvo os AWS service (Serviço da AWS) receba.

Centro de excelência em nuvem (CCoE)

Uma equipe multidisciplinar que impulsiona os esforços de adoção da nuvem em toda a organização, incluindo o desenvolvimento de práticas recomendas de nuvem, a mobilização de recursos, o estabelecimento de cronogramas de migração e a liderança da organização em transformações em grande escala. Para obter mais informações, consulte as publicações CCo E no blog de estratégia Nuvem AWS corporativa.

computação em nuvem

A tecnologia de nuvem normalmente usada para armazenamento de dados remoto e gerenciamento de dispositivos de IoT. A computação em nuvem é normalmente conectada à tecnologia de computação de borda.

modelo operacional em nuvem

Em uma organização de TI, o modelo operacional usado para criar, amadurecer e otimizar um ou mais ambientes de nuvem. Para obter mais informações, consulte Criar seu modelo operacional de nuvem.

estágios de adoção da nuvem

As quatro fases pelas quais as organizações normalmente passam ao migrar para a Nuvem AWS:

- Projeto: executar alguns projetos relacionados à nuvem para fins de prova de conceito e aprendizado
- Fundação — Fazer investimentos fundamentais para escalar sua adoção da nuvem (por exemplo, criar uma landing zone, definir um CCo E, estabelecer um modelo de operações)
- Migração: migrar aplicações individuais
- Reinvenção: otimizar produtos e serviços e inovar na nuvem

Esses estágios foram definidos por Stephen Orban na postagem do blog The Journey Toward Cloud-First & the Stages of Adoption no blog de estratégia Nuvem AWS empresarial. Para obter informações sobre como eles se relacionam com a estratégia de AWS migração, consulte o guia de preparação para migração.

CMDB

Veja banco de dados de gerenciamento de configuração.

repositório de código

Um local onde o código-fonte e outros ativos, como documentação, amostras e scripts, são armazenados e atualizados por meio de processos de controle de versão. Os repositórios de nuvem comuns incluem o GitHub ou o Bitbucket Cloud. Cada versão do código é chamada de ramificação. Em uma estrutura de microsserviços, cada repositório é dedicado a uma única peça de funcionalidade. Um único pipeline de CI/CD pode usar vários repositórios.

cache frio

Um cache de buffer que está vazio, não está bem preenchido ou contém dados obsoletos ou irrelevantes. Isso afeta a performance porque a instância do banco de dados deve ler da memória principal ou do disco, um processo que é mais lento do que a leitura do cache do buffer.

dados frios

Dados que raramente são acessados e geralmente são históricos. Ao consultar esse tipo de dados, consultas lentas geralmente são aceitáveis. Mover esses dados para níveis ou classes de armazenamento de baixo desempenho e menos caros pode reduzir os custos.

visão computacional (CV)

Um campo de IA que usa machine learning para analisar e extrair informações de formatos visuais, como vídeos e imagens digitais. Por exemplo, a Amazon SageMaker AI fornece algoritmos de processamento de imagem para CV.

desvio de configuração

Em uma workload, uma alteração de configuração em relação ao estado esperado. Isso pode fazer com que a workload se torne incompatível e, normalmente, é gradual e não intencional.

banco de dados de gerenciamento de configuração (CMDB)

Um repositório que armazena e gerencia informações sobre um banco de dados e seu ambiente de TI, incluindo componentes de hardware e software e suas configurações. Normalmente, os dados de um CMDB são usados no estágio de descoberta e análise do portfólio da migração.

pacote de conformidade

Uma coleção de AWS Config regras e ações de remediação que você pode montar para personalizar suas verificações de conformidade e segurança. Você pode implantar um pacote de conformidade como uma entidade única em uma Conta da AWS região ou em uma organização usando um modelo YAML. Para obter mais informações, consulte Pacotes de conformidade na documentação. AWS Config

integração contínua e entrega contínua (CI/CD)

O processo de automatizar os estágios de origem, criação, teste, preparação e produção do processo de lançamento do software. CI/CD é comumente descrito como um pipeline. CI/CD pode ajudá-lo a automatizar processos, melhorar a produtividade, melhorar a qualidade do código e entregar com mais rapidez. Para obter mais informações, consulte Benefícios da entrega contínua. CD também pode significar implantação contínua. Para obter mais informações, consulte Entrega contínua versus implantação contínua.

CV

Veja visão computacional.

D

dados em repouso

Dados estacionários em sua rede, por exemplo, dados que estão em um armazenamento.

classificação de dados

Um processo para identificar e categorizar os dados em sua rede com base em criticalidade e confidencialidade. É um componente crítico de qualquer estratégia de gerenciamento de riscos de

segurança cibernética, pois ajuda a determinar os controles adequados de proteção e retenção para os dados. A classificação de dados é um componente do pilar de segurança no AWS Well-Architected Framework. Para obter mais informações, consulte Classificação de dados.

desvio de dados

Uma variação significativa entre os dados de produção e os dados usados para treinar um modelo de ML ou uma alteração significativa nos dados de entrada ao longo do tempo. O desvio de dados pode reduzir a qualidade geral, a precisão e a imparcialidade das previsões do modelo de ML.

dados em trânsito

Dados que estão se movendo ativamente pela sua rede, como entre os recursos da rede.

data mesh

Um framework de arquitetura que fornece propriedade de dados distribuída e descentralizada com gerenciamento e governança centralizados.

minimização de dados

O princípio de coletar e processar apenas os dados estritamente necessários. Praticar a minimização de dados no Nuvem AWS pode reduzir os riscos de privacidade, os custos e a pegada de carbono de sua análise.

perímetro de dados

Um conjunto de proteções preventivas em seu AWS ambiente que ajudam a garantir que somente identidades confiáveis acessem recursos confiáveis das redes esperadas. Para obter mais informações, consulte Construindo um perímetro de dados em. AWS

pré-processamento de dados

A transformação de dados brutos em um formato que seja facilmente analisado por seu modelo de ML. O pré-processamento de dados pode significar a remoção de determinadas colunas ou linhas e o tratamento de valores ausentes, inconsistentes ou duplicados.

proveniência dos dados

O processo de rastrear a origem e o histórico dos dados ao longo de seu ciclo de vida, por exemplo, como os dados foram gerados, transmitidos e armazenados.

titular dos dados

Um indivíduo cujos dados estão sendo coletados e processados.

data warehouse

Um sistema de gerenciamento de dados compatível com business intelligence, como analytics. Os data warehouses geralmente contêm grandes quantidades de dados históricos e geralmente são usados para consultas e análises.

linguagem de definição de dados (DDL)

Instruções ou comandos para criar ou modificar a estrutura de tabelas e objetos em um banco de dados.

linguagem de manipulação de dados (DML)

Instruções ou comandos para modificar (inserir, atualizar e excluir) informações em um banco de dados.

DDL

Veja linguagem de definição de banco de dados.

deep ensemble

A combinação de vários modelos de aprendizado profundo para gerar previsões. Os deep ensembles podem ser usados para produzir uma previsão mais precisa ou para estimar a incerteza nas previsões.

Aprendizado profundo

Um subcampo do ML que usa várias camadas de redes neurais artificiais para identificar o mapeamento entre os dados de entrada e as variáveis-alvo de interesse.

defense-in-depth

Uma abordagem de segurança da informação na qual uma série de mecanismos e controles de segurança são cuidadosamente distribuídos por toda a rede de computadores para proteger a confidencialidade, a integridade e a disponibilidade da rede e dos dados nela contidos. Ao adotar essa estratégia AWS, você adiciona vários controles em diferentes camadas da AWS Organizations estrutura para ajudar a proteger os recursos. Por exemplo, uma defense-in-depth abordagem pode combinar autenticação multifatorial, segmentação de rede e criptografia.

administrador delegado

Em AWS Organizations, um serviço compatível pode registrar uma conta de AWS membro para administrar as contas da organização e gerenciar as permissões desse serviço. Essa conta

é chamada de administrador delegado para esse serviço Para obter mais informações e uma lista de serviços compatíveis, consulte Serviços que funcionam com o AWS Organizations na documentação do AWS Organizations .

implantação

O processo de criar uma aplicação, novos recursos ou correções de código disponíveis no ambiente de destino. A implantação envolve a implementação de mudanças em uma base de código e, em seguida, a criação e execução dessa base de código nos ambientes da aplicação

ambiente de desenvolvimento

Veja ambiente.

controle detectivo

Um controle de segurança projetado para detectar, registrar e alertar após a ocorrência de um evento. Esses controles são uma segunda linha de defesa, alertando você sobre eventos de segurança que contornaram os controles preventivos em vigor. Para obter mais informações, consulte Controles detectivos em Como implementar controles de segurança na AWS.

mapeamento do fluxo de valor de desenvolvimento (DVSM)

Um processo usado para identificar e priorizar restrições que afetam negativamente a velocidade e a qualidade em um ciclo de vida de desenvolvimento de software. O DVSM estende o processo de mapeamento do fluxo de valor originalmente projetado para práticas de manufatura enxuta. Ele se concentra nas etapas e equipes necessárias para criar e movimentar valor por meio do processo de desenvolvimento de software.

gêmeo digital

Uma representação virtual de um sistema real, como um prédio, fábrica, equipamento industrial ou linha de produção. Os gêmeos digitais oferecem suporte à manutenção preditiva, ao monitoramento remoto e à otimização da produção.

tabela de dimensões

Em um esquema em estrela, uma tabela menor que contém atributos de dados sobre dados quantitativos em uma tabela de fatos. Os atributos da tabela de dimensões geralmente são campos de texto ou números discretos que se comportam como texto. Esses atributos normalmente são usados para restringir consultas, filtrar e rotular conjuntos de resultados.

desastre

Um evento que impede que uma workload ou sistema cumpra seus objetivos de negócios em seu local principal de implantação. Esses eventos podem ser desastres naturais, falhas técnicas ou o resultado de ações humanas, como configuração incorreta não intencional ou ataque de malware.

Recuperação de desastres (RD)

A estratégia e o processo que você usa para minimizar o tempo de inatividade e a perda de dados causados por um disastre. Para obter mais informações, consulte Recuperação de desastres de cargas de trabalho em AWS: Recuperação na nuvem no AWS Well-Architected Framework.

DML

Veja linguagem de manipulação de banco de dados.

design orientado por domínio

Uma abordagem ao desenvolvimento de um sistema de software complexo conectando seus componentes aos domínios em evolução, ou principais metas de negócios, atendidos por cada componente. Esse conceito foi introduzido por Eric Evans em seu livro, Design orientado por domínio: lidando com a complexidade no coração do software (Boston: Addison-Wesley Professional, 2003). Para obter informações sobre como usar o design orientado por domínio com o padrão strangler fig, consulte Modernizar incrementalmente os serviços web herdados do Microsoft ASP.NET (ASMX) usando contêineres e o Amazon API Gateway.

DR

Veja recuperação de desastres.

Detecção da oscilação

Rastreamento de desvios de uma configuração de linha de base. Por exemplo, você pode usar AWS CloudFormation para detectar desvios nos recursos do sistema ou AWS Control Tower para detectar mudanças em seu landing zone que possam afetar a conformidade com os requisitos de governança.

DVSM

Veja mapeamento do fluxo de valor de desenvolvimento.

E

EDA

Veja análise exploratória de dados.

EDI

Veja intercâmbio eletrônico de dados.

computação de borda

A tecnologia que aumenta o poder computacional de dispositivos inteligentes nas bordas de uma rede de IoT. Quando comparada com a computação em nuvem, a computação de borda pode reduzir a latência da comunicação e melhorar o tempo de resposta.

intercâmbio eletrônico de dados (EDI)

A troca automatizada de documentos comerciais entre organizações. Para obter mais informações, consulte O que é EDI (Intercâmbio eletrônico de dados)?.

criptografia

Um processo de computação que transforma dados de texto simples, legíveis por humanos, em texto cifrado.

chave de criptografia

Uma sequência criptográfica de bits aleatórios que é gerada por um algoritmo de criptografia. As chaves podem variar em tamanho, e cada chave foi projetada para ser imprevisível e exclusiva.

endianismo

A ordem na qual os bytes são armazenados na memória do computador. Os sistemas big-endian armazenam o byte mais significativo antes. Os sistemas little-endian armazenam o byte menos significativo antes.

endpoint

Veja endpoint de serviço.

serviço de endpoint

Um serviço que pode ser hospedado em uma nuvem privada virtual (VPC) para ser compartilhado com outros usuários. Você pode criar um serviço de endpoint com AWS PrivateLink e conceder permissões a outros diretores Contas da AWS ou a AWS Identity and Access Management (IAM).

Essas contas ou entidades principais podem se conectar ao serviço de endpoint de maneira privada criando endpoints da VPC de interface. Para obter mais informações, consulte Criar um serviço de endpoint na documentação do Amazon Virtual Private Cloud (Amazon VPC).

planejamento de recursos empresariais (ERP)

Um sistema que automatiza e gerencia os principais processos de negócios (como contabilidade, MES e gerenciamento de projetos) para uma empresa.

criptografia envelopada

O processo de criptografar uma chave de criptografia com outra chave de criptografia. Para obter mais informações, consulte Criptografia de envelope na documentação AWS Key Management Service (AWS KMS).

ambiente

Uma instância de uma aplicação em execução. Estes são tipos comuns de ambientes na computação em nuvem:

- ambiente de desenvolvimento: uma instância de uma aplicação em execução que está disponível somente para a equipe principal responsável pela manutenção da aplicação. Ambientes de desenvolvimento são usados para testar mudanças antes de promovê-las para ambientes superiores. Esse tipo de ambiente às vezes é chamado de ambiente de teste.
- ambientes inferiores: todos os ambientes de desenvolvimento para uma aplicação, como aqueles usados para compilações e testes iniciais.
- ambiente de produção: uma instância de uma aplicação em execução que os usuários finais podem acessar. Em um CI/CD pipeline, o ambiente de produção é o último ambiente de implantação.
- ambientes superiores: todos os ambientes que podem ser acessados por usuários que não sejam a equipe principal de desenvolvimento. Isso pode incluir um ambiente de produção, ambientes de pré-produção e ambientes para testes de aceitação do usuário.

epic

Em metodologias ágeis, categorias funcionais que ajudam a organizar e priorizar seu trabalho. Os epics fornecem uma descrição de alto nível dos requisitos e das tarefas de implementação. Por exemplo, os épicos de segurança AWS da CAF incluem gerenciamento de identidade e acesso, controles de detetive, segurança de infraestrutura, proteção de dados e resposta a incidentes. Para obter mais informações sobre epics na estratégia de migração da AWS , consulte o guia de implementação do programa.

ERP

Veja planejamento de recursos empresariais.

análise exploratória de dados (EDA)

O processo de analisar um conjunto de dados para entender suas principais características. Você coleta ou agrega dados e, em seguida, realiza investigações iniciais para encontrar padrões, detectar anomalias e verificar suposições. O EDA é realizado por meio do cálculo de estatísticas resumidas e da criação de visualizações de dados.

F

tabela de fatos

A tabela central em um esquema em estrela. Ela armazena dados quantitativos sobre as operações comerciais. Normalmente, uma tabela de fatos contém dois tipos de colunas: as que contêm medidas e as que contêm uma chave externa para uma tabela de dimensões.

Antecipar-se à falha

Uma filosofia que usa testes frequentes e incrementais para reduzir o ciclo de vida do desenvolvimento. É uma parte essencial de uma abordagem ágil.

delimitação de isolamento contra falhas

No Nuvem AWS, um limite, como uma zona de disponibilidade, Região da AWS um plano de controle ou um plano de dados, que limita o efeito de uma falha e ajuda a melhorar a resiliência das cargas de trabalho. Para obter mais informações, consulte AWS Fault Isolation Boundaries.

ramificação de recursos

Veja ramificação.

recursos

Os dados de entrada usados para fazer uma previsão. Por exemplo, em um contexto de manufatura, os recursos podem ser imagens capturadas periodicamente na linha de fabricação.

importância do recurso

O quanto um recurso é importante para as previsões de um modelo. Isso geralmente é expresso como uma pontuação numérica que pode ser calculada por meio de várias técnicas, como

Shapley Additive Explanations (SHAP) e gradientes integrados. Para obter mais informações, consulte Interpretabilidade do modelo de aprendizado de máquina com AWS.

transformação de recursos

O processo de otimizar dados para o processo de ML, incluindo enriquecer dados com fontes adicionais, escalar valores ou extrair vários conjuntos de informações de um único campo de dados. Isso permite que o modelo de ML se beneficie dos dados. Por exemplo, se a data “2021-05-27 00:15:37” for dividida em “2021”, “maio”, “quinta” e “15”, isso poderá ajudar o algoritmo de aprendizado a aprender padrões diferenciados associados a diferentes componentes de dados.

prompt few shot

Fornecer a um LLM um pequeno número de exemplos que demonstram a tarefa e o resultado desejado antes de solicitar que ele execute uma tarefa semelhante. Essa técnica é uma aplicação do aprendizado em contexto, em que os modelos aprendem com exemplos (shots) incorporados aos prompts. Prompts few-shot podem ser eficazes para tarefas que exigem formatação, raciocínio ou conhecimento de domínio específicos. Veja também prompts zero-shot.

FGAC

Veja controle de acesso refinado.

Controle de acesso refinado (FGAC)

O uso de várias condições para permitir ou negar uma solicitação de acesso.

migração flash-cut

Um método de migração de banco de dados que usa replicação contínua de dados via captura de dados de alteração para migrar os dados no menor tempo possível, em vez de usar uma abordagem em fases. O objetivo é reduzir ao mínimo o tempo de inatividade.

FM

Veja modelo de base.

modelo de base (FM)

Uma grande rede neural de aprendizado profundo que vem treinando em grandes conjuntos de dados generalizados e não rotulados. FMs são capazes de realizar uma ampla variedade de tarefas gerais, como entender a linguagem, gerar texto e imagens e conversar em linguagem natural. Para obter mais informações, consulte O que são modelos de base?.

G

IA generativa

Um subconjunto de modelos de IA que foram treinados em grandes quantidades de dados e que podem usar um simples prompt de texto para criar novos artefatos e conteúdo, como imagens, vídeos, texto e áudio. Para obter mais informações, consulte O que é IA generativa?.

bloqueio geográfico

Veja restrições geográficas.

restrições geográficas (bloqueio geográfico)

Na Amazon CloudFront, uma opção para impedir que usuários em países específicos acessem distribuições de conteúdo. É possível usar uma lista de permissões ou uma lista de bloqueios para especificar países aprovados e banidos. Para obter mais informações, consulte Restringir a distribuição geográfica do seu conteúdo na CloudFront documentação.

Fluxo de trabalho do GitFlow

Uma abordagem na qual ambientes inferiores e superiores usam ramificações diferentes em um repositório de código-fonte. O fluxo de trabalho do Gitflow é considerado legado, e o fluxo de trabalho trunk-based é a abordagem moderna e preferencial.

golden image

Um snapshot de um sistema ou software usado como modelo para implantar novas instâncias desse sistema ou software. Por exemplo, na manufatura, uma golden image pode ser usada para provisionar software em vários dispositivos e ajudar a melhorar a velocidade, a escalabilidade e a produtividade nas operações de fabricação de dispositivos.

estratégia greenfield

A ausência de infraestrutura existente em um novo ambiente. Ao adotar uma estratégia greenfield para uma arquitetura de sistema, é possível selecionar todas as novas tecnologias sem a restrição da compatibilidade com a infraestrutura existente, também conhecida como brownfield. Se estiver expandindo a infraestrutura existente, poderá combinar as estratégias brownfield e greenfield.

barreira de proteção

Uma regra de alto nível que ajuda a governar recursos, políticas e conformidade em todas as unidades organizacionais ()OUs. Barreiras de proteção preventivas impõem políticas para

garantir o alinhamento a padrões de conformidade. Elas são implementadas usando políticas de controle de serviço e limites de permissões do IAM. Barreiras de proteção detectivas detectam violações de políticas e problemas de conformidade e geram alertas para remediação. Eles são implementados usando AWS Config, AWS Security Hub CSPM, Amazon GuardDuty AWS Trusted Advisor, Amazon Inspector e verificações personalizadas AWS Lambda .

H

HA

Veja alta disponibilidade.

migração heterogênea de bancos de dados

Migrar seu banco de dados de origem para um banco de dados de destino que usa um mecanismo de banco de dados diferente (por exemplo, Oracle para Amazon Aurora). A migração heterogênea geralmente faz parte de um esforço de redefinição da arquitetura, e converter o esquema pode ser uma tarefa complexa. O AWS fornece o AWS SCT para ajudar nas conversões de esquemas.

alta disponibilidade (HA)

A capacidade de uma workload operar continuamente, sem intervenção, em caso de desafios ou desastres. Os sistemas AH são projetados para realizar o failover automático, oferecer consistentemente desempenho de alta qualidade e lidar com diferentes cargas e falhas com impacto mínimo no desempenho.

modernização de historiador

Uma abordagem usada para modernizar e atualizar os sistemas de tecnologia operacional (OT) para melhor atender às necessidades do setor de manufatura. Um historiador é um tipo de banco de dados usado para coletar e armazenar dados de várias fontes em uma fábrica.

dados de hold-out

Uma parte dos dados históricos rotulados que são retidos de um conjunto de dados usado para treinar um modelo de machine learning. Você pode usar dados de hold-out para avaliar a performance do modelo comparando as predições do modelo com os dados de retenção.

migração homogênea de bancos de dados

Migrar seu banco de dados de origem para um banco de dados de destino que compartilha o mesmo mecanismo de banco de dados (por exemplo, Microsoft SQL Server para Amazon RDS para SQL Server). A migração homogênea geralmente faz parte de um esforço de redefinição da hospedagem ou da plataforma. É possível usar utilitários de banco de dados nativos para migrar o esquema.

dados quentes

Dados acessados com frequência, como dados em tempo real ou dados translacionais recentes. Esses dados normalmente exigem uma camada ou classe de armazenamento de alto desempenho para fornecer respostas rápidas às consultas.

hotfix

Uma correção urgente para um problema crítico em um ambiente de produção. Devido à sua urgência, um hotfix geralmente é feito fora do fluxo de trabalho normal de DevOps lançamento.

período de hipercuidados

Imediatamente após a substituição, o período em que uma equipe de migração gerencia e monitora as aplicações migradas na nuvem para resolver quaisquer problemas. Normalmente, a duração desse período é de 1 a 4 dias. No final do período de hipercuidados, a equipe de migração normalmente transfere a responsabilidade pelas aplicações para a equipe de operações de nuvem.

eu

IaC

Veja infraestrutura como código.

Política baseada em identidade

Uma política anexada a um ou mais diretores do IAM que define suas permissões no Nuvem AWS ambiente.

aplicação ociosa

Uma aplicação que tem um uso médio de CPU e memória entre 5 e 20% em um período de 90 dias. Em um projeto de migração, é comum retirar essas aplicações ou retê-las on-premises.

IIoT

Veja Internet das Coisas Industrial.

infraestrutura imutável

Um modelo que implanta uma nova infraestrutura para workloads de produção em vez de atualizar, aplicar patches ou modificar a infraestrutura existente. Infraestruturas imutáveis são inerentemente mais consistentes, confiáveis e preditivas do que infraestruturas mutáveis. Para obter mais informações, consulte a prática recomendada Implantar usando infraestrutura imutável no AWS Well-Architected Framework.

VPC de entrada (admissão)

Em uma arquitetura de AWS várias contas, uma VPC que aceita, inspeciona e roteia conexões de rede de fora de um aplicativo. A Arquitetura de Referência de AWS Segurança recomenda configurar sua conta de rede com entrada, saída e inspeção VPCs para proteger a interface bidirecional entre seu aplicativo e a Internet em geral.

migração incremental

Uma estratégia de substituição na qual você migra a aplicação em pequenas partes, em vez de realizar uma única substituição completa. Por exemplo, é possível mover inicialmente apenas alguns microsserviços ou usuários para o novo sistema. Depois de verificar se tudo está funcionando corretamente, mova os microsserviços ou usuários adicionais de forma incremental até poder descomissionar seu sistema herdado. Essa estratégia reduz os riscos associados a migrações de grande porte.

Indústria 4.0

Um termo que foi introduzido por Klaus Schwab em 2016 para se referir à modernização dos processos de manufatura por meio de avanços em conectividade, dados em tempo real, automação, analytics e IA/ML.

infraestrutura

Todos os recursos e ativos contidos no ambiente de uma aplicação.

Infraestrutura como código (IaC)

O processo de provisionamento e gerenciamento da infraestrutura de uma aplicação por meio de um conjunto de arquivos de configuração. A IaC foi projetada para ajudar você a centralizar o gerenciamento da infraestrutura, padronizar recursos e escalar rapidamente para que novos ambientes sejam reproduzíveis, confiáveis e consistentes.

Internet industrial das coisas (IIoT)

O uso de sensores e dispositivos conectados à Internet nos setores industriais, como manufatura, energia, automotivo, saúde, ciências biológicas e agricultura. Para obter mais informações, consulte Criando uma estratégia de transformação digital industrial da Internet das Coisas (IIoT).

VPC de inspeção

Em uma arquitetura de AWS várias contas, uma VPC centralizada que gerencia as inspeções do tráfego de rede entre VPCs (na mesma ou em diferentes Regiões da AWS) a Internet e as redes locais. A Arquitetura de Referência de AWS Segurança recomenda configurar sua conta de rede com entrada, saída e inspeção VPCs para proteger a interface bidirecional entre seu aplicativo e a Internet em geral.

Internet das coisas (IoT)

A rede de objetos físicos conectados com sensores ou processadores incorporados que se comunicam com outros dispositivos e sistemas pela Internet ou por uma rede de comunicação local. Para obter mais informações, consulte O que é IoT?

interpretabilidade

Uma característica de um modelo de machine learning que descreve o grau em que um ser humano pode entender como as previsões do modelo dependem de suas entradas. Para obter mais informações, consulte Interpretabilidade do modelo de aprendizado de máquina com AWS.

IoT

Veja Internet das Coisas.

Biblioteca de informações de TI (ITIL)

Um conjunto de práticas recomendadas para fornecer serviços de TI e alinhar esses serviços a requisitos de negócios. A ITIL fornece a base para o ITSM.

Gerenciamento de serviços de TI (ITSM)

Atividades associadas a design, implementação, gerenciamento e suporte de serviços de TI para uma organização. Para obter informações sobre a integração de operações em nuvem com ferramentas de ITSM, consulte o guia de integração de operações.

ITIL

Veja biblioteca de informações de TI.

ITSM

Veja gerenciamento de serviços de TI.

L

controle de acesso baseado em etiqueta (LBAC)

Uma implementação do controle de acesso obrigatório (MAC) em que os usuários e os dados em si recebem explicitamente um valor de etiqueta de segurança. A interseção entre a etiqueta de segurança do usuário e a etiqueta de segurança dos dados determina quais linhas e colunas podem ser vistas pelo usuário.

zona de pouso

Uma landing zone é um AWS ambiente bem arquitetado, com várias contas, escalável e seguro. Um ponto a partir do qual suas organizações podem iniciar e implantar rapidamente workloads e aplicações com confiança em seu ambiente de segurança e infraestrutura. Para obter mais informações sobre zonas de pouso, consulte Configurar um ambiente da AWS com várias contas seguro e escalável.

grande modelo de linguagem (LLM)

Um modelo de IA de aprendizado profundo pré-treinado em uma grande quantidade de dados. Um LLM pode realizar várias tarefas, como responder a perguntas, resumir documentos, traduzir texto para outros idiomas e completar frases. Para obter mais informações, consulte O que são LLMs.

migração de grande porte

Uma migração de 300 servidores ou mais.

LBAC

Veja controle de acesso baseado em rótulo.

privilégio mínimo

A prática recomendada de segurança de conceder as permissões mínimas necessárias para executar uma tarefa. Para obter mais informações, consulte Aplicar permissões de privilégios mínimos na documentação do IAM.

mover sem alterações (lift-and-shift)

Veja 7 Rs.

sistema little-endian

Um sistema que armazena o byte menos significativo antes. Veja também endianness.

LLM

Veja grande modelo de linguagem.

ambientes inferiores

Veja ambiente.

M

machine learning (ML)

Um tipo de inteligência artificial que usa algoritmos e técnicas para reconhecimento e aprendizado de padrões. O ML analisa e aprende com dados gravados, por exemplo, dados da Internet das Coisas (IoT), para gerar um modelo estatístico baseado em padrões. Para obter mais informações, consulte Machine learning.

ramificação principal

Veja ramificação.

Malware

Software projetado para comprometer a segurança ou a privacidade do computador. O malware pode interromper os sistemas do computador, vazar informações sensíveis ou obter acesso não autorizado. Exemplos de malware incluem vírus, worms, ransomware, cavalos de Troia, spyware e keyloggers.

Serviços gerenciados

Serviços da AWS para o qual AWS opera a camada de infraestrutura, o sistema operacional e as plataformas, e você acessa os endpoints para armazenar e recuperar dados. O Amazon Simple Storage Service (Amazon S3) e o Amazon DynamoDB são exemplos de serviços gerenciados. Eles também são conhecidos como serviços abstraídos.

sistema de execução de manufatura (MES)

Um sistema de software para rastrear, monitorar, documentar e controlar processos de produção que convertem matérias-primas em produtos acabados no chão de fábrica.

MAP

Veja Programa de Aceleração da Migração.

mecanismo

Um processo completo em que você cria uma ferramenta, impulsiona a adoção da ferramenta e, em seguida, inspeciona os resultados para fazer ajustes. Um mecanismo é um ciclo que se reforça e se aprimora à medida que opera. Para obter mais informações, consulte Construindo mecanismos no AWS Well-Architected Framework.

conta de membro

Todos, Contas da AWS exceto a conta de gerenciamento, que fazem parte de uma organização em AWS Organizations. Uma conta só pode ser membro de uma organização de cada vez.

MES

Veja sistema de execução de manufatura.

Transporte de Telemetria de Enfileiramento de Mensagens (MQTT)

Um protocolo de comunicação leve machine-to-machine (M2M), baseado no padrão de publicação/assinatura, para dispositivos de IoT com recursos limitados.

microsserviço

Um serviço pequeno e independente que se comunica de forma bem definida APIs e normalmente é de propriedade de equipes pequenas e independentes. Por exemplo, um sistema de seguradora pode incluir microsserviços que mapeiam as capacidades comerciais, como vendas ou marketing, ou subdomínios, como compras, reclamações ou análises. Os benefícios dos microsserviços incluem agilidade, escalabilidade flexível, fácil implantação, código reutilizável e resiliência. Para obter mais informações, consulte Integração de microsserviços usando serviços sem AWS servidor.

arquitetura de microsserviços

Uma abordagem à criação de aplicações com componentes independentes que executam cada processo de aplicação como um microsserviço. Esses microsserviços se comunicam por meio

de uma interface bem definida usando leveza. APIs Cada microsserviço nessa arquitetura pode ser atualizado, implantado e escalado para atender à demanda por funções específicas de uma aplicação. Para obter mais informações, consulte Implementação de microsserviços em. AWS

Programa de Aceleração da Migração (MAP)

Um AWS programa que fornece suporte de consultoria, treinamento e serviços para ajudar as organizações a criar uma base operacional sólida para migrar para a nuvem e ajudar a compensar o custo inicial das migrações. O MAP inclui uma metodologia de migração para executar migrações legadas de forma metódica e um conjunto de ferramentas para automatizar e acelerar cenários comuns de migração.

migração em escala

O processo de mover a maior parte do portfólio de aplicações para a nuvem em ondas, com mais aplicações sendo movidas em um ritmo mais rápido a cada onda. Essa fase usa as práticas recomendadas e lições aprendidas nas fases anteriores para implementar uma fábrica de migração de equipes, ferramentas e processos para agilizar a migração de workloads por meio de automação e entrega ágeis. Esta é a terceira fase da estratégia de migração para a AWS.

fábrica de migração

Equipes multifuncionais que simplificam a migração de workloads por meio de abordagens automatizadas e ágeis. As equipes da fábrica de migração geralmente incluem operações, analistas e proprietários de negócios, engenheiros de migração, desenvolvedores e DevOps profissionais que trabalham em sprints. Entre 20 e 50% de um portfólio de aplicações corporativas consiste em padrões repetidos que podem ser otimizados por meio de uma abordagem de fábrica. Para obter mais informações, consulte discussão sobre fábricas de migração e o guia do Cloud Migration Factory neste conjunto de conteúdo.

metadados de migração

As informações sobre a aplicação e o servidor necessárias para concluir a migração. Cada padrão de migração exige um conjunto de metadados de migração diferente. Exemplos de metadados de migração incluem a sub-rede, o grupo de segurança e AWS a conta de destino.

padrão de migração

Uma tarefa de migração repetível que detalha a estratégia de migração, o destino da migração e a aplicação ou o serviço de migração usado. Exemplo: rehospede a migração para o Amazon EC2 AWS com o Application Migration Service.

Avaliação de Portfólio para Migração (MPA)

Uma ferramenta on-line que fornece informações para validar o caso de negócios para migrar para a Nuvem AWS. O MPA fornece avaliação detalhada do portfólio (dimensionamento correto do servidor, preços, comparações de TCO, análise de custos de migração), bem como planejamento de migração (análise e coleta de dados de aplicações, agrupamento de aplicações, priorização de migração e planejamento de ondas). A ferramenta MPA (requer login) está disponível gratuitamente para todos os AWS consultores e consultores parceiros da APN.

Avaliação de Preparação para Migração (MRA)

O processo de obter insights sobre o status de prontidão de uma organização para a nuvem, identificar pontos fortes e fracos e criar um plano de ação para fechar as lacunas identificadas, usando o CAF. AWS Para mais informações, consulte o guia de preparação para migração. A MRA é a primeira fase da estratégia de migração para a AWS.

estratégia de migração

A abordagem usada para migrar uma workload para a Nuvem AWS. Para obter mais informações, veja a entrada 7 Rs neste glossário e consulte Mobilize sua organização para acelerar migrações em grande escala.

ML

Veja machine learning.

modernização

Transformar uma aplicação desatualizada (herdada ou monolítica) e sua infraestrutura em um sistema ágil, elástico e altamente disponível na nuvem para reduzir custos, ganhar eficiência e aproveitar as inovações. Para obter mais informações, consulte Strategy for modernizing applications in the Nuvem AWS.

avaliação de preparação para modernização

Uma avaliação que ajuda a determinar a preparação para modernização das aplicações de uma organização. Ela identifica benefícios, riscos e dependências e determina o quão bem a organização pode acomodar o estado futuro dessas aplicações. O resultado da avaliação é um esquema da arquitetura de destino, um roteiro que detalha as fases de desenvolvimento e os marcos do processo de modernização e um plano de ação para abordar as lacunas identificadas. Para obter mais informações, consulte Evaluating modernization readiness for applications in the Nuvem AWS.

aplicações monolíticas (monólitos)

Aplicações que são executadas como um único serviço com processos fortemente acoplados. As aplicações monolíticas apresentam várias desvantagens. Se um recurso da aplicação apresentar um aumento na demanda, toda a arquitetura deverá ser escalada. Adicionar ou melhorar os recursos de uma aplicação monolítica também se torna mais complexo quando a base de código cresce. Para resolver esses problemas, é possível criar uma arquitetura de microsserviços. Para obter mais informações, consulte Decompor monólitos em microsserviços.

MPA

Veja Avaliação do Portfólio para Migração.

MQTT

Veja Transporte de Telemetria de Enfileiramento de Mensagens.

classificação multiclasse

Um processo que ajuda a gerar previsões para várias classes (prevendo um ou mais de dois resultados). Por exemplo, um modelo de ML pode perguntar “Este produto é um livro, um carro ou um telefone?” ou “Qual categoria de produtos é mais interessante para este cliente?”

infraestrutura mutável

Um modelo que atualiza e modifica a infraestrutura existente para workloads de produção. Para melhorar a consistência, confiabilidade e previsibilidade, o AWS Well-Architected Framework recomenda o uso de infraestrutura imutável como uma prática recomendada.

O

OAC

Veja controle de acesso de origem.

OAI

Veja identidade de acesso de origem.

OCM

Veja gerenciamento de alterações organizacionais.

migração offline

Um método de migração no qual a workload de origem é desativada durante o processo de migração. Esse método envolve tempo de inatividade prolongado e geralmente é usado para workloads pequenas e não críticas.

OI

Veja integração de operações.

Ola

Veja acordo de nível operacional.

migração online

Um método de migração no qual a workload de origem é copiada para o sistema de destino sem ser colocada offline. As aplicações conectadas à workload podem continuar funcionando durante a migração. Esse método envolve um tempo de inatividade nulo ou mínimo e normalmente é usado para workloads essenciais para a produção.

OPC-UA

Veja Open Process Communications - Unified Architecture.

Open Process Communications - Unified Architecture (OPC-UA)

Um protocolo de comunicação machine-to-machine (M2M) para automação industrial. O OPC-UA fornece um padrão de interoperabilidade com esquemas de criptografia, autenticação e autorização de dados.

acordo de nível operacional (OLA)

Um acordo que esclarece o que os grupos funcionais de TI prometem oferecer uns aos outros para apoiar um acordo de serviço (SLA).

análise de prontidão operacional (ORR)

Uma lista de verificação de perguntas e práticas recomendadas associadas que ajudam você a entender, avaliar, prevenir ou reduzir o escopo de incidentes e possíveis falhas. Para obter mais informações, consulte Operational Readiness Reviews (ORR) no AWS Well-Architected Framework.

tecnologia operacional (TO)

Sistemas de hardware e software que trabalham com o ambiente físico para controlar operações, equipamentos e infraestrutura industriais. Na manufatura, a integração dos sistemas de

tecnologia da informação (TI) e tecnologia operacional (TO) é o foco principal das transformações da Indústria 4.0.

integração de operações (OI)

O processo de modernização das operações na nuvem, que envolve planejamento de preparação, automação e integração. Para obter mais informações, consulte o guia de integração de operações.

trilha organizacional

Uma trilha criada por ela AWS CloudTrail registra todos os eventos de todos Contas da AWS em uma organização em AWS Organizations. Essa trilha é criada em cada Conta da AWS que faz parte da organização e monitora a atividade em cada conta. Para obter mais informações, consulte Criação de uma trilha para uma organização na CloudTrail documentação.

gerenciamento de alterações organizacionais (OCM)

Uma estrutura para gerenciar grandes transformações de negócios disruptivas de uma perspectiva de pessoas, cultura e liderança. O OCM ajuda as organizações a se prepararem e fazerem a transição para novos sistemas e estratégias, acelerando a adoção de alterações, abordando questões de transição e promovendo mudanças culturais e organizacionais. Na estratégia de AWS migração, essa estrutura é chamada de aceleração de pessoas, devido à velocidade de mudança exigida nos projetos de adoção da nuvem. Para obter mais informações, consulte o guia do OCM.

controle de acesso de origem (OAC)

Em CloudFront, uma opção aprimorada para restringir o acesso para proteger seu conteúdo do Amazon Simple Storage Service (Amazon S3). O OAC oferece suporte a todos os buckets S3 Regiões da AWS, criptografia do lado do servidor com AWS KMS (SSE-KMS) e solicitações dinâmicas ao bucket S3. `PUT` `DELETE`

Identidade do acesso de origem (OAI)

Em CloudFront, uma opção para restringir o acesso para proteger seu conteúdo do Amazon S3. Quando você usa o OAI, CloudFront cria um principal com o qual o Amazon S3 pode se autenticar. Os diretores autenticados podem acessar o conteúdo em um bucket do S3 somente por meio de uma distribuição específica. CloudFront Veja também OAC, que fornece um controle de acesso mais granular e aprimorado.

ORR

Veja análise de prontidão operacional.

OT

Veja tecnologia operacional.

VPC de saída (egresso)

Em uma arquitetura de AWS várias contas, uma VPC que gerencia conexões de rede que são iniciadas de dentro de um aplicativo. A Arquitetura de Referência de AWS Segurança recomenda configurar sua conta de rede com entrada, saída e inspeção VPCs para proteger a interface bidirecional entre seu aplicativo e a Internet em geral.

P

limite de permissões

Uma política de gerenciamento do IAM anexada a entidades principais do IAM para definir as permissões máximas que o usuário ou perfil podem ter. Para obter mais informações, consulte Limites de permissões na documentação do IAM.

Informações de identificação pessoal (PII)

Informações que, quando visualizadas diretamente ou combinadas com outros dados relacionados, podem ser usadas para inferir razoavelmente a identidade de um indivíduo. Exemplos de PII incluem nomes, endereços e informações de contato.

PII

Veja informações de identificação pessoal.

manual

Um conjunto de etapas predefinidas que capturam o trabalho associado às migrações, como a entrega das principais funções operacionais na nuvem. Um manual pode assumir a forma de scripts, runbooks automatizados ou um resumo dos processos ou etapas necessários para operar seu ambiente modernizado.

PLC

Veja controlador lógico programável.

PLM

Veja gerenciamento do ciclo de vida do produto.

política

Um objeto que pode definir permissões (veja política baseada em identidade), especificar condições de acesso (veja polítca baseada em recurso) ou definir as permissões máximas para todas as contas em uma organização no AWS Organizations (veja política de controle de serviços).

persistência poliglota

Escolher de forma independente a tecnologia de armazenamento de dados de um microsserviço com base em padrões de acesso a dados e outros requisitos. Se seus microsserviços tiverem a mesma tecnologia de armazenamento de dados, eles poderão enfrentar desafios de implementação ou apresentar baixa performance. Os microsserviços serão implementados com mais facilidade e alcançarão performance e escalabilidade melhores se usarem o armazenamento de dados mais bem adaptado às suas necessidades.

avaliação do portfólio

Um processo de descobrir, analisar e priorizar o portfólio de aplicações para planejar a migração. Para obter mais informações, consulte Avaliar a preparação para a migração.

predicado

Uma condição de consulta que retorna `true` ou `false`, normalmente localizada em uma cláusula `WHERE`.

pushdown de predicados

Uma técnica de otimização de consultas de banco de dados que filtra os dados na consulta antes da transferência. Isso reduz a quantidade de dados que devem ser recuperados e processados do banco de dados relacional e melhora a performance das consultas.

controle preventivo

Um controle de segurança projetado para evitar que um evento ocorra. Esses controles são a primeira linha de defesa para ajudar a evitar acesso não autorizado ou alterações indesejadas em sua rede. Para obter mais informações, consulte Controles preventivos em Como implementar controles de segurança na AWS.

principal (entidade principal)

Uma entidade AWS que pode realizar ações e acessar recursos. Essa entidade geralmente é um usuário raiz para um Conta da AWS, uma função do IAM ou um usuário. Para obter mais

informações, consulte Entidade principal em Termos e conceitos de perfis na documentação do IAM.

Privacidade por design

Uma abordagem em engenharia de sistemas que leva em consideração a privacidade em todo o processo de desenvolvimento.

zonas hospedadas privadas

Um contêiner que contém informações sobre como você deseja que o Amazon Route 53 responda às consultas de DNS para um domínio e seus subdomínios em um ou mais. VPCs Para obter mais informações, consulte Como trabalhar com zonas hospedadas privadas na documentação do Route 53.

controle proativo

Um controle de segurança desenvolvido para evitar a implantação de recursos não conformes. Esses controles verificam os recursos antes de serem provisionados. Se o recurso não estiver em conformidade com o controle, ele não será provisionado. Para obter mais informações, consulte o guia de referência de controles na AWS Control Tower documentação e consulte Controles proativos em Implementação de controles de segurança em AWS.

gerenciamento do ciclo de vida do produto (PLM)

O gerenciamento de dados e processos de um produto em todo o seu ciclo de vida, desde a concepção, o desenvolvimento e o lançamento, passando pelo crescimento e maturidade, até o declínio e a remoção.

ambiente de produção

Veja ambiente.

controlador lógico programável (PLC)

Na manufatura, um computador altamente confiável e adaptável que monitora as máquinas e automatiza os processos de fabricação.

encadeamento de prompts

Uso da saída de um prompt do LLM como entrada para o próximo prompt para gerar respostas melhores. Essa técnica é usada para dividir uma tarefa complexa em subtarefas, ou para refinar ou expandir iterativamente uma resposta preliminar. Isso ajuda a melhorar a precisão e a relevância das respostas de um modelo e permite resultados mais granulares e personalizados.

pseudonimização

O processo de substituir identificadores pessoais em um conjunto de dados por valores de espaço reservado. A pseudonimização pode ajudar a proteger a privacidade pessoal. Os dados pseudonimizados ainda são considerados dados pessoais.

publish/subscribe (pub/sub)

Um padrão que permite comunicações assíncronas entre microsserviços para melhorar a escalabilidade e a capacidade de resposta. Por exemplo, em um MES baseado em microsserviços, um microsserviço pode publicar mensagens de eventos em um canal em que outros microsserviços possam assinar. O sistema pode adicionar novos microsserviços sem alterar o serviço de publicação.

Q

plano de consulta

Uma série de etapas, como instruções, usadas para acessar os dados em um sistema de banco de dados relacional SQL.

regressão de planos de consultas

Quando um otimizador de serviço de banco de dados escolhe um plano menos adequado do que escolhia antes de uma determinada alteração no ambiente de banco de dados ocorrer. Isso pode ser causado por alterações em estatísticas, restrições, configurações do ambiente, associações de parâmetros de consulta e atualizações do mecanismo de banco de dados.

R

Matriz RACI

Veja responsável, aprovador, consultado, informado (RACI).

RAG

Veja geração aumentada via recuperação.

ransomware

Um software mal-intencionado desenvolvido para bloquear o acesso a um sistema ou dados de computador até que um pagamento seja feito.

Matriz RASCI

Veja responsável, aprovador, consultado, informado (RACI).

RCAC

Veja controle de acesso por linha e coluna.

réplica de leitura

Uma cópia de um banco de dados usada somente para leitura. É possível encaminhar consultas para a réplica de leitura e reduzir a carga no banco de dados principal.

Redefinir arquitetura

Veja 7 Rs.

objetivo de ponto de recuperação (RPO).

O máximo período de tempo aceitável desde o último ponto de recuperação de dados. Isso determina o que é considerado uma perda aceitável de dados entre o último ponto de recuperação e a interrupção do serviço.

objetivo de tempo de recuperação (RTO)

O máximo atraso aceitável entre a interrupção e a restauração do serviço.

refatorar

Veja 7 Rs.

Região

Uma coleção de AWS recursos em uma área geográfica. Cada um Região da AWS é isolado e independente dos outros para fornecer tolerância a falhas, estabilidade e resiliência. Para obter informações, consulte Specify which Regiões da AWS your account can use.

regressão

Uma técnica de ML que prevê um valor numérico. Por exemplo, para resolver o problema de “Por qual preço esta casa será vendida?” um modelo de ML pode usar um modelo de regressão linear para prever o preço de venda de uma casa com base em fatos conhecidos sobre a casa (por exemplo, a metragem quadrada).

redefinir a hospedagem

Veja 7 Rs.

versão

Em um processo de implantação, o ato de promover mudanças em um ambiente de produção.

realocar

Veja 7 Rs.

redefinir a plataforma

Veja 7 Rs.

recomprar

Veja 7 Rs.

resiliência

A capacidade de uma aplicação de resistir ou se recuperar de interrupções. Alta disponibilidade e recuperação de desastres são considerações comuns ao planejar a resiliência na Nuvem AWS. Para obter mais informações, consulte Nuvem AWS Resilience.

política baseada em recurso

Uma política associada a um recurso, como um bucket do Amazon S3, um endpoint ou uma chave de criptografia. Esse tipo de política especifica quais entidades principais têm acesso permitido, ações válidas e quaisquer outras condições que devem ser atendidas.

matriz responsável, accountable, consultada, informada (RACI)

Uma matriz que define as funções e responsabilidades de todas as partes envolvidas nas atividades de migração e nas operações de nuvem. O nome da matriz é derivado dos tipos de responsabilidade definidos na matriz: responsável (R), responsabilizável (A), consultado (C) e informado (I). O tipo de suporte (S) é opcional. Se você incluir suporte, a matriz será chamada de matriz RASCI e, se excluir, será chamada de matriz RACI.

controle responsivo

Um controle de segurança desenvolvido para conduzir a remediação de eventos adversos ou desvios em relação à linha de base de segurança. Para obter mais informações, consulte Controles responsivos em Como implementar controles de segurança na AWS.

reter

Veja 7 Rs.

Retirada

Veja 7 Rs.

Geração Aumentada de Recuperação (RAG)

Uma tecnologia de IA generativa em que um LLM faz referência a uma fonte de dados autorizada que está fora de suas fontes de dados de treinamento antes de gerar uma resposta. Por exemplo, um modelo RAG pode realizar uma pesquisa semântica na base de conhecimento ou nos dados personalizados de uma organização. Para obter mais informações, consulte O que é RAG (geração aumentada via recuperação)?.

alternância

O processo de atualizar periodicamente um segredo para dificultar o acesso de um invasor às credenciais.

controle de acesso por linha e coluna (RCAC)

O uso de expressões SQL básicas e flexíveis que tenham regras de acesso definidas. O RCAC consiste em permissões de linha e máscaras de coluna.

RPO

Veja objetivo de ponto de recuperação.

RTO

Veja objetivo de tempo de recuperação.

runbook

Um conjunto de procedimentos manuais ou automatizados necessários para realizar uma tarefa específica. Eles são normalmente criados para agilizar operações ou procedimentos repetitivos com altas taxas de erro.

S

SAML 2.0

Um padrão aberto que muitos provedores de identidade (IdPs) usam. Esse recurso permite o login único federado (SSO), para que os usuários possam fazer login Console de gerenciamento da AWS ou chamar as operações da AWS API sem que você precise criar um usuário no IAM

para todos em sua organização. Para obter mais informações sobre a federação baseada em SAML 2.0, consulte Sobre a federação baseada em SAML 2.0 na documentação do IAM.

SCADA

Veja controle de supervisão e aquisição de dados.

SCP

Veja política de controle de serviço.

secret

Em AWS Secrets Manager, informações confidenciais ou restritas, como uma senha ou credenciais de usuário, que você armazena de forma criptografada. Consiste no valor secreto e em seus metadados. O valor secreto pode ser binário, uma única string ou várias strings. Para obter mais informações, consulte What's in a Secrets Manager secret? na documentação do Secrets Manager.

segurança desde a concepção

Uma abordagem em engenharia de sistemas que leva em consideração a segurança em todo o processo de desenvolvimento.

controle de segurança

Uma barreira de proteção técnica ou administrativa que impede, detecta ou reduz a capacidade de uma ameaça explorar uma vulnerabilidade de segurança. Existem quatro tipos primários de controles de segurança: preventivos, detectivos, responsivos e proativos.

hardening da segurança

O processo de reduzir a superfície de ataque para torná-la mais resistente a ataques. Isso pode incluir ações como remover recursos que não são mais necessários, implementar a prática recomendada de segurança de conceder privilégios mínimos ou desativar recursos desnecessários em arquivos de configuração.

sistema de gerenciamento de eventos e informações de segurança (SIEM)

Ferramentas e serviços que combinam sistemas de gerenciamento de informações de segurança (SIM) e gerenciamento de eventos de segurança (SEM). Um sistema SIEM coleta, monitora e analisa dados de servidores, redes, dispositivos e outras fontes para detectar ameaças e violações de segurança e gerar alertas.

automação de resposta de segurança

Uma ação predefinida e programada projetada para responder ou remediar automaticamente um evento de segurança. Essas automações servem como controles de segurança responsivos ou detectivos que ajudam você a implementar as melhores práticas AWS de segurança. Exemplos de ações de resposta automatizada incluem a modificação de um grupo de segurança da VPC, a aplicação de patches em uma instância do Amazon EC2 ou a alternância de credenciais.

Criptografia do lado do servidor

Criptografia dos dados em seu destino, por AWS service (Serviço da AWS) quem os recebe.

política de controle de serviços (SCP)

Uma política que fornece controle centralizado sobre as permissões de todas as contas em uma organização em AWS Organizations. SCPs defina barreiras ou estabeleça limites nas ações que um administrador pode delegar a usuários ou funções. Você pode usar SCPs como listas de permissão ou listas de negação para especificar quais serviços ou ações são permitidos ou proibidos. Para obter mais informações, consulte Políticas de controle de serviço na AWS Organizations documentação.

service endpoint (endpoint de serviço)

O URL do ponto de entrada para um AWS service (Serviço da AWS). Você pode usar o endpoint para se conectar programaticamente ao serviço de destino. Para obter mais informações, consulte Endpoints do AWS service (Serviço da AWS) na Referência geral da AWS.

acordo de serviço (SLA)

Um acordo que esclarece o que uma equipe de TI promete fornecer aos clientes, como tempo de atividade e performance do serviço.

indicador de nível de serviço (SLI)

Uma avaliação de um aspecto de performance de um serviço, como taxa de erro, disponibilidade ou throughput.

objetivo de nível de serviço (SLO)

Uma métrica alvo que representa a integridade de um serviço, conforme avaliado por um indicador de nível de serviço.

modelo de responsabilidade compartilhada

Um modelo que descreve a responsabilidade com a qual você compartilha AWS pela segurança e conformidade na nuvem. AWS é responsável pela segurança da nuvem, enquanto você é responsável pela segurança na nuvem. Para obter mais informações, consulte o Modelo de responsabilidade compartilhada.

SIEM

Veja sistema de gerenciamento de eventos e informações de segurança.

ponto único de falha (SPOF)

Uma falha em um único componente crítico de uma aplicação que pode interromper o sistema.

SLA

Veja cacordo de serviço.

SLI

Veja indicador de nível de serviço.

SLO

Veja objetivo de nível de serviço.

split-and-seed modelo

Um padrão para escalar e acelerar projetos de modernização. À medida que novos recursos e lançamentos de produtos são definidos, a equipe principal se divide para criar novas equipes de produtos. Isso ajuda a escalar os recursos e os serviços da sua organização, melhora a produtividade do desenvolvedor e possibilita inovações rápidas. Para obter mais informações, consulte Phased approach to modernizing applications in the Nuvem AWS.

SPOF

Veja ponto único de falha.

esquema em estrela

Uma estrutura organizacional de banco de dados que usa uma grande tabela de fatos para armazenar dados transacionais ou medidos e usa uma ou mais tabelas dimensionais menores para armazenar atributos de dados. Essa estrutura foi projetada para ser usada em um data warehouse ou para fins de inteligência comercial.

padrão strangler fig

Uma abordagem à modernização de sistemas monolíticos que consiste em reescrever e substituir incrementalmente a funcionalidade do sistema até que o sistema herdado possa ser desativado. Esse padrão usa a analogia de uma videira que cresce e se torna uma árvore estabelecida e, eventualmente, supera e substitui sua hospedeira. O padrão foi apresentado por Martin Fowler como forma de gerenciar riscos ao reescrever sistemas monolíticos. Para ver um exemplo de como aplicar esse padrão, consulte Modernizar incrementalmente os serviços Web herdados do Microsoft ASP.NET (ASMX) usando contêineres e o Amazon API Gateway.

sub-rede

Um intervalo de endereços IP na VPC. Cada sub-rede fica alocada em uma única zona de disponibilidade.

controle supervisório e aquisição de dados (SCADA)

Na manufatura, um sistema que usa hardware e software para monitorar ativos físicos e operações de produção.

symmetric encryption (criptografia simétrica)

Um algoritmo de criptografia que usa a mesma chave para criptografar e descriptografar dados.

testes sintéticos

Testar um sistema de forma que simule as interações do usuário para detectar possíveis problemas ou monitorar a performance. Você pode usar o Amazon CloudWatch Synthetics para criar esses testes.

prompt do sistema

Uma técnica para fornecer contexto, instruções ou orientações a um LLM a fim de direcionar seu comportamento. Os prompts do sistema ajudam a definir o contexto e a estabelecer regras para interações com os usuários.

T

tags

Pares de valores-chave que atuam como metadados para organizar seus recursos. AWS As tags podem ajudar você a gerenciar, identificar, organizar, pesquisar e filtrar recursos da . Para obter mais informações, consulte Marcar seus recursos do AWS.

variável-alvo

O valor que você está tentando prever no ML supervisionado. Ela também é conhecida como variável de resultado. Por exemplo, em uma configuração de fabricação, a variável-alvo pode ser um defeito do produto.

lista de tarefas

Uma ferramenta usada para monitorar o progresso por meio de um runbook. Uma lista de tarefas contém uma visão geral do runbook e uma lista de tarefas gerais a serem concluídas. Para cada tarefa geral, ela inclui o tempo estimado necessário, o proprietário e o progresso.

ambiente de teste

Veja ambiente.

treinamento

O processo de fornecer dados para que seu modelo de ML aprenda. Os dados de treinamento devem conter a resposta correta. O algoritmo de aprendizado descobre padrões nos dados de treinamento que mapeiam os atributos dos dados de entrada no destino (a resposta que você deseja prever). Ele gera um modelo de ML que captura esses padrões. Você pode usar o modelo de ML para obter previsões de novos dados cujo destino você não conhece.

gateway de trânsito

Um hub de trânsito de rede que você pode usar para interconectar sua rede com VPCs a rede local. Para obter mais informações, consulte O que é um gateway de trânsito na AWS Transit Gateway documentação.

fluxo de trabalho baseado em troncos

Uma abordagem na qual os desenvolvedores criam e testam recursos localmente em uma ramificação de recursos e, em seguida, mesclam essas alterações na ramificação principal. A ramificação principal é então criada para os ambientes de desenvolvimento, pré-produção e produção, sequencialmente.

Acesso confiável

Conceder permissões a um serviço que você especifica para realizar tarefas em sua organização AWS Organizations e em suas contas em seu nome. O serviço confiável cria um perfil vinculado ao serviço em cada conta, quando esse perfil é necessário, para realizar tarefas de

gerenciamento para você. Para obter mais informações, consulte Usando AWS Organizations com outros AWS serviços na AWS Organizations documentação.

tuning (ajustar)

Alterar aspectos do processo de treinamento para melhorar a precisão do modelo de ML. Por exemplo, você pode treinar o modelo de ML gerando um conjunto de rótulos, adicionando rótulos e repetindo essas etapas várias vezes em configurações diferentes para otimizar o modelo.

equipe de duas pizzas

Uma pequena DevOps equipe que você pode alimentar com duas pizzas. Uma equipe de duas pizzas garante a melhor oportunidade possível de colaboração no desenvolvimento de software.

U

incerteza

Um conceito que se refere a informações imprecisas, incompletas ou desconhecidas que podem minar a confiabilidade dos modelos preditivos de ML. Há dois tipos de incertezas: a incerteza epistêmica é causada por dados limitados e incompletos, enquanto a incerteza aleatória é causada pelo ruído e pela aleatoriedade inerentes aos dados. Para obter mais informações, consulte o guia Como quantificar a incerteza em sistemas de aprendizado profundo.

tarefas indiferenciadas

Também conhecido como trabalho pesado, trabalho necessário para criar e operar um aplicativo, mas que não fornece valor direto ao usuário final nem oferece vantagem competitiva. Exemplos de tarefas indiferenciadas incluem aquisição, manutenção e planejamento de capacidade.

ambientes superiores

Veja ambiente.

V

aspiração

Uma operação de manutenção de banco de dados que envolve limpeza após atualizações incrementais para recuperar armazenamento e melhorar a performance.

controle de versões

Processos e ferramentas que rastreiam mudanças, como alterações no código-fonte em um repositório.

emparelhamento da VPC

Uma conexão entre duas VPCs que permite rotear o tráfego usando endereços IP privados. Para ter mais informações, consulte O que é emparelhamento de VPC? na documentação da Amazon VPC.

Vulnerabilidade

Uma falha de software ou hardware que compromete a segurança do sistema.

W

cache quente

Um cache de buffer que contém dados atuais e relevantes que são acessados com frequência. A instância do banco de dados pode ler do cache do buffer, o que é mais rápido do que ler da memória principal ou do disco.

dados mornos

Dados acessados raramente. Ao consultar esse tipo de dados, consultas moderadamente lentas geralmente são aceitáveis.

função de janela

Uma função SQL que executa um cálculo em um grupo de linhas que se relacionam de alguma forma com o registro atual. As funções de janela são úteis para processar tarefas, como calcular uma média móvel ou acessar o valor das linhas com base na posição relativa da linha atual.

workload

Uma coleção de códigos e recursos que geram valor empresarial, como uma aplicação voltada para o cliente ou um processo de backend.

workstreams

Grupos funcionais em um projeto de migração que são responsáveis por um conjunto específico de tarefas. Cada workstream é independente, mas oferece suporte aos outros workstreams do

projeto. Por exemplo, o workstream de portfólio é responsável por priorizar aplicações, planejar ondas e coletar metadados de migração. O workstream de portfólio entrega esses ativos ao workstream de migração, que então migra os servidores e as aplicações.

WORM

Veja gravação única e várias leituras.

WQF

Veja AWS Workload Qualification Framework.

gravação única e várias leituras (WORM)

Um modelo de armazenamento que grava dados uma única vez e evita que os dados sejam excluídos ou modificados. Os usuários autorizados podem ler os dados quantas vezes forem necessárias, mas não podem alterá-los. Essa infraestrutura de armazenamento de dados é considerada imutável.

Z

exploração de dia zero

Um ataque, normalmente malware, que tira proveito de uma vulnerabilidade zero-day.

vulnerabilidade de dia zero

Uma falha ou vulnerabilidade não mitigada em um sistema de produção. Os agentes de ameaças podem usar esse tipo de vulnerabilidade para atacar o sistema. Os desenvolvedores frequentemente ficam cientes da vulnerabilidade como resultado do ataque.

prompt zero shot

Fornecer a um LLM instruções para realizar uma tarefa, mas sem exemplos (shots) que possam ajudar a orientá-lo. O LLM deve usar seu conhecimento pré-treinado para lidar com a tarefa. A eficácia dos prompts zero-shot depende da complexidade da tarefa e da qualidade do prompt. Veja também prompts few-shot.

aplicação zumbi

Uma aplicação que tem um uso médio de CPU e memória inferior a 5%. Em um projeto de migração, é comum retirar essas aplicações.

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