Mejores prácticas para ajustar el rendimiento de AWS Glue los trabajos de Apache Spark

: Mejores prácticas para ajustar el rendimiento de AWS Glue los trabajos de Apache Spark

Table of Contents

Introducción 1
Temas clave de 2
Arquitectura 2
Conjunto de datos distribuido resiliente 3
Evaluación perezosa 5
Terminología de las aplicaciones de Spark 6
Paralelismo 7
Optimizador Catalyst 8
Investiga los problemas de rendimiento 11
Identifica los cuellos de botella mediante la interfaz de usuario de Spark 11
Estrategias para ajustar el rendimiento 13
Estrategia básica para ajustar el rendimiento 13
Perfeccionar las prácticas para mejorar el desempeño laboral de Spark 14
Amplíe la capacidad del clúster 15
CloudWatch métricas 15
Interfaz de usuario de Spark 16
Uso de la versión más reciente 17
Reduzca la cantidad de datos escaneados 18
CloudWatch métricas 18
Interfaz de usuario de Spark 19
Paraleliza las tareas 28
CloudWatch métricas 28
Interfaz de usuario de Spark 29
Optimiza los barajados 35
CloudWatch métricas 36
Interfaz de usuario de Spark 36
Minimice los gastos de planificación 45
CloudWatch métricas 45
Interfaz de usuario de Spark 46
Optimice las funciones definidas por el usuario 47
Python estándar UDF 49
Vectorizado UDF 49
Spark SQL 50
Uso de pandas para macrodatos 51

Recursos 52
Historial de documentos 53
Glosario 54
..... 54
A 55
B 58
C 60
D 63
E 68
F 70
G 71
H 72
I 73
L 76
M 77
O 81
P 84
Q 87
R 87
S 90
T 94
U 95
V 96
W 96
Z 98
..... xcix

Mejores prácticas para ajustar el rendimiento de AWS Glue los trabajos de Apache Spark

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

Diciembre de 2023 (historial de documentos)

AWS Glue ofrece diferentes opciones para ajustar el rendimiento. Esta guía define los temas clave para el ajuste AWS Glue de Apache Spark. Luego, proporciona una estrategia básica que puede seguir al ajustarlos AWS Glue para los trabajos de Apache Spark. Utilice esta guía para aprender a identificar los problemas de rendimiento mediante la interpretación de las métricas disponibles en AWS Glue. Luego, incorpore estrategias para abordar estos problemas, maximizando el rendimiento y minimizando los costos.

Esta guía cubre las siguientes prácticas de ajuste:

- Amplíe la capacidad del clúster
- Utilice la AWS Glue versión más reciente
- Reduzca la cantidad de datos escaneados
- Paraleliza las tareas
- Minimice la sobrecarga de planificación
- Optimice los barajamientos
- Optimice las funciones definidas por el usuario

Temas clave de Apache Spark

En esta sección se explican los conceptos básicos de Apache Spark y los temas clave AWS Glue para ajustar el rendimiento de Apache Spark. Es importante entender estos conceptos y temas antes de analizar las estrategias de ajuste del mundo real.

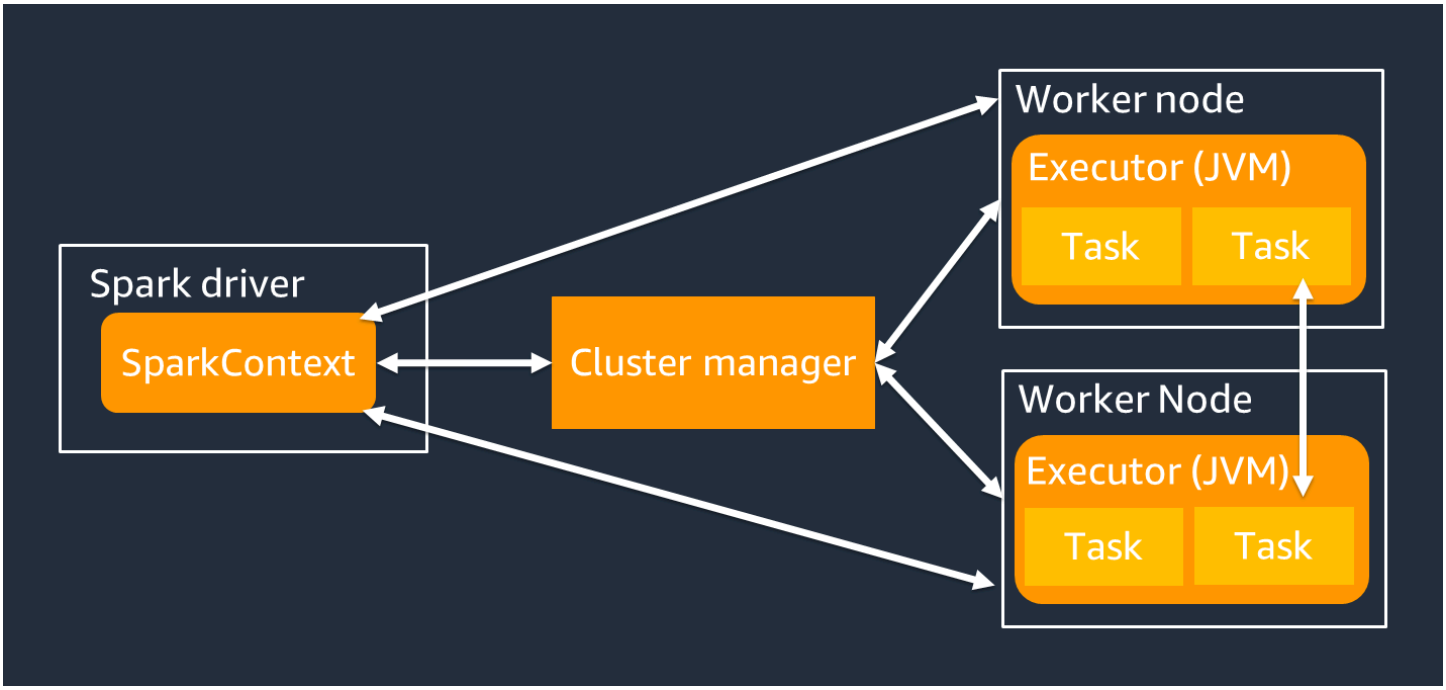
Arquitectura

El controlador de Spark es el principal responsable de dividir tu aplicación Spark en tareas que puedan realizar trabajadores individuales. El conductor de Spark tiene las siguientes responsabilidades:

- Se ejecuta `main()` en tu código
- Generando planes de ejecución
- Aprovisionamiento de los ejecutores de Spark junto con el administrador de clústeres, que gestiona los recursos del clúster
- Programar tareas y solicitar tareas para los ejecutores de Spark
- Gestionar el progreso y la recuperación de las tareas

Utilizas un `SparkContext` objeto para interactuar con el controlador de Spark durante tu trabajo.

Un ejecutor de Spark es un elemento que se encarga de almacenar datos y ejecutar tareas que se transfieren desde el controlador de Spark. El número de ejecutores de Spark aumentará y disminuirá con el tamaño del clúster.

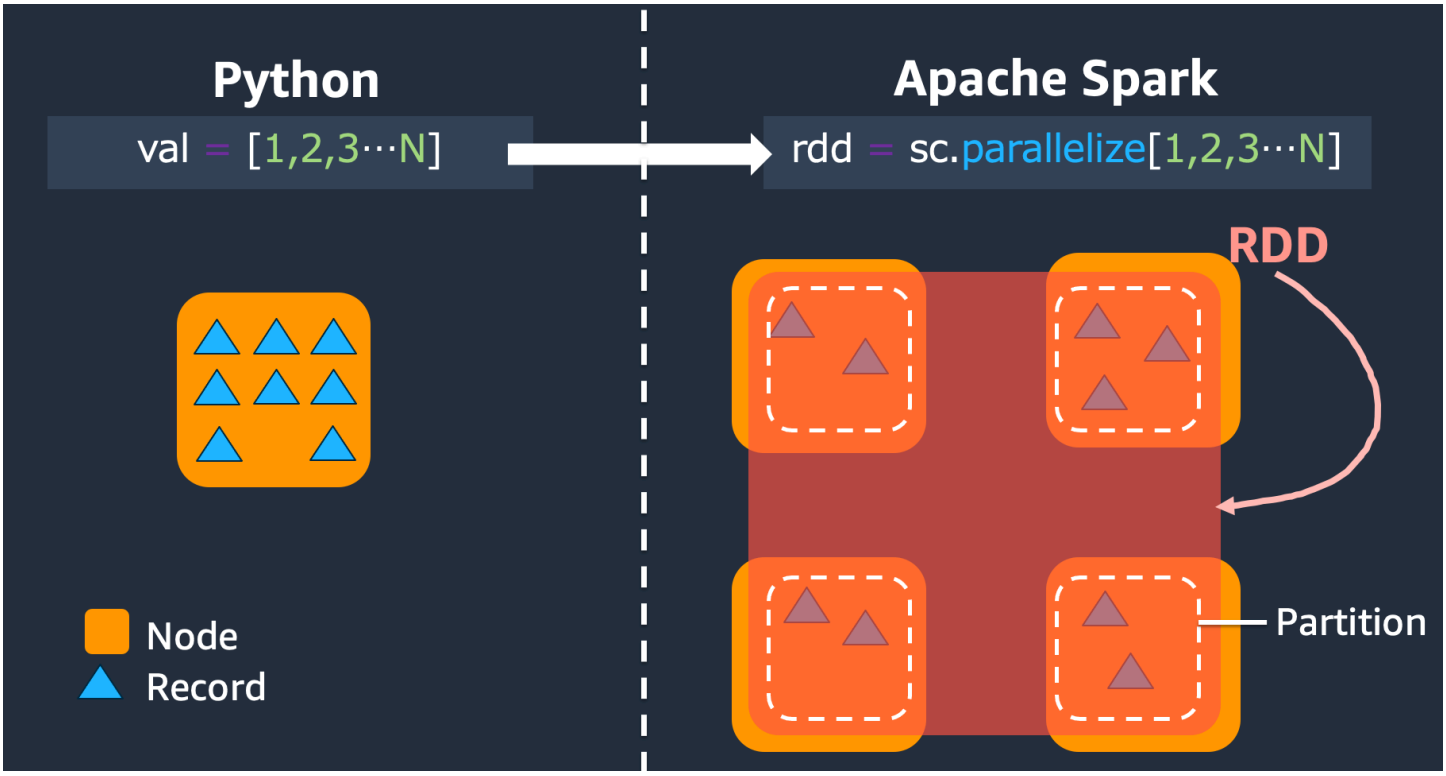


> **Note**
>
> Un ejecutor de Spark tiene múltiples ranuras para procesar múltiples tareas en paralelo. De forma predeterminada, Spark admite una tarea para cada núcleo de CPU virtual (vCPU). Por ejemplo, si un ejecutor tiene cuatro núcleos de CPU, puede ejecutar cuatro tareas simultáneas.

Conjunto de datos distribuido resiliente

Spark realiza la compleja tarea de almacenar y rastrear grandes conjuntos de datos en los ejecutores de Spark. Cuando escribes código para los trabajos de Spark, no necesitas pensar en los detalles del almacenamiento. Spark proporciona la abstracción de conjuntos de datos distribuidos resilientes (RDD), que es una colección de elementos que se pueden operar en paralelo y que se pueden dividir en los ejecutores de Spark del clúster.

La siguiente figura muestra la diferencia en la forma de almacenar datos en la memoria cuando se ejecuta un script de Python en su entorno típico y cuando se ejecuta en el marco Spark (PySpark).



- Python: la escritura `val = [1,2,3...N]` en un script de Python mantiene los datos en la memoria de la única máquina en la que se ejecuta el código.
- PySpark— Spark proporciona la estructura de datos RDD para cargar y procesar los datos distribuidos en la memoria de varios ejecutores de Spark. Puedes generar un RDD con código como`rdd = sc.parallelize[1,2,3...N]`, por ejemplo, y Spark puede distribuir y almacenar automáticamente los datos en la memoria entre varios ejecutores de Spark.

  En muchos AWS Glue trabajos, utilizas los RDD a través AWS Glue DynamicFramesde Spark. DataFrames Se trata de abstracciones que permiten definir el esquema de datos de un RDD y realizar tareas de nivel superior con esa información adicional. Como utilizan los RDD internamente, los datos se distribuyen y cargan de forma transparente en varios nodos en el siguiente 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>")
```

Un RDD tiene las siguientes características:

- Los RDD consisten en datos divididos en varias partes denominadas particiones. Cada ejecutor de Spark almacena una o más particiones en la memoria y los datos se distribuyen entre varios ejecutores.
- Los RDD son inmutables, lo que significa que no se pueden cambiar una vez creados. Para cambiar a DataFrame, puede utilizar las transformaciones, que se definen en la siguiente sección.
- Los RDD replican los datos en los nodos disponibles para que puedan recuperarse automáticamente de los fallos de los nodos.

Evaluación perezosa

Los RDD admiten dos tipos de operaciones: las transformaciones, que crean un nuevo conjunto de datos a partir de uno existente, y las acciones, que devuelven un valor al programa controlador tras ejecutar un cálculo en el conjunto de datos.

- Transformaciones: dado que los RDD son inmutables, solo puede cambiarlos mediante una transformación.

  Por ejemplo, `map` es una transformación que pasa cada elemento del conjunto de datos a través de una función y devuelve un nuevo RDD que representa los resultados. Observe que el `map` método no devuelve una salida. Spark almacena la transformación abstracta para el futuro, en lugar de permitirte interactuar con el resultado. Spark no actuará en función de las transformaciones hasta que solicites una acción.
- Acciones: al usar las transformaciones, construyes tu plan de transformación lógica. Para iniciar el cálculo, ejecute una acción como`write`, `countshow`, `ocollect`.

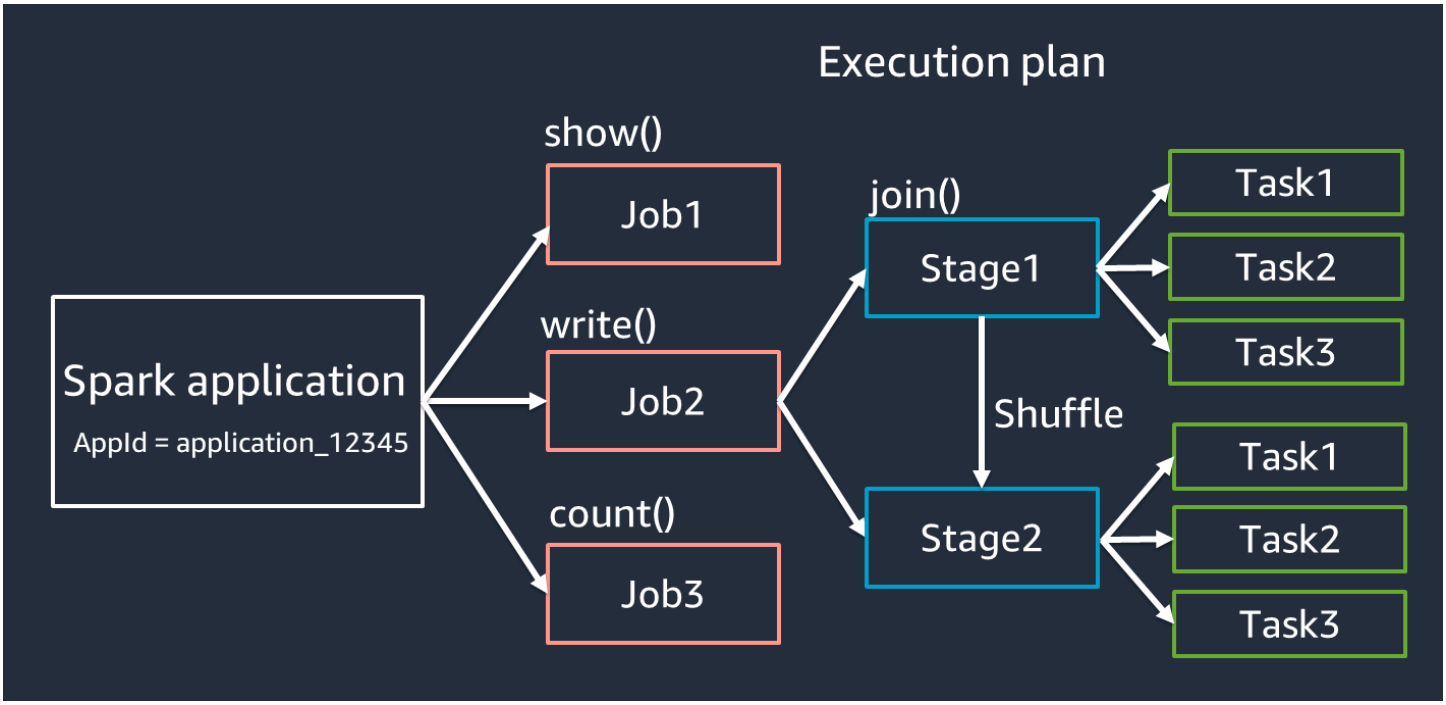
  Todas las transformaciones en Spark son perezosas, ya que no calculan sus resultados de forma inmediata. En su lugar, Spark recuerda una serie de transformaciones aplicadas a algún conjunto de datos base, como los objetos del Amazon Simple Storage Service (Amazon S3). Las transformaciones se calculan solo cuando una acción requiere que se devuelva un

resultado al controlador. Este diseño permite a Spark funcionar de forma más eficiente. Por ejemplo, consideremos la situación en la que un conjunto de datos creado mediante la `map` transformación solo lo consume una transformación que reduce sustancialmente el número de filas, por ejemplo`reduce`. A continuación, puede pasar el conjunto de datos más pequeño que ha sufrido ambas transformaciones al controlador, en lugar de pasar el conjunto de datos mapeado más grande.

Terminología de las aplicaciones de Spark

Esta sección trata sobre la terminología de las aplicaciones de Spark. El controlador Spark crea un plan de ejecución y controla el comportamiento de las aplicaciones en varias abstracciones. Los siguientes términos son importantes para el desarrollo, la depuración y el ajuste del rendimiento con la interfaz de usuario de Spark.

- Aplicación: basada en una sesión de Spark (contexto de Spark). Se identifica mediante un identificador único, como`<application_XXX>`.
- Trabajos: se basan en las acciones creadas para un RDD. Un trabajo consta de una o más etapas.
- Etapas: se basan en las combinaciones creadas para un RDD. Una etapa consta de una o más tareas. La mezcla es el mecanismo de Spark para redistribuir los datos de forma que se agrupen de forma diferente en las particiones del RDD. Algunas transformaciones, por ejemplo`join()`, requieren una mezcla. La mezcla aleatoria se analiza con más detalle en la práctica de ajuste de Optimize shuffles.
- Tareas: una tarea es la unidad mínima de procesamiento programada por Spark. Las tareas se crean para cada partición RDD y el número de tareas es el número máximo de ejecuciones simultáneas en la etapa.

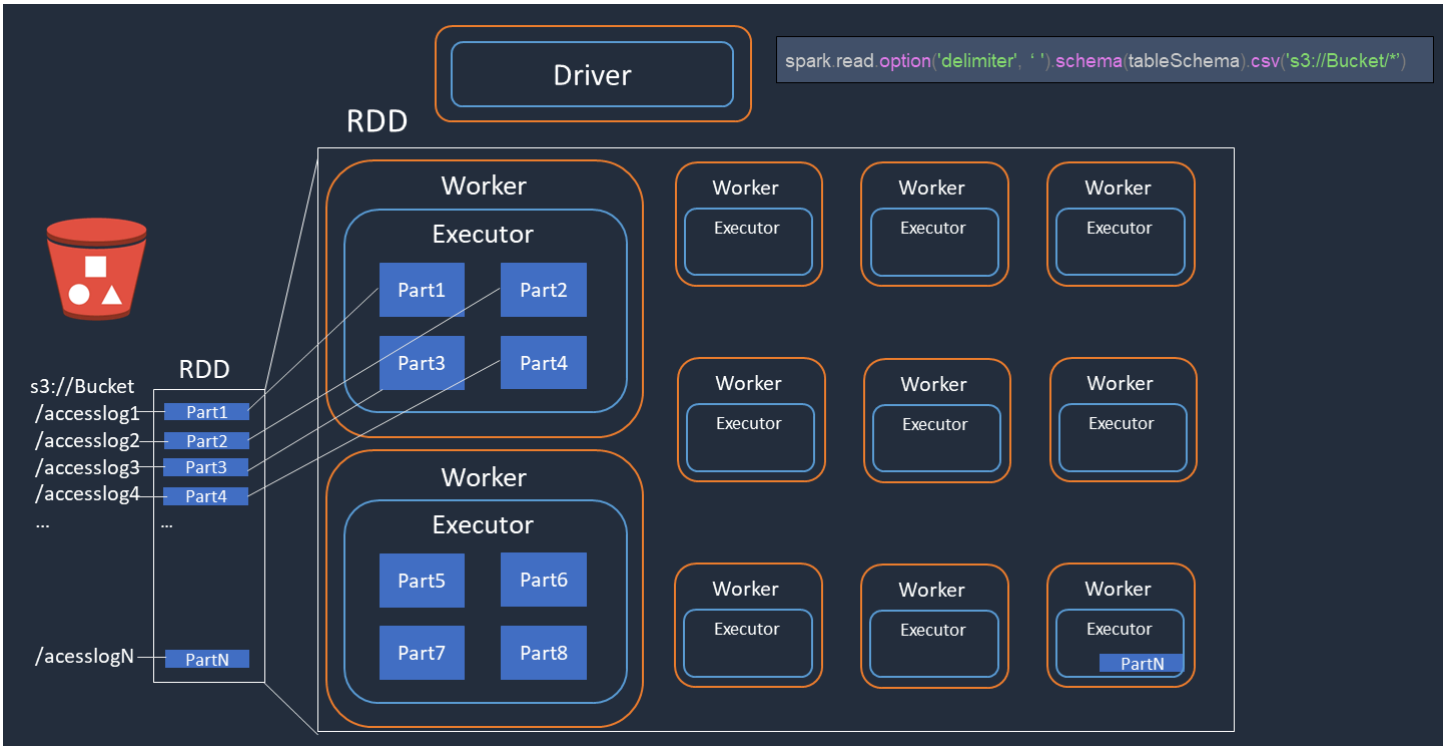


> **Note**
>
> Las tareas son lo más importante a tener en cuenta a la hora de optimizar el paralelismo. El número de tareas aumenta con el número de RDD

Paralelismo

Spark paraleliza las tareas de carga y transformación de datos.

Considere un ejemplo en el que realiza un procesamiento distribuido de los archivos de registro de acceso (denominados`accesslog1 ... accesslog`N) en Amazon S3. El siguiente diagrama muestra el flujo de procesamiento distribuido.

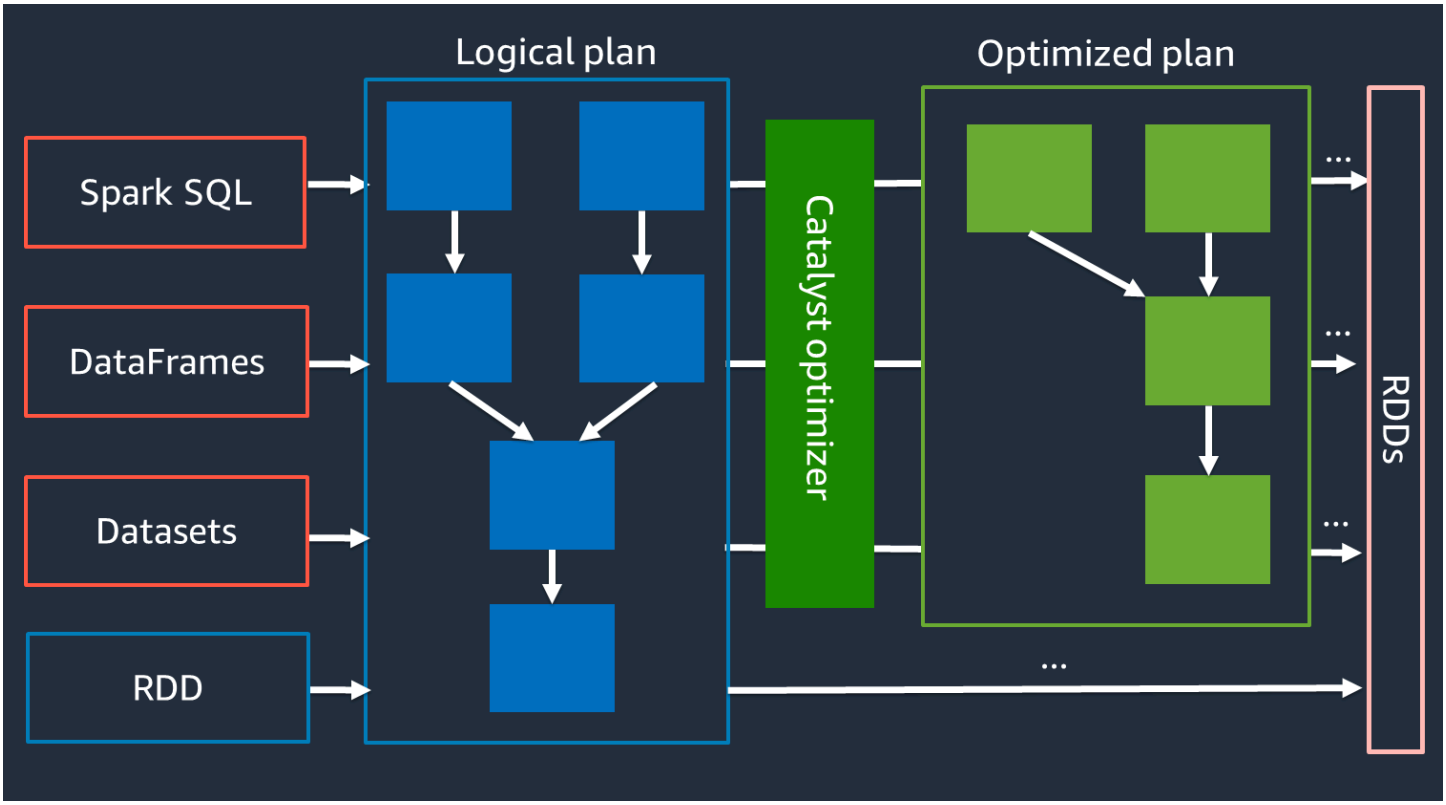


1. El controlador Spark crea un plan de ejecución para el procesamiento distribuido entre muchos ejecutores de Spark.
2. El controlador Spark asigna tareas a cada ejecutor en función del plan de ejecución. De forma predeterminada, el controlador Spark crea particiones RDD (cada una de las cuales corresponde a una tarea de Spark) para cada objeto de S3 (). `Part1 ... N` A continuación, el controlador Spark asigna tareas a cada ejecutor.
3. Cada tarea de Spark descarga su objeto S3 asignado y lo almacena en la memoria de la partición RDD. De esta forma, varios ejecutores de Spark descargan y procesan la tarea asignada en paralelo.

Para obtener más información sobre el número inicial de particiones y la optimización, consulta la sección Paralelizar tareas.

Optimizador Catalyst

Internamente, Spark usa un motor llamado Catalyst optimizer para optimizar los planes de ejecución. Catalyst cuenta con un optimizador de consultas que puede utilizar al ejecutar API de Spark de alto nivel, como Spark SQL, y conjuntos de datosDataFrame, tal y como se describe en el siguiente diagrama.



Como el optimizador de Catalyst no funciona directamente con la API RDD, las API de alto nivel suelen ser más rápidas que las API RDD de bajo nivel. En el caso de uniones complejas, el optimizador Catalyst puede mejorar considerablemente el rendimiento al optimizar el plan de ejecución del trabajo. Puedes ver el plan optimizado de tu trabajo de Spark en la pestaña SQL de la interfaz de usuario de Spark.

Ejecución de consultas adaptativa

El optimizador Catalyst optimiza el tiempo de ejecución mediante un proceso denominado Adaptive Query Execution. Adaptive Query Execution utiliza estadísticas de tiempo de ejecución para volver a optimizar el plan de ejecución de las consultas mientras el trabajo está en ejecución. Adaptive Query Execution ofrece varias soluciones a los desafíos de rendimiento, como la fusión de particiones posteriores a la reorganización, la conversión de uniones por orden y fusión en uniones por transmisión y la optimización de las uniones sesgadas, tal y como se describe en las siguientes secciones.

La ejecución adaptativa de consultas está disponible en la AWS Glue versión 3.0 y versiones posteriores, y está habilitada de forma predeterminada en AWS Glue la versión 4.0 (Spark 3.3.0) y versiones posteriores. La ejecución adaptativa de consultas se puede activar y desactivar utilizando `spark.conf.set("spark.sql.adaptive.enabled", "true")` tu código.

Combinación de particiones posteriores a la reproducción aleatoria

Esta función reduce las particiones RDD (se fusionan) después de cada mezcla en función de las estadísticas de salida. `map` Simplifica el ajuste del número de particiones de reproducción aleatoria al ejecutar consultas. No necesita establecer un número de partición aleatoria que se ajuste a su conjunto de datos. Spark puede elegir el número de partición aleatoria adecuado en tiempo de ejecución una vez que el número inicial de particiones aleatorio sea lo suficientemente grande.

La fusión de particiones posteriores a la reproducción aleatoria está habilitada cuando ambas están configuradas como verdaderas. `spark.sql.adaptive.enabled` `spark.sql.adaptive.coalescePartitions.enabled` Para obtener más información, consulte la documentación de Apache Spark.

Convertir sort-merge join en broadcast join

Esta función reconoce cuando se unen dos conjuntos de datos de un tamaño sustancialmente diferente y adopta un algoritmo de unión más eficiente basado en esa información. Para obtener más información, consulte la documentación de Apache Spark. Las estrategias de unión se describen en la sección Optimizar las combinaciones.

Optimización de uniones sesgadas

La asimetría de los datos es uno de los obstáculos más comunes en los trabajos de Spark. Describe una situación en la que los datos se desvían hacia particiones RDD específicas (y, en consecuencia, hacia tareas específicas), lo que retrasa el tiempo total de procesamiento de la aplicación. A menudo, esto puede reducir el rendimiento de las operaciones de unión. La función de optimización de uniones asimétricas gestiona de forma dinámica la asimetría en las uniones de clasificación y fusión dividiendo (y replicando si es necesario) las tareas asimétricas en tareas de tamaño aproximadamente uniforme.

Esta función está habilitada cuando se establece en true. `spark.sql.adaptive.skewJoin.enabled` Para obtener más información, consulte la documentación de Apache Spark. El sesgo de los datos se analiza con más detalle en la sección Optimizar las mezclas.

Investiga los problemas de rendimiento mediante la interfaz de usuario de Spark

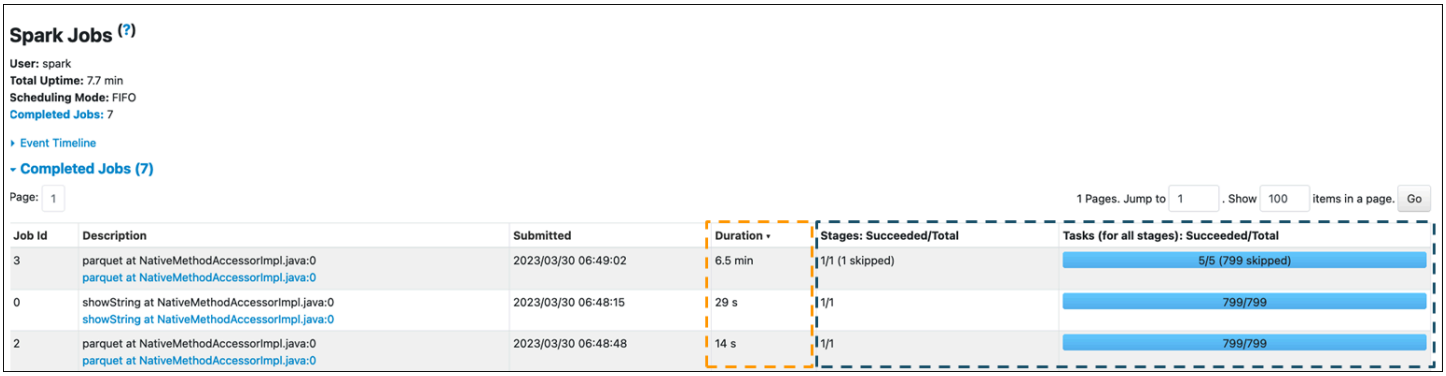
Antes de aplicar las mejores prácticas para ajustar el rendimiento de sus AWS Glue trabajos, le recomendamos encarecidamente que perfile el rendimiento e identifique los cuellos de botella. Esto le ayudará a centrarse en las cosas correctas.

Para un análisis rápido, CloudWatch las métricas de Amazon proporcionan una vista básica de las métricas de tu trabajo. La interfaz de usuario de Spark proporciona una visión más profunda para ajustar el rendimiento. Para usar la interfaz de usuario de Spark con AWS Glue, debes habilitar la interfaz de usuario de Spark para tus AWS Glue trabajos. Una vez que te familiarices con la interfaz de usuario de Spark, sigue las estrategias para ajustar el rendimiento laboral de Spark a fin de identificar y reducir el impacto de los cuellos de botella en función de tus hallazgos.

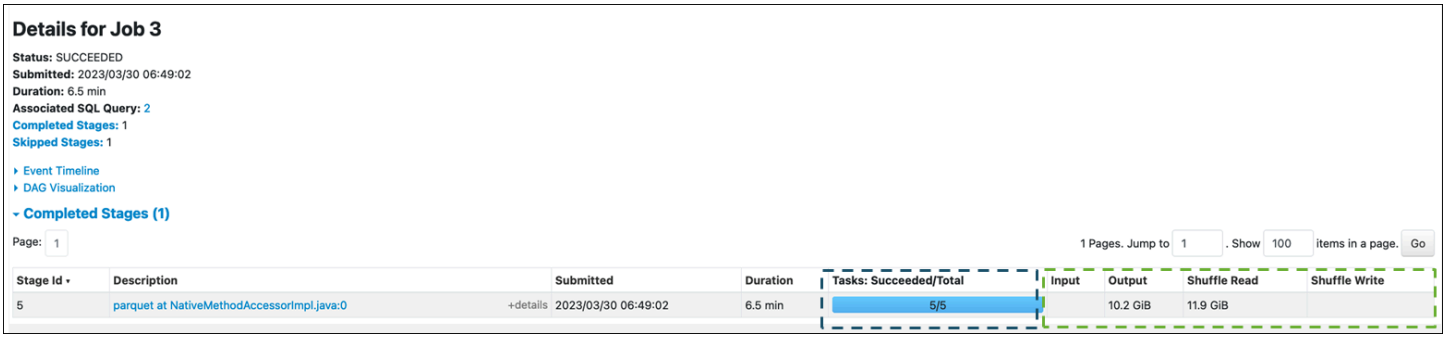
Identifica los cuellos de botella mediante la interfaz de usuario de Spark

Al abrir la interfaz de usuario de Spark, las aplicaciones de Spark aparecen en una tabla. De forma predeterminada, el nombre de la aplicación de un AWS Glue trabajo es`nativespark-<Job Name>-<Job Run ID>`. Elige la aplicación Spark de destino en función del ID de ejecución del trabajo para abrir la pestaña Trabajos. Las ejecuciones de tareas incompletas, como las ejecuciones de tareas en streaming, aparecen en Mostrar solicitudes incompletas.

La pestaña Trabajos muestra un resumen de todos los trabajos de la aplicación Spark. Para determinar los errores de alguna etapa o tarea, comprueba el número total de tareas. Para encontrar los cuellos de botella, clasifíquelos seleccionando Duración. Consulta los detalles de los trabajos de larga duración seleccionando el enlace que se muestra en la columna Descripción.



La página Detalles del trabajo muestra las etapas. En esta página, puedes ver información general, como la duración, el número de tareas realizadas y en total, el número de entradas y salidas y la cantidad de lectura aleatoria y escritura aleatoria.



Details for Job 3

Status: SUCCEEDED
Submitted: 2023/03/30 06:49:02
Duration: 6.5 min
Associated SQL Query: 2
Completed Stages: 1
Skipped Stages: 1

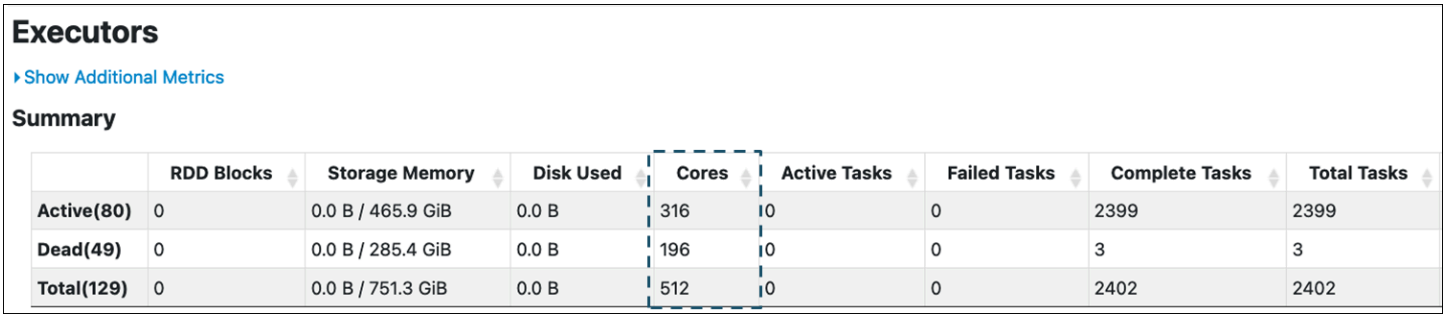
▸ Event Timeline
▸ DAG Visualization

▾ **Completed Stages (1)**

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
5	parquet at NativeMethodAccessorImpl.java:0 +details	2023/03/30 06:49:02	6.5 min	5/5		10.2 GiB	11.9 GiB	

La pestaña Executor muestra en detalle la capacidad del clúster de Spark. Puede comprobar el número total de núcleos. El clúster que se muestra en la siguiente captura de pantalla contiene 316 núcleos activos y 512 núcleos en total. De forma predeterminada, cada núcleo puede procesar una tarea de Spark al mismo tiempo.



Executors

▸ Show Additional Metrics

Summary

	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

Según el valor que 5/5 se muestra en la página Detalles del trabajo, la etapa 5 es la más larga, pero solo utiliza 5 núcleos de un total de 512. Como el paralelismo de esta etapa es muy bajo, pero lleva mucho tiempo, puede identificarlo como un cuello de botella. Para mejorar el rendimiento, debes entender por qué. Para obtener más información sobre cómo reconocer y reducir el impacto de los obstáculos de rendimiento más comunes, consulta Estrategias para optimizar el rendimiento laboral de Spark.

Estrategias para ajustar el desempeño laboral de Spark

Cuando se prepare para ajustar los parámetros, siga las siguientes prácticas recomendadas:

- Determine sus objetivos de rendimiento antes de empezar a identificar los problemas.
- Utilice las métricas para identificar los problemas antes de intentar cambiar los parámetros de ajuste.

Para obtener resultados más consistentes al afinar un trabajo, desarrolle una estrategia básica para su trabajo de ajuste.

Estrategia básica para ajustar el rendimiento

Por lo general, el ajuste del rendimiento se realiza en el siguiente flujo de trabajo:

1. Determine los objetivos de rendimiento.
2. Mida las métricas.
3. Identifique los cuellos de botella.
4. Reduzca el impacto de los cuellos de botella.
5. Repita los pasos 2 a 4 hasta alcanzar el objetivo deseado.

En primer lugar, determine sus objetivos de rendimiento. Por ejemplo, uno de tus objetivos podría ser completar una AWS Glue tarea en un plazo de 3 horas. Después de definir tus objetivos, mide las métricas de desempeño laboral. Identifica las tendencias en las métricas y los cuellos de botella para cumplir los objetivos. En particular, identificar los cuellos de botella es lo más importante para solucionar problemas, depurar y ajustar el rendimiento. Durante la ejecución de una aplicación de Spark, Spark registra el estado y las estadísticas de cada tarea en el registro de eventos de Spark.

En AWS Glue, puedes ver las métricas de Spark a través de la interfaz web de Spark que proporciona el servidor de historial de Spark. AWS Glue para Spark, los trabajos pueden enviar los registros de eventos de Spark a la ubicación que especifique en Amazon S3. AWS Glue también proporciona una AWS CloudFormation plantilla de ejemplo y un Dockerfile para iniciar el servidor de historial de Spark en una EC2 instancia de Amazon o en tu ordenador local, de forma que puedas usar la interfaz de usuario de Spark con los registros de eventos.

Una vez que hayas determinado tus objetivos de rendimiento e identificado las métricas para evaluarlos, puedes empezar a identificar y corregir los cuellos de botella mediante las estrategias que se describen en las siguientes secciones.

Perfeccionar las prácticas para mejorar el desempeño laboral de Spark

Puedes usar las siguientes estrategias para ajustar el rendimiento de los trabajos AWS Glue de Spark:

- AWS Glue recursos:
  - Amplíe la capacidad del clúster
  - Utilice la última versión AWS Glue
- Aplicaciones Spark:
  - Reduzca la cantidad de datos escaneados
  - Paraleliza las tareas
  - Optimice las combinaciones
  - Minimice los gastos de planificación
  - Optimice las funciones definidas por el usuario

Antes de usar estas estrategias, debes tener acceso a las métricas y la configuración de tu trabajo en Spark. Puedes encontrar esta información en la AWS Glue documentación.

Desde la perspectiva de los AWS Glue recursos, puede lograr mejoras en el rendimiento añadiendo AWS Glue trabajadores y utilizando la AWS Glue versión más reciente.

Desde la perspectiva de la aplicación Apache Spark, tiene acceso a varias estrategias que pueden mejorar el rendimiento. Si se cargan datos innecesarios en el clúster de Spark, puede eliminarlos para reducir la cantidad de datos cargados. Si has infrautilizado los recursos del clúster de Spark y tienes pocos datos de E/S, puedes identificar las tareas que deseas paralelizar. También puedes optimizar las operaciones de transferencia de datos pesadas, como las uniones, si requieren mucho tiempo. También puedes optimizar tu plan de consultas de trabajo o reducir la complejidad computacional de las tareas individuales de Spark.

Para aplicar estas estrategias de manera eficiente, debes identificar cuándo son aplicables consultando tus métricas. Para obtener más información, consulta cada una de las siguientes

secciones. Estas técnicas funcionan no solo para ajustar el rendimiento, sino también para resolver problemas típicos, como los errores out-of-memory (OOM).

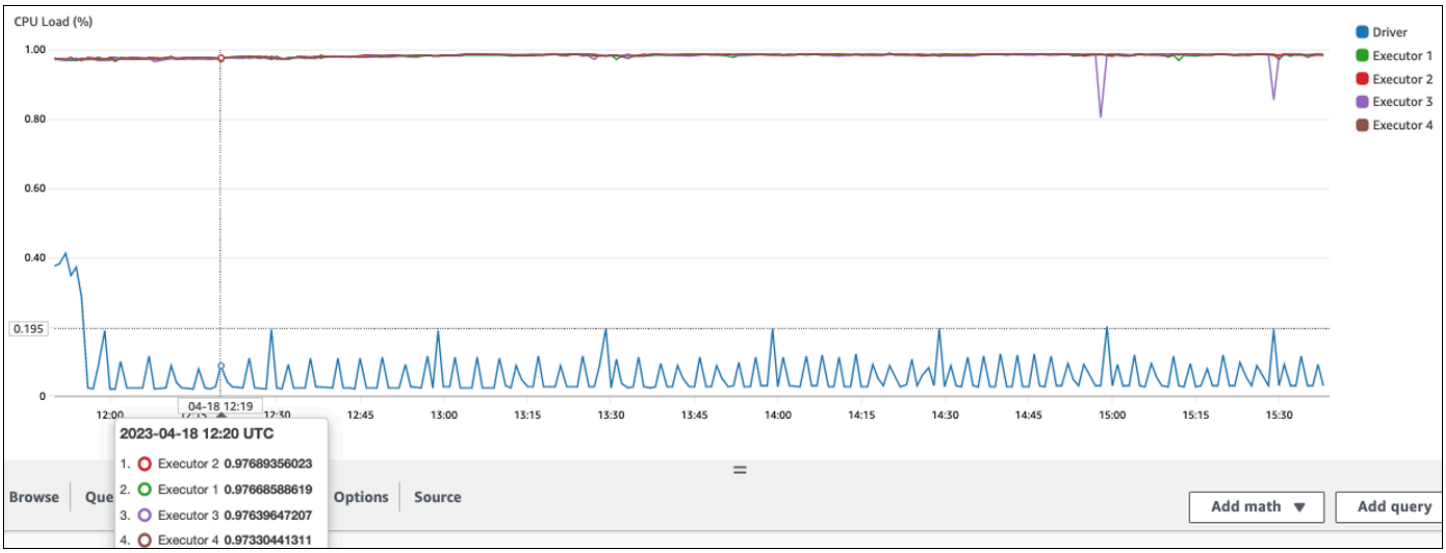
Amplíe la capacidad del clúster

Si tu trabajo te lleva demasiado tiempo, pero los ejecutores consumen suficientes recursos y Spark crea un gran volumen de tareas en relación con los núcleos disponibles, considera la posibilidad de ampliar la capacidad del clúster. Para evaluar si esto es apropiado, usa las siguientes métricas.

CloudWatch métricas

- Compruebe la CPUcarga y el uso de la memoria para determinar si los ejecutores consumen recursos suficientes.
- Compruebe cuánto tiempo ha durado el trabajo para evaluar si el tiempo de procesamiento es demasiado largo para cumplir sus objetivos de rendimiento.

En el siguiente ejemplo, cuatro ejecutores se ejecutan con una CPU carga superior al 97 por ciento, pero el procesamiento no se ha completado después de unas tres horas.

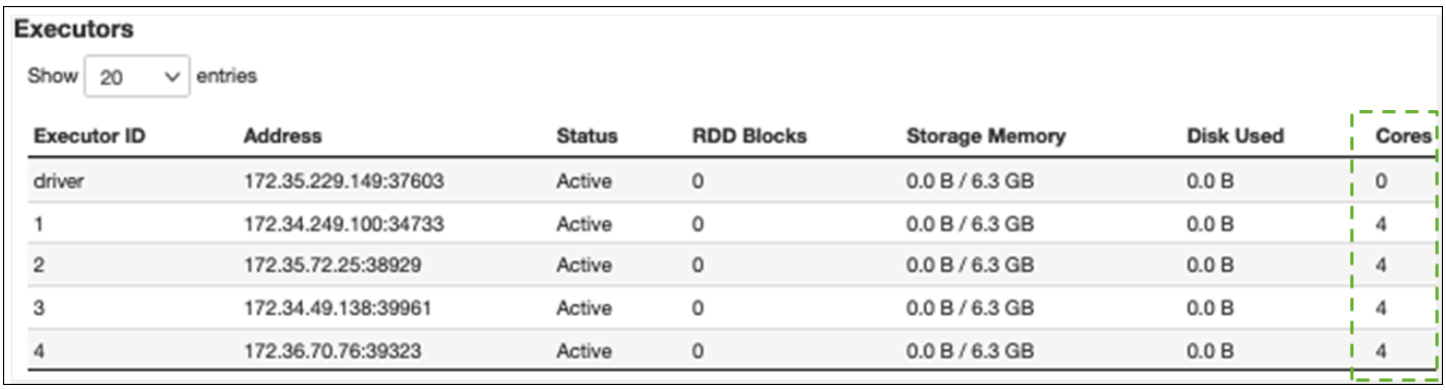


> **Note**
>
> Si la CPU carga es baja, es probable que no se beneficie de la ampliación de la capacidad del clúster.

Interfaz de usuario de Spark

En la pestaña Trabajo o Etapa, puede ver el número de tareas de cada trabajo o etapa. En el siguiente ejemplo, Spark ha creado `58100` tareas.

En la pestaña Ejecutor, puedes ver el número total de ejecutores y tareas. En la siguiente captura de pantalla, cada ejecutor de Spark tiene cuatro núcleos y puede realizar cuatro tareas simultáneamente.



En este ejemplo, el número de tareas de Spark (`58100`)es mucho mayor que las 16 tareas que los ejecutores pueden procesar simultáneamente (4 ejecutores × 4 núcleos).

Si observas estos síntomas, considera la posibilidad de escalar el clúster. Puede escalar la capacidad del clúster mediante las siguientes opciones:

- Activar AWS Glue Auto Scaling: Auto Scaling está disponible para sus trabajos de AWS Glue extracción, transformación, carga (ETL) y streaming en la AWS Glue versión 3.0 o posterior. AWS Glue agrega y elimina automáticamente trabajadores del clúster en función del número de particiones en cada etapa o de la velocidad a la que se generan los microlotes durante la ejecución del trabajo.

  Si observa una situación en la que el número de trabajadores no aumenta aunque Auto Scaling esté activado, considere agregar trabajadores manualmente. Sin embargo, tenga en cuenta que escalar manualmente para una etapa puede provocar que muchos trabajadores permanezcan inactivos durante las etapas posteriores, lo que podría costar más y no aumentar el rendimiento.

Después de activar Auto Scaling, puede ver el número de ejecutores en las métricas del CloudWatch ejecutor. Usa las siguientes métricas para monitorear la demanda de ejecutores en las aplicaciones de Spark:

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

Para obtener más información sobre las métricas, consulta Cómo monitorizar AWS Glue con CloudWatch las métricas de Amazon.

- Amplíe la escala: aumente la cantidad de AWS Glue trabajadores: puede aumentar la cantidad de AWS Glue trabajadores manualmente. Agregue trabajadores solo hasta que observe a los trabajadores inactivos. En ese momento, agregar más trabajadores aumentará los costos sin mejorar los resultados. Para obtener más información, consulte Paralelizar tareas.
- Amplíe: utilice un tipo de trabajador más grande: puede cambiar manualmente el tipo de instancia de sus AWS Glue trabajadores para utilizar trabajadores con más núcleos, memoria y almacenamiento. Los tipos de trabajadores más grandes le permiten escalar verticalmente y ejecutar tareas de integración de datos intensivas, como transformaciones de datos que consumen mucha memoria, agregaciones asimétricas y comprobaciones de detección de entidades con petabytes de datos.

  La ampliación también es útil en los casos en que el controlador de Spark necesita una mayor capacidad, por ejemplo, porque el plan de búsqueda de empleo es bastante amplio. Para obtener más información sobre los tipos de trabajadores y su rendimiento, consulte la entrada del blog sobre AWS macrodatos: escale sus trabajos AWS Glue para Apache Spark con nuevos tipos de trabajadores más grandes, G.4X y G.8X.

  El uso de trabajadores más grandes también puede reducir la cantidad total de trabajadores necesarios, lo que aumenta el rendimiento al reducir la confusión en operaciones intensivas, como la incorporación de personal.

Utilice la última versión AWS Glue

Recomendamos utilizar la AWS Glue versión más reciente. Hay varias optimizaciones y actualizaciones integradas en cada versión que pueden mejorar automáticamente el rendimiento laboral. Por ejemplo, la AWS Glue versión 4.0 ofrece las siguientes funciones nuevas:

- Nuevo entorno de ejecución optimizado de Apache Spark 3.3.0: la AWS Glue versión 4.0 se basa en el entorno de ejecución de Apache Spark 3.3.0 y ofrece mejoras de rendimiento comparables a las de Spark de código abierto. El tiempo de ejecución de Spark 3.3.0 se basa en muchas de las innovaciones de Spark 2.x.
- Conector Amazon Redshift mejorado: las versiones AWS Glue 4.0 y posteriores proporcionan la integración de Amazon Redshift para Apache Spark. La integración se basa en un conector de código abierto existente y lo mejora en términos de rendimiento y seguridad. La integración ayuda a que las aplicaciones funcionen hasta 10 veces más rápido. Para obtener más información, consulte la entrada del blog sobre la integración de Amazon Redshift con Apache Spark.
- SIMDejecución basada en lecturas vectorizadas con JSON datos CSV y lecturas: la AWS Glue versión 3.0 y las versiones posteriores incorporan lectores optimizados que pueden acelerar considerablemente el rendimiento general del trabajo en comparación con los lectores basados en filas. Para obtener más información sobre CSV los datos, consulte Optimizar el rendimiento de lectura con un lector vectorizado. SIMD CSV Para obtener más información sobre JSON los datos, consulte Uso de un SIMD JSON lector vectorizado con el formato de columnas Apache Arrow.

Cada AWS Glue versión incluirá actualizaciones de este tipo, entre otras muchas, como actualizaciones de conectores, controladores y bibliotecas. Para obtener más información, consulte AWS Glue Versiones y Migración de AWS Glue trabajos a la AWS Glue versión 4.0.

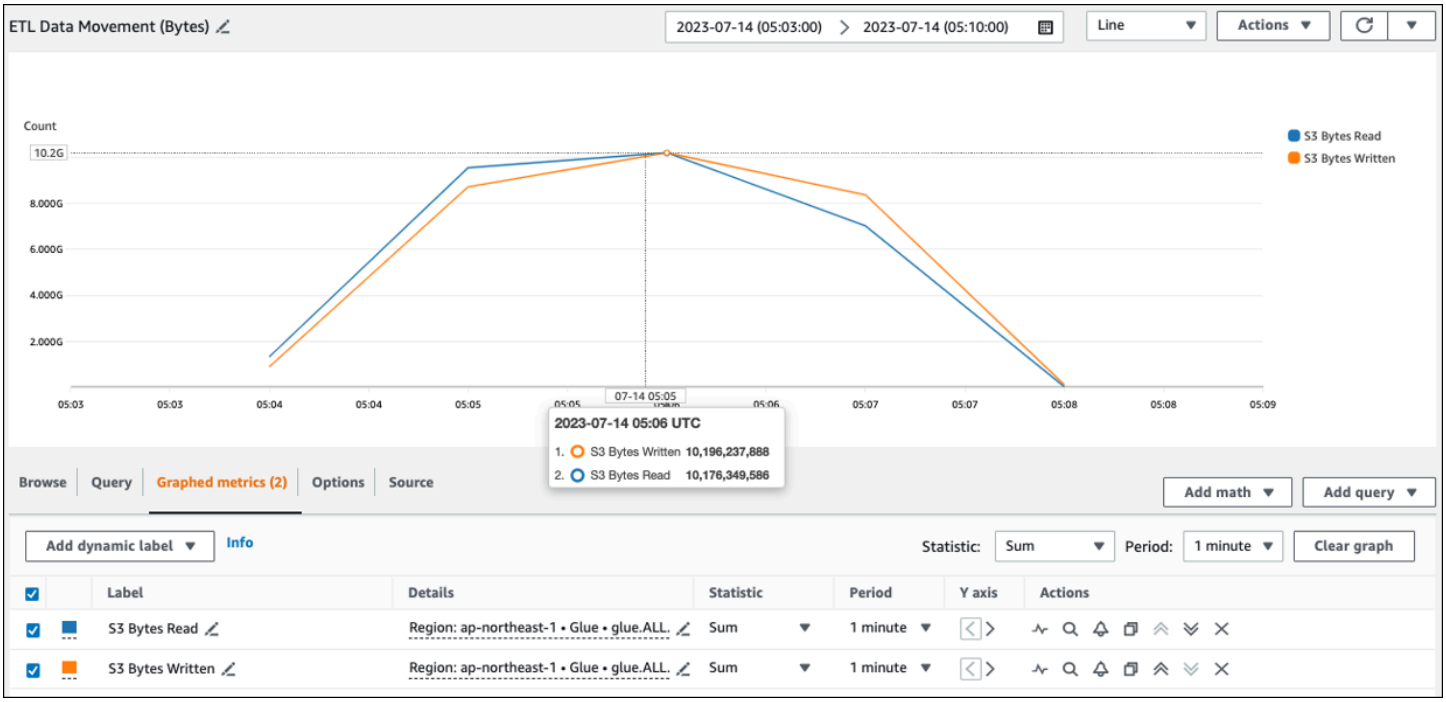
Reduzca la cantidad de datos escaneados

Para empezar, considera cargar solo los datos que necesites. Puedes mejorar el rendimiento simplemente reduciendo la cantidad de datos que se cargan en tu clúster de Spark para cada fuente de datos. Para evaluar si este enfoque es apropiado, usa las siguientes métricas.

Puedes comprobar los bytes leídos de Amazon S3 en CloudWatchlas métricas y obtener más detalles en la interfaz de usuario de Spark, tal y como se describe en la sección de interfaz de usuario de Spark.

CloudWatch métricas

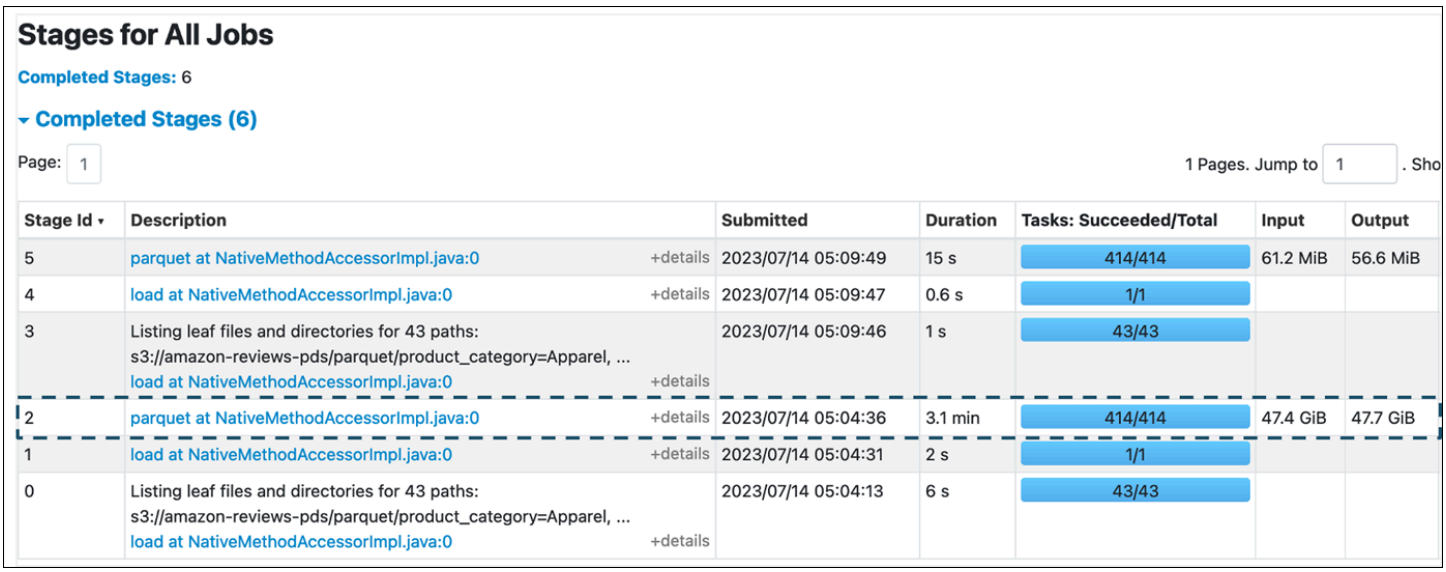
Puede ver el tamaño de lectura aproximado de Amazon S3 en ETLData Movement (bytes). Esta métrica muestra el número de bytes leídos de Amazon S3 por todos los ejecutores desde el informe anterior. Puede utilizarla para supervisar el movimiento de ETL datos desde Amazon S3 y comparar las tasas de lectura con las tasas de ingesta de fuentes de datos externas.



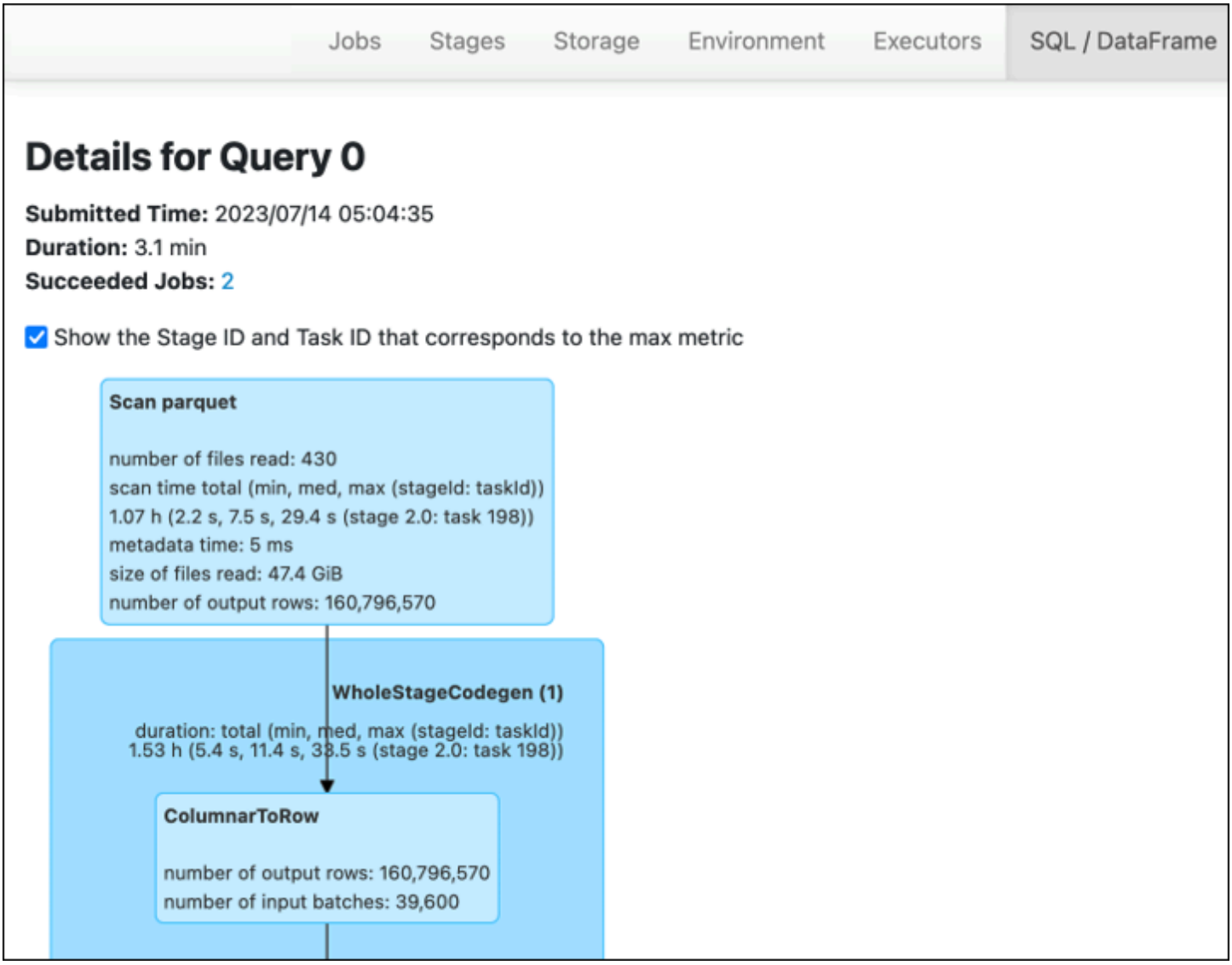
Si observa un punto de datos de lectura de S3 bytes mayor de lo esperado, considere las siguientes soluciones.

Interfaz de usuario de Spark

En la pestaña Stage de la interfaz AWS Glue de usuario de Spark, puedes ver el tamaño de entrada y salida. En el siguiente ejemplo, la etapa 2 lee 47,4 GiB de entrada y 47,7 GiB de salida, mientras que la etapa 5 lee 61,2 MiB de entrada y 56,6 MiB de salida.



Cuando usas la chispa SQL o los DataFrame enfoques en tu AWS Glue trabajo, la pestaña SQL ataFrame /D muestra más estadísticas sobre estas etapas. En este caso, la etapa 2 muestra el número de archivos leídos: 430, el tamaño de los archivos leídos: 47,4 GiB y el número de filas de salida: 160.796.570.



Si observa que hay una diferencia sustancial de tamaño entre los datos que lee y los datos que utiliza, pruebe las siguientes soluciones.

Amazon S3

Para reducir la cantidad de datos que se cargan en su trabajo al leer desde Amazon S3, tenga en cuenta el tamaño, la compresión, el formato y el diseño del archivo (particiones) del conjunto de datos. AWS Glue en el caso ETL de Spark, los trabajos se suelen utilizar para datos sin procesar,

pero para un procesamiento distribuido eficiente, es necesario inspeccionar las características del formato de la fuente de datos.

- Tamaño del archivo: recomendamos mantener el tamaño del archivo de entradas y salidas dentro de un rango moderado (por ejemplo, 128 MB). Los archivos demasiado pequeños y los archivos demasiado grandes pueden causar problemas.

  Un gran número de archivos pequeños provoca los siguientes problemas:

  - La elevada carga de E/S de la red en Amazon S3 se debe a la sobrecarga necesaria para realizar solicitudes (por ejemplo `ListGet`, o`Head`) para muchos objetos (en comparación con unos pocos objetos que almacenan la misma cantidad de datos).
  - El controlador Spark soporta una gran carga de procesamiento y E/S, lo que generará muchas particiones y tareas y generará un paralelismo excesivo.

  Por otro lado, si el tipo de archivo no se puede dividir (por ejemplo, gzip) y los archivos son demasiado grandes, la aplicación Spark debe esperar a que una sola tarea termine de leer todo el archivo.

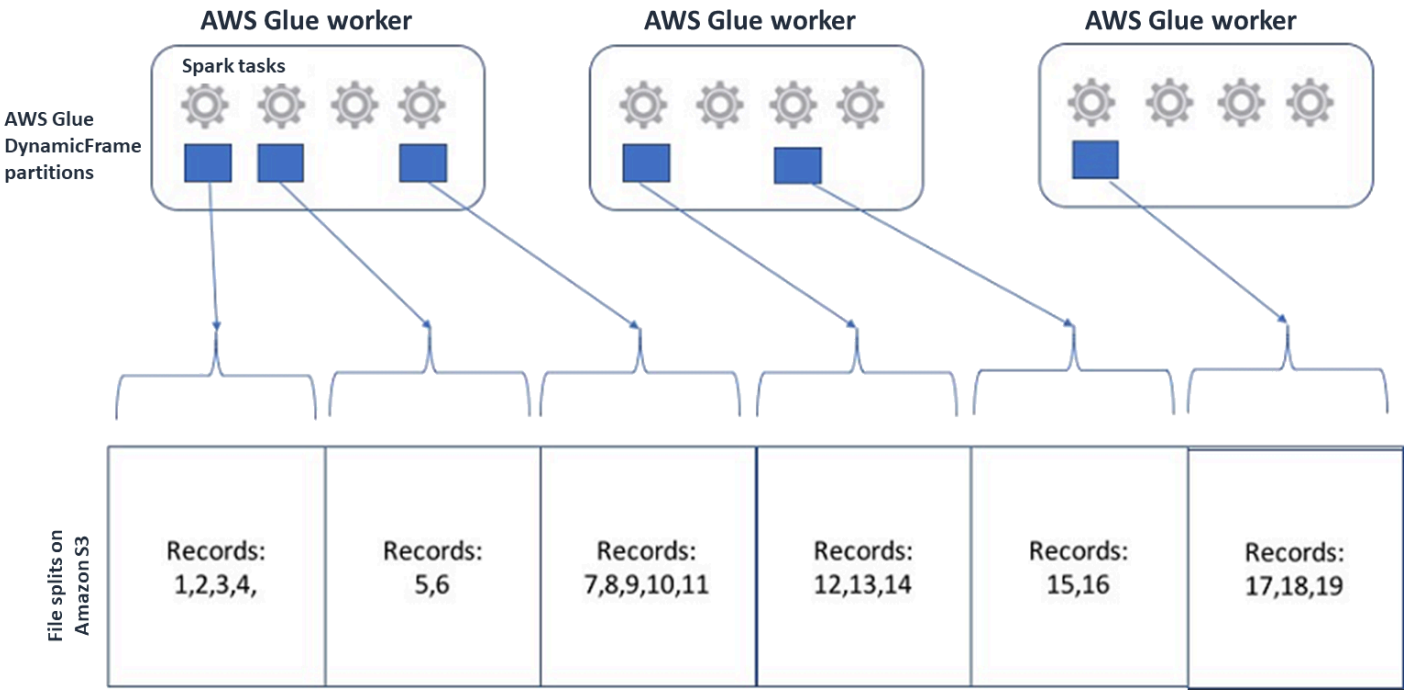
  Para reducir el paralelismo excesivo que se produce al crear una tarea de Apache Spark para cada archivo pequeño, utilice la agrupación de archivos para. DynamicFrames Este enfoque reduce las posibilidades de que se produzca una OOM excepción por parte del controlador Spark. Para configurar la agrupación de archivos, defina los `groupSize` parámetros `groupFiles` y. En el siguiente ejemplo de código, se utiliza AWS Glue DynamicFrame API en un ETL script con estos parámetros.

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

- Compresión: si los objetos de S3 ocupan cientos de megabytes, considere comprimirlos. Existen varios formatos de compresión, que se pueden clasificar a grandes rasgos en dos tipos:

  - Los formatos de compresión que no se pueden dividir, como gzip, requieren que un trabajador descomprima todo el archivo.
  - Los formatos de compresión separables, como bzip2 o LZO (indexado), permiten la descompresión parcial de un archivo, que se puede paralelizar.

Para Spark (y otros motores de procesamiento distribuido comunes), dividirás el archivo de datos fuente en fragmentos que tu motor pueda procesar en paralelo. Estas unidades suelen denominarse divisiones. Una vez que los datos estén en un formato separable, los AWS Glue lectores optimizados pueden recuperar las divisiones de un objeto S3 al ofrecer la `Range` opción de `GetObject` API recuperar solo bloques específicos. Considera el siguiente diagrama para ver cómo funcionaría esto en la práctica.



Los datos comprimidos pueden acelerar considerablemente la aplicación, siempre que los archivos tengan un tamaño óptimo o se puedan dividir. Los tamaños de datos más pequeños reducen los datos escaneados desde Amazon S3 y el tráfico de red desde Amazon S3 a su clúster de Spark. Por otro lado, CPU se necesita más para comprimir y descomprimir los datos. La cantidad de cómputo requerida aumenta con la relación de compresión del algoritmo de compresión. Tenga en cuenta esta desventaja a la hora de elegir el formato de compresión separable.

> **Note**
>
> Si bien los archivos gzip generalmente no se pueden dividir, puedes comprimir bloques de parquet individuales con gzip y esos bloques se pueden paralelizar.

- Formato de archivo: utilice un formato de columnas. Apache Parquet y Apache ORC son formatos de datos en columnas populares. Comprima y ORC almacene los datos de manera eficiente mediante la compresión basada en columnas, codificando y comprimiendo cada columna en función de su tipo de datos. Para obtener más información sobre las codificaciones de Parquet, consulte Definiciones de codificación de Parquet. Los archivos de parquet también se pueden dividir.

  Los formatos en columnas agrupan los valores por columnas y los almacenan juntos en bloques. Al usar formatos en columnas, puede omitir los bloques de datos que corresponden a columnas que no planea usar. Las aplicaciones de Spark solo pueden recuperar las columnas que necesitas. Por lo general, unas relaciones de compresión mejores o la omisión de bloques de datos implican leer menos bytes de Amazon S3, lo que se traduce en un mejor rendimiento. Ambos formatos también admiten los siguientes enfoques de compresión descendente para reducir la E/S:

  - Empuje por proyección: la compresión por proyección es una técnica para recuperar solo las columnas especificadas en la aplicación. Las columnas se especifican en la aplicación Spark, como se muestra en los siguientes ejemplos:
    - DataFrame ejemplo: `df.select("star_rating")`
    - SQLEjemplo de Spark: `spark.sql("select start_rating from <table>")`
  - Depresión de predicados: la compresión de predicados es una técnica para procesar cláusulas de forma eficiente. `WHERE GROUP BY` Ambos formatos tienen bloques de datos que representan valores de columnas. Cada bloque contiene las estadísticas del bloque, como los valores máximo y mínimo. Spark puede usar estas estadísticas para determinar si el bloque debe leerse u omitirse en función del valor del filtro utilizado en la aplicación. Para utilizar esta función, añade más filtros en las condiciones, como se muestra en los siguientes ejemplos:
    - DataFrame ejemplo: `df.select("star_rating").filter("star_rating < 2")`
    - SQLEjemplo de Spark: `spark.sql("select * from <table> where star_rating < 2")`
- Diseño de archivos: al almacenar los datos de S3 en objetos situados en diferentes rutas en función de cómo se vayan a utilizar los datos, podrá recuperar los datos relevantes de forma eficiente. Para obtener más información, consulte Organizar objetos mediante prefijos en la documentación de Amazon S3. AWS Glue admite almacenar claves y valores en prefijos de Amazon S3 en este formato`key=value`, dividiendo los datos según la ruta de Amazon S3. Al particionar sus datos, puede restringir la cantidad de datos escaneados por cada aplicación de análisis posterior, lo que mejora el rendimiento y reduce los costos. Para obtener más información, consulte Administrar particiones para su ETL salida. AWS Glue

La partición divide la tabla en diferentes partes y mantiene los datos relacionados en archivos agrupados en función de los valores de las columnas, como el año, el mes y el día, como se muestra en el siguiente ejemplo.

```
# 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
...
```

Puede definir particiones para su conjunto de datos modelándolo con una tabla en. AWS Glue Data Catalog A continuación, puede restringir la cantidad de datos escaneados mediante la reducción de particiones de la siguiente manera:

- Para AWS Glue DynamicFrame, configure `push_down_predicate` (`ocatalogPartitionPredicate`).

  ```
  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'",
  )
  ```

- En el caso de Spark DataFrame, establece una ruta fija para podar las particiones.

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

- En el caso de SparkSQL, puedes configurar la cláusula where para eliminar las particiones del catálogo de datos.

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

- Para particionar por fecha cuando escribas tus datos AWS Glue, introduce DynamicFrame o partitionBy() DataFrame con la información de fecha de tus columnas de la siguiente manera. partitionKeys

  - 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>/')
```

Esto puede mejorar el rendimiento de los consumidores de los datos de salida.

Si no tiene acceso para modificar la canalización que crea el conjunto de datos de entrada, la partición no es una opción. En su lugar, puedes excluir las rutas S3 innecesarias mediante patrones globales. Establece exclusiones al leer. DynamicFrame Por ejemplo, el siguiente código excluye los días de los meses 01 a 09 del año 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'
)
```

También puede establecer exclusiones en las propiedades de la tabla del catálogo de datos:

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

- Demasiadas particiones de Amazon S3: evite particionar los datos de Amazon S3 en columnas que contengan una amplia gama de valores, como una columna de ID con miles de valores. Esto puede aumentar considerablemente el número de particiones del bucket, ya que el número de particiones posibles es el producto de todos los campos por los que ha realizado la partición. Demasiadas particiones pueden provocar lo siguiente:
  - Aumento de la latencia para recuperar los metadatos de las particiones del catálogo de datos
  - Mayor número de archivos pequeños, lo que requiere más API solicitudes de Amazon S3 (`ListGet`, `yHead`)

Por ejemplo, si estableces un tipo de fecha en `partitionBy` o`partitionKeys`, la partición a nivel de fecha `yyyy/mm/dd` es adecuada para muchos casos de uso. Sin embargo, `yyyy/mm/dd/<ID>` podría generar tantas particiones que afectaría negativamente al rendimiento en su conjunto.

Por otro lado, algunos casos de uso, como las aplicaciones de procesamiento en tiempo real, requieren muchas particiones, por ejemplo`yyyy/mm/dd/hh`. Si su caso de uso requiere particiones importantes, considere la posibilidad de utilizar índices de AWS Glue particiones para reducir la latencia a la hora de recuperar los metadatos de las particiones del catálogo de datos.

Bases de datos y JDBC

Para reducir el escaneo de datos al recuperar información de una base de datos, puede especificar un `where` predicado (o cláusula) en una SQL consulta. Las bases de datos que no proporcionan una SQL interfaz proporcionarán su propio mecanismo de consulta o filtrado.

Cuando utilice conexiones de conectividad de bases de datos Java (JDBC), proporcione una consulta de selección con la `where` cláusula para los siguientes parámetros:

- Para DynamicFrame, utilice la sampleQueryopción. Cuando `create_dynamic_frame.from_catalog` lo utilice, configure el `additional_options` argumento de la siguiente manera.

```
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"
)
```

Cuando`using create_dynamic_frame.from_options`, configure el `connection_options` argumento de la siguiente manera.

```
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, utilice la opción 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()
```

- En el caso de Amazon Redshift, utilice la AWS Glue versión 4.0 o una versión posterior para aprovechar la compatibilidad con pulsaciones descendentes del 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 otras bases de datos, consulte la documentación de esa base de datos.

AWS Glue opciones

- Para evitar un análisis completo de todas las ejecuciones continuas de tareas y procesar solo los datos que no estaban presentes durante la última ejecución de la tarea, habilite los marcadores de tareas.
- Para limitar la cantidad de datos de entrada que se van a procesar, habilite la ejecución limitada con marcadores de trabajos. Esto ayuda a reducir la cantidad de datos escaneados para cada ejecución de trabajo.

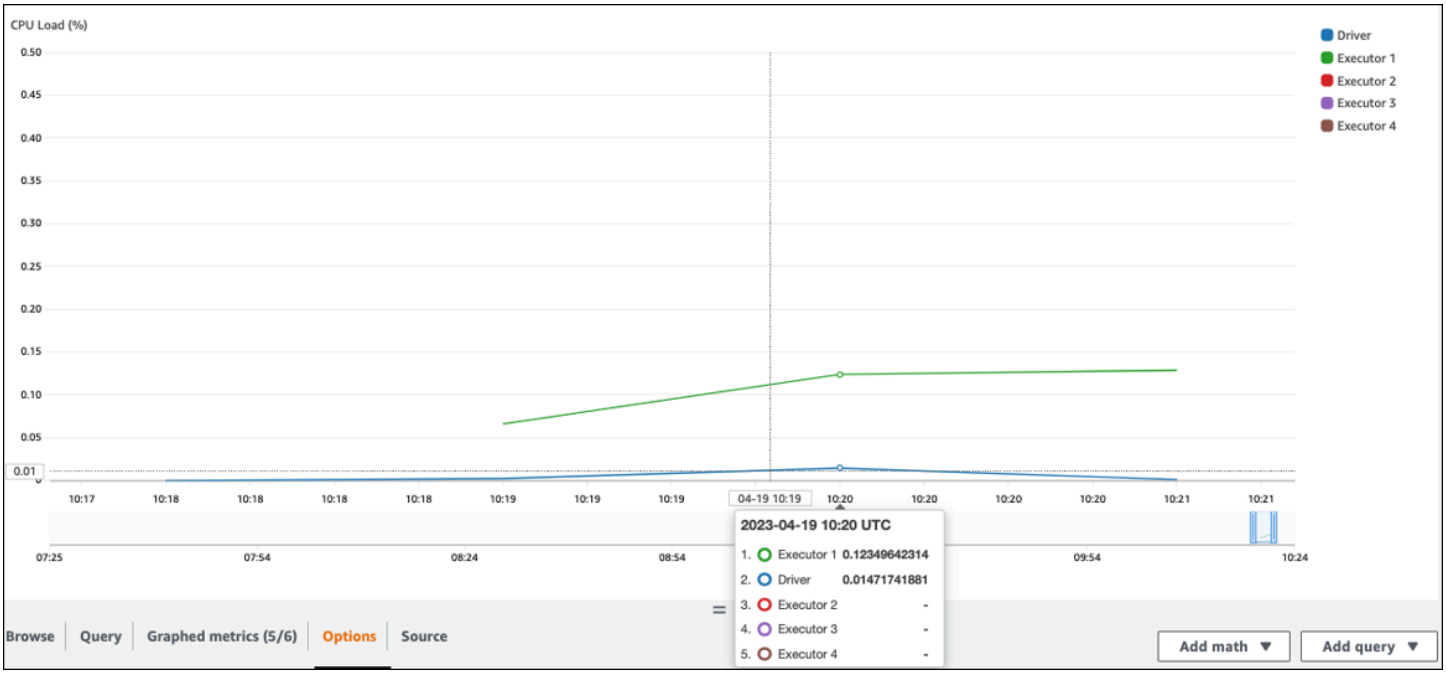
Paraleliza las tareas

Para optimizar el rendimiento, es importante paralelizar las tareas de carga y transformación de datos. Como explicamos en Temas clave de Apache Spark, el número de particiones resilientes de conjuntos de datos distribuidos (RDD) es importante porque determina el grado de paralelismo. Cada tarea que crea Spark corresponde a una RDD partición en proporción 1:1. Para lograr el mejor rendimiento, debes entender cómo se determina el número de RDD particiones y cómo se optimiza ese número.

Si no tienes suficiente paralelismo, los siguientes síntomas se registrarán en las CloudWatchmétricas y en la interfaz de usuario de Spark.

CloudWatch métricas

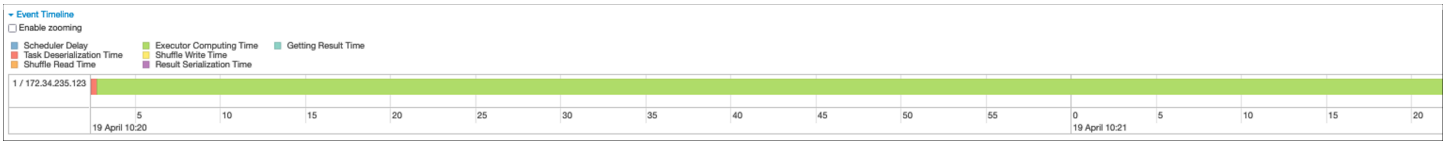
Compruebe la CPUcarga y el uso de la memoria. Si algunos ejecutores no se procesan durante una fase de su trabajo, es conveniente mejorar el paralelismo. En este caso, durante el período de tiempo visualizado, el ejecutor 1 estaba realizando una tarea, pero los ejecutores restantes (2, 3 y 4) no. Se puede deducir que el controlador de Spark no asignó tareas a esos ejecutores.



Interfaz de usuario de Spark

En la pestaña Escenario de la interfaz de usuario de Spark, puedes ver el número de tareas de una etapa. En este caso, Spark solo ha realizado una tarea.

Además, la cronología del evento muestra al Ejecutor 1 procesando una tarea. Esto significa que el trabajo de esta etapa se ejecutó íntegramente con un ejecutor, mientras que los demás estaban inactivos.



Si observa estos síntomas, pruebe las siguientes soluciones para cada fuente de datos.

Paralelizar la carga de datos desde Amazon S3

Para paralelizar las cargas de datos de Amazon S3, compruebe primero el número predeterminado de particiones. A continuación, puede determinar manualmente el número objetivo de particiones, pero asegúrese de evitar tener demasiadas particiones.

Determine el número predeterminado de particiones

En Amazon S3, el número inicial de RDD particiones de Spark (cada una de las cuales corresponde a una tarea de Spark) viene determinado por las características del conjunto de datos de Amazon S3 (por ejemplo, el formato, la compresión y el tamaño). Al crear un Spark AWS Glue DynamicFrame o un Spark DataFrame a partir de CSV objetos almacenados en Amazon S3, el número inicial de RDD particiones (`NumPartitions`) se puede calcular aproximadamente de la siguiente manera:

- Tamaño del objeto <= 64 MB: `NumPartitions = Number of Objects`
- Tamaño del objeto > 64 MB: `NumPartitions = Total Object Size / 64 MB`
- No se puede dividir (gzip): `NumPartitions = Number of Objects`

Como se explica en la sección Reducir la cantidad de datos escaneados, Spark divide los objetos S3 grandes en divisiones que se pueden procesar en paralelo. Cuando el objeto es más grande que el tamaño de la división, Spark divide el objeto y crea una RDD partición (y una tarea) para cada división. El tamaño de división de Spark se basa en el formato de los datos y en el entorno de ejecución, pero esta es una aproximación inicial razonable. Algunos objetos se comprimen con formatos de compresión que no se pueden dividir, como gzip, por lo que Spark no puede dividirlos.

El `NumPartitions` valor puede variar en función del formato de datos, la compresión, la AWS Glue versión, el número de AWS Glue trabajadores y la configuración de Spark.

Por ejemplo, cuando cargas un único `csv.gz` objeto de 10 GB con un Spark DataFrame, el controlador de Spark solo creará una RDD partición (`NumPartitions=1`) porque gzip no se puede dividir. Esto supone una gran carga para un ejecutor de Spark en concreto y no se asigna ninguna tarea a los ejecutores restantes, como se describe en la siguiente figura.

Comprueba el número real de tareas (**`NumPartitions`**) de la etapa en la pestaña Stage de la interfaz de usuario web de Spark o ejecuta **`df.rdd.getNumPartitions()`** tu código para comprobar el paralelismo.

Cuando encuentres un archivo gzip de 10 GB, comprueba si el sistema que lo genera puede generarlo en un formato divisible. Si no es una opción, es posible que tengas que escalar la capacidad del clúster para procesar el archivo. Para ejecutar las transformaciones de manera eficiente en los datos que ha cargado, tendrá que reequilibrar los trabajadores del RDD clúster mediante la repartición.

Determine manualmente el número objetivo de particiones

En función de las propiedades de tus datos y de la implementación de determinadas funcionalidades por parte de Spark, es posible que acabes con un `NumPartitions` valor bajo, aunque el trabajo subyacente pueda seguir siendo paralelizado. Si `NumPartitions` es demasiado pequeño, `df.repartition(N)` ejecútalo para aumentar el número de particiones y poder distribuir el procesamiento entre varios ejecutores de Spark.

En este caso, la ejecución `df.repartition(100)` aumentará `NumPartitions` de 1 a 100, lo que generará 100 particiones de sus datos, cada una con una tarea que podrá asignarse a los demás ejecutores.

La operación `repartition(N)` divide todos los datos en partes iguales (10 GB/100 particiones = 100 MB/partición), lo que evita que los datos se desvíen hacia determinadas particiones.

> **Note**
>
> Cuando se ejecuta una operación de mezcla, como la que `join` se ejecuta, el número de particiones aumenta o disminuye de forma dinámica en función del valor de o. `spark.sql.shuffle.partitions` `spark.default.parallelism` Esto facilita un intercambio de datos más eficiente entre los ejecutores de Spark. Para obtener más información, consulta la documentación de Spark.

Su objetivo al determinar el número objetivo de particiones es maximizar el uso de los AWS Glue trabajadores aprovisionados. El número de AWS Glue trabajadores y el número de tareas de Spark están relacionados con el número devCPUs. Spark admite una tarea por cada CPU núcleo v. En AWS Glue la versión 3.0 o posterior, puedes calcular el número objetivo de particiones mediante la siguiente fórmula.

```
# 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
```

En este ejemplo, cada elemento de trabajo de G.1X proporciona cuatro CPU núcleos v a un ejecutor de Spark ()`spark.executor.cores = 4`. Spark admite una tarea para cada CPU núcleo v, por lo que los ejecutores de G.1X Spark pueden ejecutar cuatro tareas simultáneamente (). `numSlotPerExecutor` Este número de particiones aprovecha al máximo el clúster si las tareas tardan el mismo tiempo. Sin embargo, algunas tareas tardarán más que otras y crearán núcleos inactivos. Si esto ocurre, considere la posibilidad de multiplicar `numPartitions` por 2 o 3 para dividir y programar de manera eficiente las tareas más complicadas.

Demasiadas particiones

Un número excesivo de particiones crea un número excesivo de tareas. Esto provoca una gran carga en el controlador de Spark debido a la sobrecarga relacionada con el procesamiento distribuido, como las tareas de administración y el intercambio de datos entre los ejecutores de Spark.

Si el número de particiones de su trabajo es considerablemente mayor que el número objetivo de particiones, considere reducir el número de particiones. Puede reducir las particiones mediante las siguientes opciones:

- Si el tamaño de los archivos es muy pequeño, utilice AWS Glue groupFiles. Puede reducir el paralelismo excesivo que resulta del lanzamiento de una tarea de Apache Spark para procesar cada archivo.
- Se usa `coalesce(N)` para unir particiones. Se trata de un proceso de bajo coste. A la hora de reducir el número de particiones, `coalesce(N)` se prefiere en lugar de `repartition(N)` hacerlo, ya que `repartition(N)` realiza una reproducción aleatoria para distribuir equitativamente la cantidad de registros de cada partición. Esto aumenta los costos y la sobrecarga de administración.
- Utilice Spark 3.x Adaptive Query Execution. Como se explica en la sección Temas clave de Apache Spark, Adaptive Query Execution proporciona una función para unir automáticamente el número de particiones. Puedes usar este enfoque cuando no sepas el número de particiones hasta que realices la ejecución.

Paraleliza la carga de datos desde JDBC

El número de RDD particiones de Spark viene determinado por la configuración. Tenga en cuenta que, de forma predeterminada, solo se ejecuta una tarea para escanear un conjunto de datos fuente completo mediante una `SELECT` consulta.

AWS Glue DynamicFrames Tanto Spark como Spark DataFrames admiten la carga de JDBC datos paralelizada en varias tareas. Esto se hace mediante el uso de `where` predicados para dividir una `SELECT` consulta en varias consultas. Para paralelizar las lecturas desdeJDBC, configure las siguientes opciones:

- Para AWS Glue DynamicFrame, defina `hashfield` (o) y. `hashexpression)` `hashpartition` Para obtener más información, consulte Lectura de JDBC tablas en 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, establece `numPartitionspartitionColumn`,`lowerBound`, y`upperBound`. Para obtener más información, consulte JDBCA otras bases de datos.

```
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/")
```

Paralelice la carga de datos de DynamoDB al utilizar el conector ETL

El número de RDD particiones de Spark viene determinado por el parámetro. `dynamodb.splits` Para paralelizar las lecturas de Amazon DynamoDB, configure las siguientes opciones:

- Aumente el valor de. `dynamodb.splits`
- Optimice el parámetro siguiendo la fórmula explicada en Tipos y opciones de conexión AWS Glue para Spark. ETL

Paralelice la carga de datos de Kinesis Data Streams

El número de RDD particiones de Spark viene determinado por el número de particiones de la transmisión de datos de Amazon Kinesis Data Streams de origen. Si solo tiene unos pocos fragmentos en su transmisión de datos, solo habrá unas pocas tareas de Spark. Esto puede provocar un bajo paralelismo en los procesos posteriores. Para paralelizar las lecturas de Kinesis Data Streams, configure las siguientes opciones:

- Aumente el número de fragmentos para obtener más paralelismo al cargar datos de Kinesis Data Streams.
- Si la lógica del microlote es lo suficientemente compleja, considere la posibilidad de volver a particionar los datos al principio del lote, después de eliminar las columnas innecesarias.

Para obtener más información, consulte Prácticas recomendadas para optimizar el coste y el rendimiento de los trabajos de streaming. AWS Glue ETL

Paraleliza las tareas después de cargar los datos

Para paralelizar las tareas tras la carga de datos, aumente el número de RDD particiones mediante las siguientes opciones:

- Reparticione los datos para generar un mayor número de particiones, especialmente justo después de la carga inicial si la carga en sí no se pudo paralelizar.

Llame a `repartition()` DynamicFrame o especifique el DataFrame número de particiones. Una buena regla general es multiplicar por dos o tres veces el número de núcleos disponibles.

Sin embargo, al escribir una tabla particionada, esto puede provocar una explosión de archivos (cada partición puede generar un archivo en cada partición de la tabla). Para evitarlo, puedes reparticionar tu columna DataFrame por columnas. Esto utiliza las columnas de partición de la tabla para que los datos estén organizados antes de escribirlos. Puede especificar un número mayor de particiones sin tener archivos pequeños en las particiones de la tabla. Sin embargo, tenga cuidado de evitar la distorsión de los datos, ya que algunos valores de partición acaban quedándose con la mayoría de los datos y retrasando la finalización de la tarea.

- Cuando haya combinaciones, aumente el valor. `spark.sql.shuffle.partitions` Esto también puede ayudar a solucionar cualquier problema de memoria durante la reproducción aleatoria.

  Cuando tienes más de 2.001 particiones de modo aleatorio, Spark utiliza un formato de memoria comprimido. Si tienes un número cercano a ese valor, quizás quieras establecer el `spark.sql.shuffle.paritions` valor por encima de ese límite para obtener una representación más eficiente.

Optimice las combinaciones

Algunas operaciones, como `join()` y`groupByKey()`, requieren que Spark realice una mezcla aleatoria. La mezcla es el mecanismo de Spark para redistribuir los datos de forma que se agrupen de forma diferente en las particiones. RDD La reorganización puede ayudar a solucionar los cuellos de botella en el rendimiento. Sin embargo, dado que la mezcla suele implicar la copia de datos entre los ejecutores de Spark, la mezcla es una operación compleja y costosa. Por ejemplo, las mezclas generan los siguientes costes:

- E/S de disco:
  - Genera una gran cantidad de archivos intermedios en el disco.
- E/S de red:
  - Necesita muchas conexiones de red (número de conexiones =`Mapper × Reducer`).
  - Como los registros se agregan a nuevas RDD particiones que podrían estar alojadas en un ejecutor de Spark diferente, una parte sustancial del conjunto de datos podría moverse entre los ejecutores de Spark a través de la red.

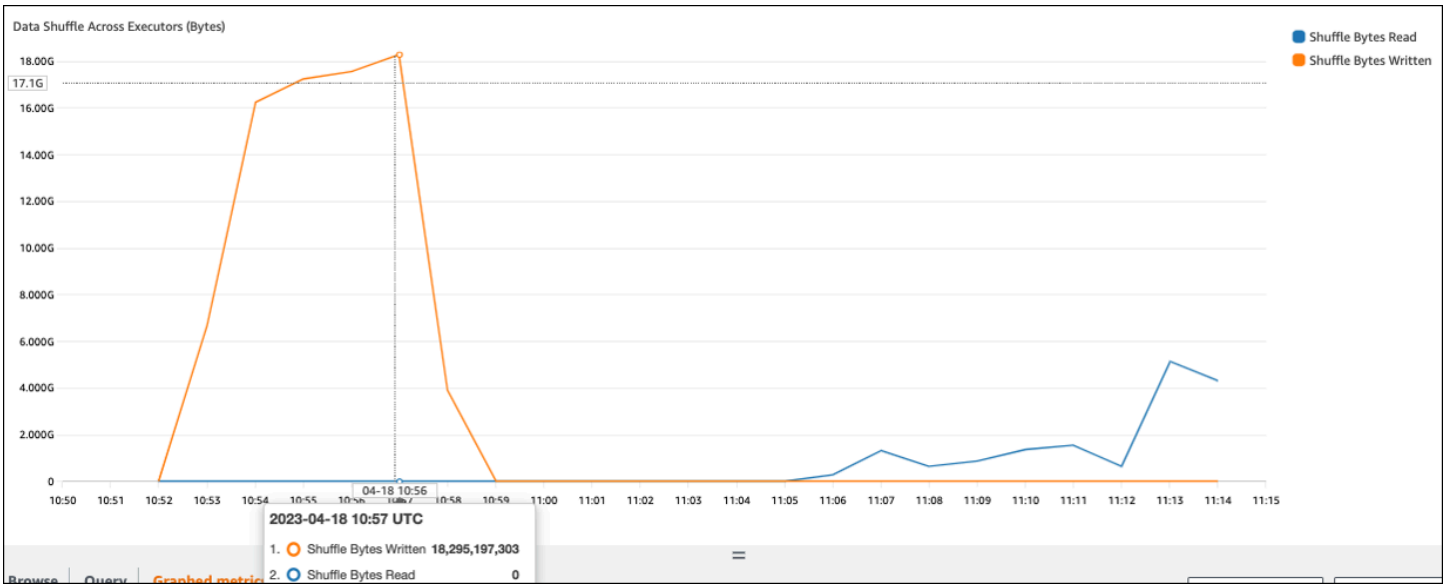
- CPUy carga de memoria:
  - Ordena los valores y fusiona conjuntos de datos. Estas operaciones se planifican en el ejecutor, lo que supone una gran carga para el ejecutor.

La reproducción aleatoria es uno de los factores más importantes en la degradación del rendimiento de tu aplicación Spark. Al almacenar los datos intermedios, puede agotar espacio en el disco local del ejecutor, lo que provoca un error en la tarea de Spark.

Puedes evaluar tu rendimiento de shuffle en CloudWatch las métricas y en la interfaz de usuario de Spark.

CloudWatch métricas

Si el valor de Shuffle Bytes Written es alto en comparación con Shuffle Bytes Read, tu trabajo de Spark podría utilizar operaciones de mezcla aleatoria como o. `join()` `groupByKey()`

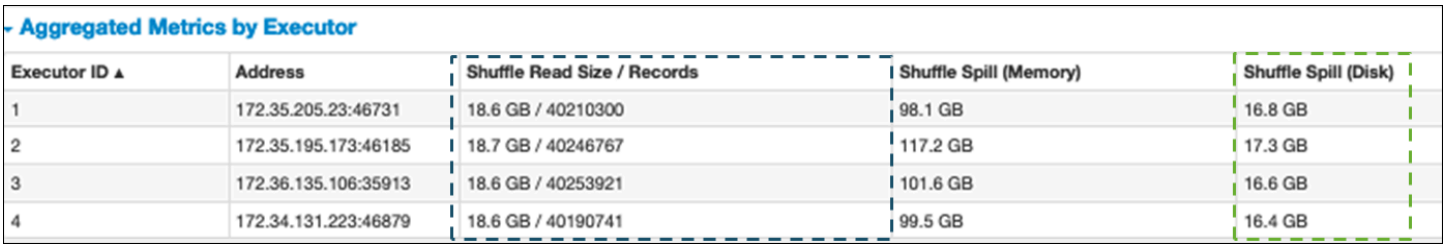


Interfaz de usuario de Spark

En la pestaña Stage de la interfaz de usuario de Spark, puedes comprobar los valores de tamaño de lectura aleatoria y de registros. También puedes verla en la pestaña Ejecutores.

En la siguiente captura de pantalla, cada ejecutor intercambia aproximadamente 18,6 GB/4020000 registros con el proceso de aleatorización, lo que supone un tamaño total de lectura aleatoria de unos 75 GB (aproximadamente).

La columna Shuffle Spill (Disco) muestra una gran cantidad de datos que se derrama en la memoria del disco, lo que puede provocar un disco lleno o un problema de rendimiento.



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

Si observas estos síntomas y la etapa tarda demasiado en comparación con tus objetivos de rendimiento, o si no funciona o presenta `No space left on device` errores, considera las siguientes soluciones. `Out Of Memory`

Optimice la unión

La `join()` operación, que une tablas, es la operación de barajado más utilizada, pero suele suponer un obstáculo en el rendimiento. Como la unión es una operación costosa, le recomendamos no utilizarla a menos que sea esencial para los requisitos de su empresa. Haga las siguientes preguntas para comprobar que está haciendo un uso eficiente de su canalización de datos:

- ¿Está volviendo a calcular una unión que también se realiza en otras tareas que puede reutilizar?
- ¿Va a unirse para resolver las claves ajenas a los valores que no utilizan los consumidores de su producción?

Tras confirmar que las operaciones de unión son esenciales para los requisitos de su empresa, consulte las siguientes opciones para optimizar la unión de forma que se ajuste a sus requisitos.

Presiona hacia abajo antes de unirte

Filtre las filas y columnas innecesarias DataFrame antes de realizar una unión. Esto tiene las siguientes ventajas:

- Reduce la cantidad de transferencia de datos durante la reproducción aleatoria
- Reduce la cantidad de procesamiento en el ejecutor Spark
- Reduce la cantidad de datos escaneados

```
# 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"])
```

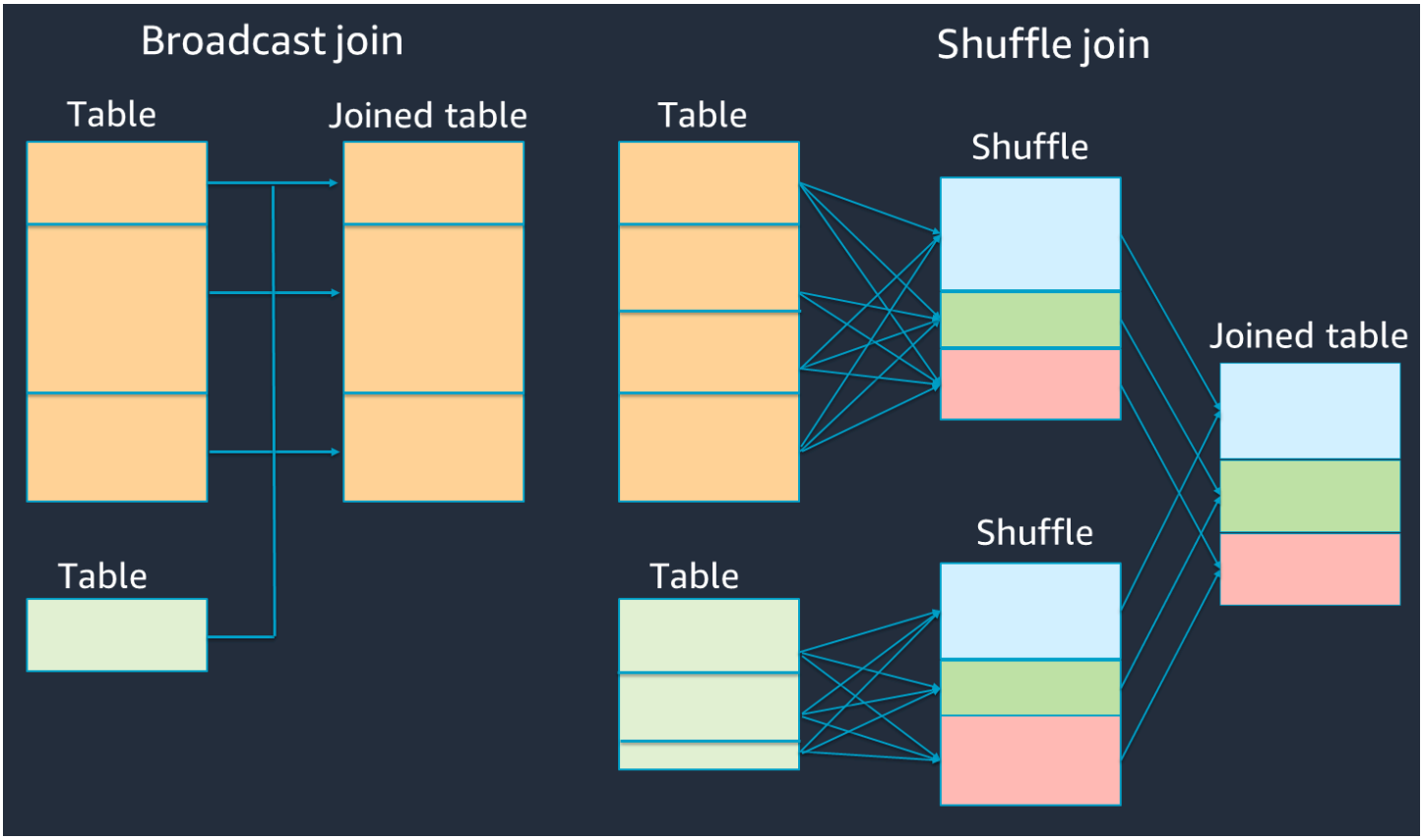
Utilice DataFrame Join

Intenta usar un Spark de alto nivelSQL, API como Spark y Datasets DataFrame, en lugar de usar RDD API o DynamicFrame join. Puedes convertir DynamicFrame a DataFrame con una llamada a un método comodyf.toDF(). Como se explica en la sección Temas clave de Apache Spark, estas operaciones de unión aprovechan internamente la optimización de consultas realizada por el optimizador de Catalyst.

Mezcla y difunde combinaciones de hash y sugerencias

Spark admite dos tipos de unión: shuffle join y broadcast hash join. Una unión hash de difusión no requiere una combinación aleatoria y puede requerir menos procesamiento que una combinación aleatoria. Sin embargo, solo se aplica al unir una mesa pequeña a una mesa grande. Al unirte a una tabla que quepa en la memoria de un solo ejecutor de Spark, considera usar una combinación hash broadcast.

El siguiente diagrama muestra la estructura de alto nivel y los pasos de una combinación hash de difusión y una combinación aleatoria.



Los detalles de cada combinación son los siguientes:

- Combinación aleatoria:
  - La combinación de hash aleatorio une dos tablas sin ordenar y distribuye la unión entre las dos tablas. Es adecuada para uniones de tablas pequeñas que se pueden almacenar en la memoria del ejecutor de Spark.
  - La combinación ordenar-combinar distribuye las dos tablas que se van a unir por clave y las ordena antes de unirlas. Es adecuada para unir tablas grandes.
- Combinación de hash de transmisión:
  - Una combinación de hash de transmisión empuja la tabla RDD o más pequeña a cada uno de los nodos de trabajo. Luego, combina el lado del mapa con cada partición de la tabla o más grandeRDD.

    Es adecuado para las uniones cuando una de tus RDDs tablas cabe en la memoria o se puede hacer que quepa en la memoria. Siempre que sea posible, es recomendable hacer una combinación hash de transmisión, ya que no es necesario mezclarla. Puedes usar una sugerencia para unirte a una transmisión desde Spark de la siguiente manera.

```
# 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 obtener más información sobre las sugerencias para unirse, consulta las sugerencias para unirse.

En la AWS Glue versión 3.0 y versiones posteriores, puede aprovechar las uniones hash emitidas automáticamente activando la ejecución adaptativa de consultas y otros parámetros. Adaptive Query Execution convierte una combinación de clasificación y fusión en una combinación de hash de transmisión cuando las estadísticas de tiempo de ejecución de cualquiera de los lados de la unión son inferiores al umbral de unión por hash de transmisión adaptable.

En la AWS Glue versión 3.0, se puede activar la ejecución adaptativa de consultas mediante la configuración. `spark.sql.adaptive.enabled=true` La ejecución adaptativa de consultas está habilitada de forma predeterminada en AWS Glue 4.0.

Puedes configurar parámetros adicionales relacionados con las combinaciones aleatorias y las uniones hash emitidas:

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

Para obtener más información sobre los parámetros relacionados, consulte Convertir una unión de clasificación y fusión en una unión de transmisión.

En la AWS Glue versión 3.0 o versiones posteriores, puedes usar otras sugerencias de unión para mezclar y así ajustar tu comportamiento.

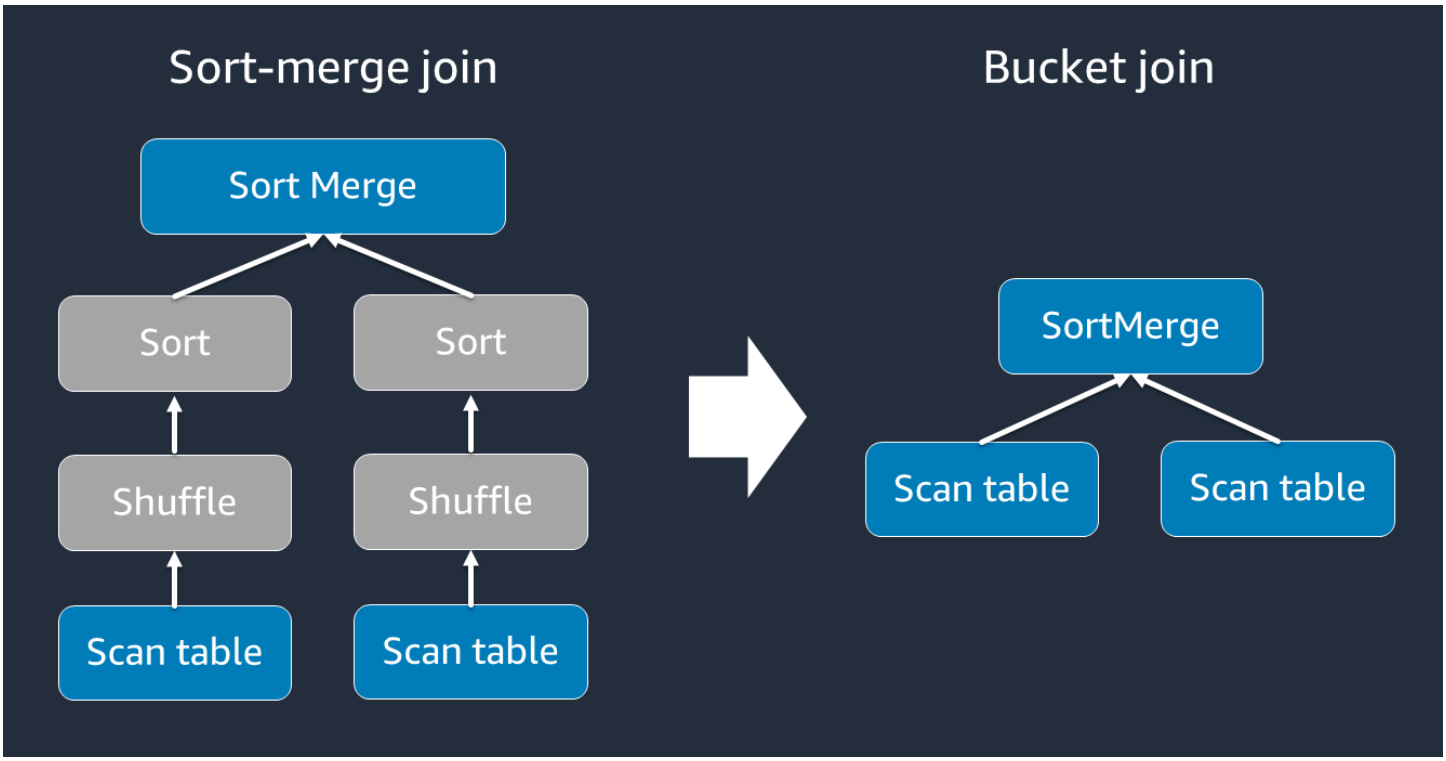
```
-- Join Hints for shuffle sort merge join
SELECT /*+ SHUFFLE_MERGE(t1) /  FROM t1 INNER JOIN t2 ON t1.key = t2.key;
SELECT /*+ MERGEJOIN(t2) /  FROM t1 INNER JOIN t2 ON t1.key = t2.key;
SELECT /*+ MERGE(t1) /  FROM t1 INNER JOIN t2 ON t1.key = t2.key;
```

```
-- 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;
```

Usa el agrupamiento

La combinación ordenar-fusionar requiere dos fases: barajar y ordenar y, a continuación, fusionar. Estas dos fases pueden sobrecargar el ejecutor de Spark OOM y provocar problemas de rendimiento cuando algunos ejecutores se fusionan y otros se clasifican simultáneamente. En esos casos, podría ser posible realizar una unión eficiente mediante agrupamiento. La agrupación premezclará y ordenará previamente las entradas en las teclas de combinación y, a continuación, escribirá los datos ordenados en una tabla intermedia. El coste de los pasos de barajar y ordenar se puede reducir al unir tablas grandes definiendo las tablas intermedias ordenadas con antelación.



Las tablas agrupadas son útiles para lo siguiente:

- Los datos se unían con frecuencia a través de la misma clave, como `account_id`
- Cargar tablas acumulativas diarias, como las tablas base y delta, que podrían agruparse en una columna común

Puede crear una tabla agrupada mediante el siguiente código.

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

Reparticione DataFrames las claves de unión antes de la unión

Para reparticionar los dos elementos DataFrames de las claves de combinación antes de la unión, utilice las siguientes instrucciones.

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

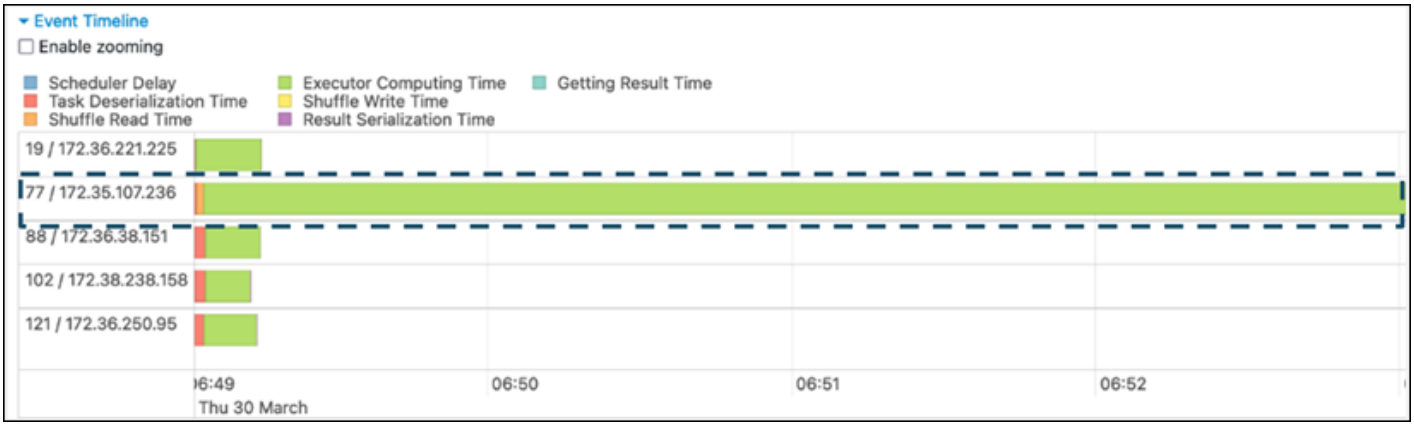
Esto dividirá dos (aún separados) de RDDs la clave de unión antes de iniciar la unión. Si las dos RDDs están particionadas en la misma clave y con el mismo código de partición, es muy probable que los RDD registros que planea unir estén ubicados en el mismo elemento de trabajo antes de volver a barajarlos para unirlos. Esto podría mejorar el rendimiento al reducir la actividad de la red y el sesgo de datos durante la unión.

Supere el sesgo de datos

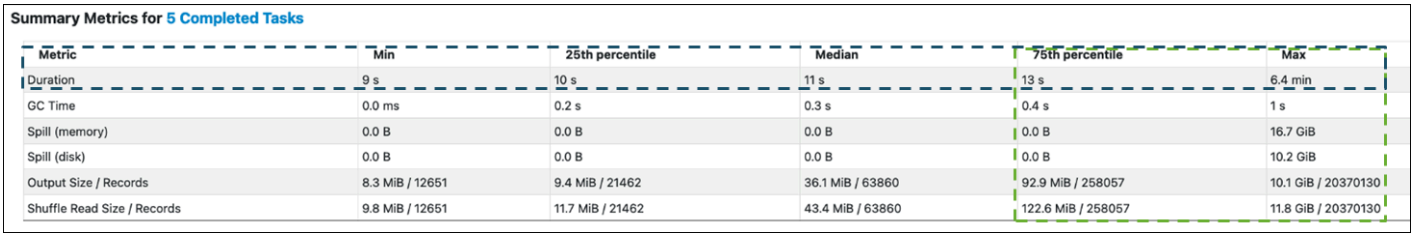
La asimetría de los datos es una de las causas más comunes de un cuello de botella en los trabajos de Spark. Se produce cuando los datos no se distribuyen uniformemente entre las particiones. RDD Esto hace que las tareas de esa partición tarden mucho más que en otras, lo que retrasa el tiempo total de procesamiento de la aplicación.

Para identificar el sesgo de los datos, evalúa las siguientes métricas en la interfaz de usuario de Spark:

- En la pestaña Escenario de la interfaz de usuario de Spark, examina la página de cronología del evento. Puedes ver una distribución desigual de las tareas en la siguiente captura de pantalla. Las tareas que se distribuyen de forma desigual o que tardan demasiado en ejecutarse pueden indicar un sesgo en los datos.



- Otra página importante es Summary Metrics, que muestra las estadísticas de las tareas de Spark. La siguiente captura de pantalla muestra las métricas con los percentiles de duración, tiempo de GC, pérdida (memoria), pérdida (disco), etc.



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

  Cuando las tareas estén distribuidas uniformemente, verás números similares en todos los percentiles. Cuando los datos estén sesgados, verá valores muy sesgados en cada percentil. En el ejemplo, la duración de la tarea es inferior a 13 segundos en el percentil mínimo, el percentil 25, la mediana y el percentil 75. Si bien la tarea Max procesó 100 veces más datos que el percentil 75, su duración de 6,4 minutos es aproximadamente 30 veces mayor. Esto significa que al menos una tarea (o hasta un 25 por ciento de las tareas) llevó mucho más tiempo que el resto de las tareas.

Si ves que los datos están sesgados, prueba lo siguiente:

- Si usa AWS Glue 3.0, habilite la ejecución adaptativa de consultas mediante la configuración`spark.sql.adaptive.enabled=true`. La ejecución adaptativa de consultas está habilitada de forma predeterminada en la AWS Glue versión 4.0.

  También puede utilizar la ejecución de consultas adaptativa para el sesgo de datos introducido por las uniones configurando los siguientes parámetros 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 obtener más información, consulte la documentación de Apache Spark.

- Utilice claves con un amplio rango de valores para las claves de unión. En una combinación aleatoria, las particiones se determinan para cada valor hash de una clave. Si la cardinalidad de una clave de unión es demasiado baja, es más probable que la función hash distribuya mal los datos entre las particiones. Por lo tanto, si su aplicación y su lógica empresarial lo admiten, considere la posibilidad de utilizar una clave de cardinalidad más alta o una clave compuesta.

```
# 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"])
```

Utilice la memoria caché

Cuando utilices el método repetitivo DataFrames, evita tener que `df.persist()` hacer cálculos o barajarlos de forma adicional utilizando `df.cache()` o almacenando en caché los resultados del cálculo en la memoria de cada ejecutor de Spark y en el disco. Spark también admite la persistencia RDDs en el disco o la replicación en varios nodos (nivel de almacenamiento).

Por ejemplo, puedes conservarlos DataFrames añadiendo. `df.persist()` Cuando la memoria caché ya no sea necesaria, puede utilizarla `unpersist` para descartar los datos almacenados en la memoria caché.

```
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()
```

Elimina las acciones innecesarias de Spark

Evita ejecutar acciones innecesarias como `countshow`, `ocollect`. Como se explica en la sección Temas clave de Apache Spark, Spark es perezoso. Cada transformación se RDD puede volver a calcular cada vez que ejecutes una acción en ella. Cuando utilizas muchas acciones de Spark, se recurre a varios accesos a fuentes, se realizan cálculos de tareas y se ejecuta aleatoriamente cada acción.

Si no necesitas ninguna `collect()` otra acción en tu entorno comercial, considera eliminarlas.

> **Note**
>
> Evita usar Spark `collect()` en entornos comerciales en la medida de lo posible. La `collect()` acción devuelve todos los resultados de un cálculo realizado en el ejecutor de Spark al controlador de Spark, lo que podría provocar que el controlador de Spark devuelva un OOM error. Para evitar OOM errores, Spark lo establece de forma `spark.driver.maxResultSize = 1GB` predeterminada, lo que limita el tamaño máximo de los datos devueltos al controlador de Spark a 1 GB.

Minimice los gastos de planificación

Como se discutió sobre los temas clave de Apache Spark, el controlador Spark genera el plan de ejecución. Según ese plan, las tareas se asignan al ejecutor de Spark para su procesamiento distribuido. Sin embargo, el controlador Spark puede convertirse en un cuello de botella si hay una gran cantidad de archivos pequeños o si AWS Glue Data Catalog contiene una gran cantidad de particiones. Para identificar los altos gastos de planificación, evalúe las siguientes métricas.

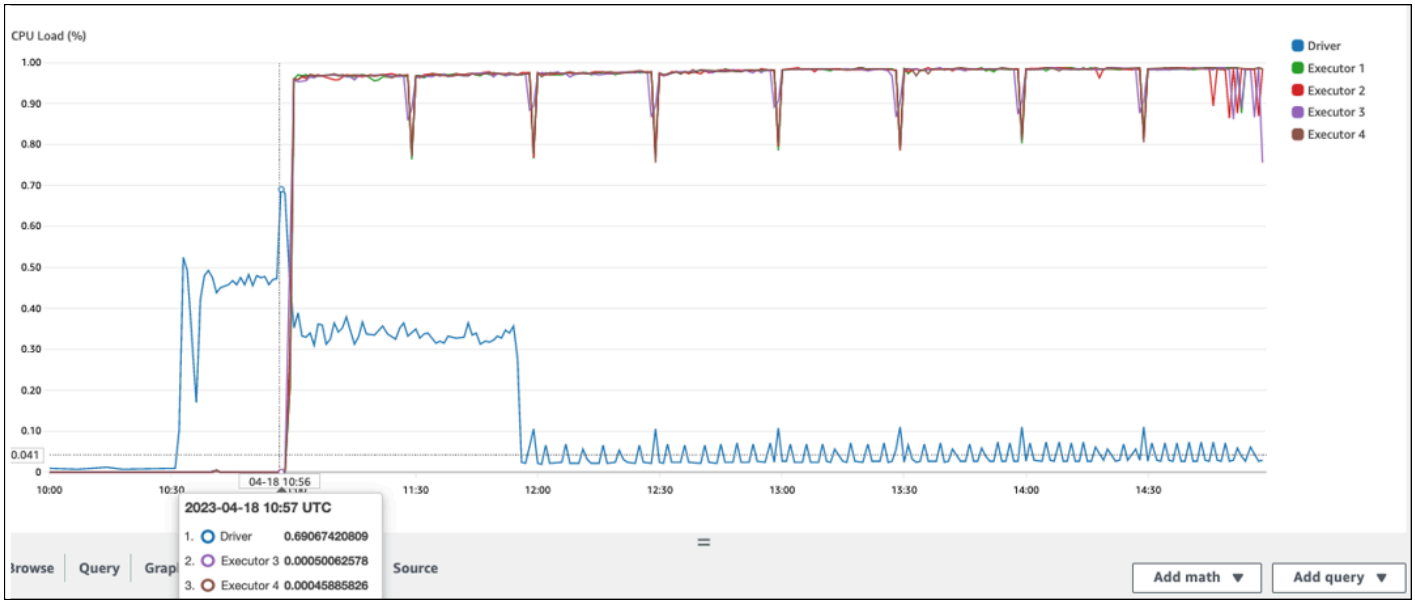
CloudWatch métricas

Compruebe CPUla carga y el uso de la memoria en las siguientes situaciones:

- La CPUcarga del controlador Spark y el uso de la memoria se registran como altos. Normalmente, el controlador Spark no procesa los datos, por lo que el uso de la CPU carga y la memoria no aumentan. Sin embargo, si la fuente de datos de Amazon S3 tiene demasiados archivos

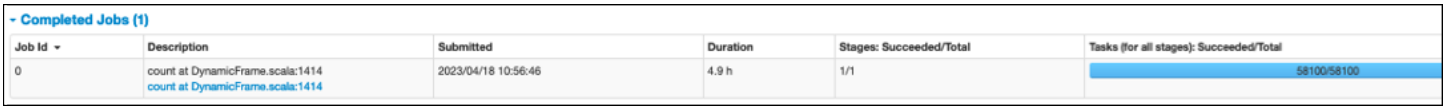
pequeños, la enumeración de todos los objetos de S3 y la gestión de un gran número de tareas puede provocar un uso elevado de los recursos.

- Hay un largo intervalo antes de que comience el procesamiento en el ejecutor de Spark. En la siguiente captura de pantalla de ejemplo, la CPU carga del ejecutor de Spark es demasiado baja hasta las 10:57, a pesar de que el trabajo comenzó a las AWS Glue 10:00. Esto indica que es posible que el controlador de Spark esté tardando mucho en generar un plan de ejecución. En este ejemplo, recuperar la gran cantidad de particiones del catálogo de datos y enumerar la gran cantidad de archivos pequeños en el controlador de Spark lleva mucho tiempo.



Interfaz de usuario de Spark

En la pestaña Job de la interfaz de usuario de Spark, puedes ver la hora enviada. En el siguiente ejemplo, el controlador de Spark inició el job0 a las 10:56:46, aunque el trabajo comenzó a las 10:00:00. AWS Glue



También puede ver las tareas (para todas las etapas): Finalizadas/Tiempo total en la pestaña Trabajo. En este caso, el número de tareas se registra como. `58100` Como se explica en la sección Amazon S3 de la página Paralelizar tareas, la cantidad de tareas corresponde aproximadamente a la cantidad de objetos de S3. Esto significa que hay unos 58.100 objetos en Amazon S3.

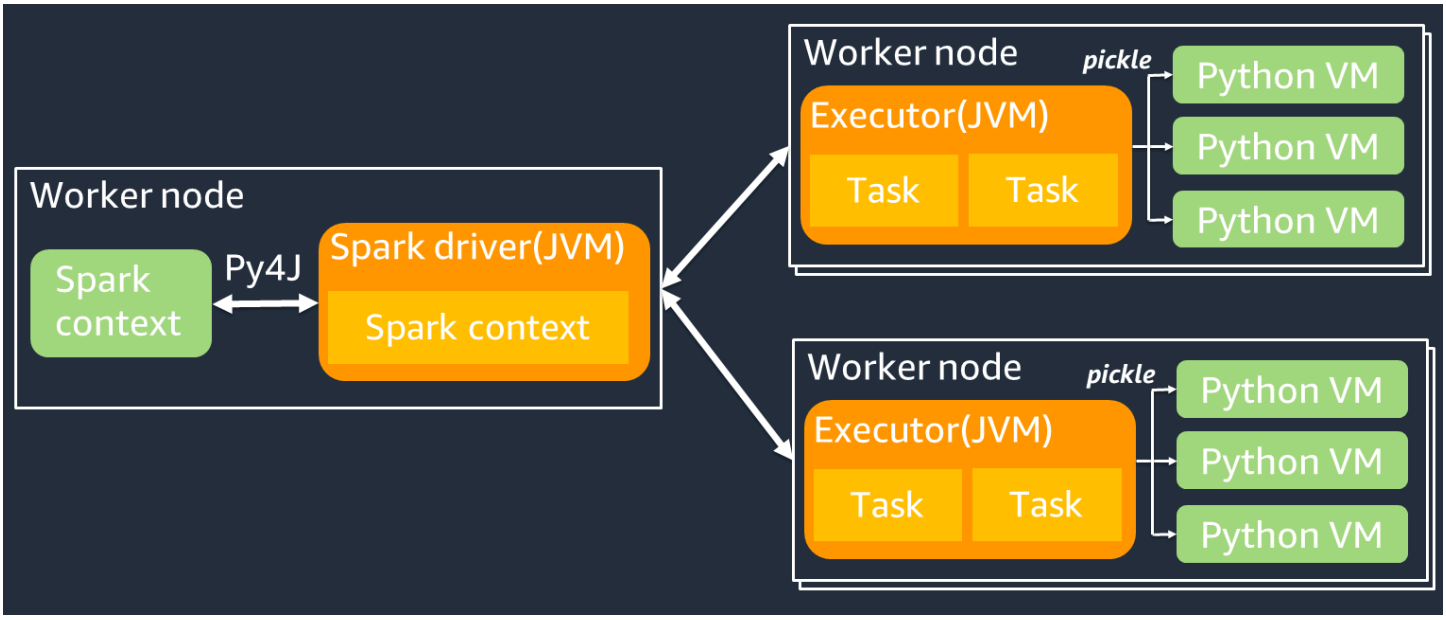
Para obtener más información sobre este trabajo y su cronograma, consulte la pestaña Etapa. Si observas un cuello de botella en el controlador Spark, considera las siguientes soluciones:

- Si Amazon S3 tiene demasiados archivos, consulte las instrucciones sobre el paralelismo excesivo de la sección Demasiadas particiones de la página Paralelizar tareas.
- Si Amazon S3 tiene demasiadas particiones, consulte las instrucciones sobre el uso excesivo de particiones en la sección Demasiadas particiones de Amazon S3 de la página Reducir la cantidad de datos escaneados. Habilite los índices de AWS Glue particiones si hay muchas particiones para reducir la latencia a la hora de recuperar los metadatos de las particiones del catálogo de datos. Para obtener más información, consulte Mejorar el rendimiento de las consultas mediante índices de AWS Glue partición.
- Si JDBC tiene demasiadas particiones, reduzca el `hashpartition` valor.
- Si DynamoDB tiene demasiadas particiones, reduzca el valor. `dynamodb.splits`
- Si los trabajos de streaming tienen demasiadas particiones, reduzca el número de particiones.

Optimice las funciones definidas por el usuario

Las funciones definidas por el usuario (UDFs) y, a PySpark menudo, reducen el rendimiento de `RDD.map` manera significativa. Esto se debe a la sobrecarga necesaria para representar con precisión tu código Python en la implementación de Scala subyacente de Spark.

El siguiente diagrama muestra la arquitectura de los PySpark trabajos. Cuando lo usas PySpark, el controlador Spark usa la biblioteca Py4j para llamar a los métodos de Java desde Python. Al llamar a Spark SQL o a funciones DataFrame integradas, hay poca diferencia de rendimiento entre Python y Scala porque las funciones se ejecutan en cada ejecutor JVM mediante un plan de ejecución optimizado.



Si usa su propia lógica de Python, como usar`map/ mapPartitions/ udf`, la tarea se ejecutará en un entorno de ejecución de Python. La administración de dos entornos genera un costo general. Además, los datos de la memoria deben transformarse para que puedan utilizarlos las funciones integradas del entorno de JVM ejecución. Pickle es un formato de serialización que se utiliza de forma predeterminada para el intercambio entre los tiempos de ejecución y JVM Python. Sin embargo, el coste de esta serialización y deserialización es muy elevado, por lo que UDFs escribir en Java o Scala es más rápido que en Python. UDFs

Para evitar la sobrecarga de serialización y deserialización, tenga en cuenta lo siguiente: PySpark

- Usa las SQL funciones de Spark integradas: considera reemplazar una función propia UDF o de mapa por una función de Spark SQL o DataFrame funciones integradas. Al ejecutar Spark SQL o funciones DataFrame integradas, hay poca diferencia de rendimiento entre Python y Scala porque las tareas se gestionan en el ejecutor de cada ejecutor. JVM
- Implemente UDFs en Scala o Java: considere usar uno UDF que esté escrito en Java o Scala, ya que se ejecutan en. JVM
- Use Apache basado en Arrow UDFs para cargas de trabajo vectorizadas: considere usar el basado en Arrow. UDFs Esta función también se conoce como vectorizada (Pandas). UDF UDF Apache Arrow es un formato de datos en memoria independiente del lenguaje que se AWS Glue puede utilizar para transferir datos de manera eficiente entre procesos de Python. JVM Actualmente, esto es más beneficioso para los usuarios de Python que trabajan con Pandas o NumPy datos.

Arrow es un formato columnar (vectorizado). Su uso no es automático y puede requerir algunos cambios menores en la configuración o el código para aprovechar al máximo y garantizar la compatibilidad. Para obtener más detalles y conocer las limitaciones, consulte Apache Arrow en PySpark.

El siguiente ejemplo compara un incremental básico UDF en Python estándar, como vectorizado UDF y en Spark. SQL

Python estándar UDF

El tiempo de ejemplo es 3,20 (segundos).

Código de ejemplo

```
# 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()
```

Plan de ejecución

```
== 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)
```

Vectorizado UDF

El tiempo de ejemplo es 0.59 (seg).

El vectorizado UDF es 5 veces más rápido que el ejemplo anterior. UDF Al comprobarlo`Physical Plan`, puede ver `ArrowEvalPython` que Apache Arrow vectoriza esta aplicación. Para habilitar VectorizedUDF, debe `spark.sql.execution.arrow.pyspark.enabled = true` especificarlo en su código.

Código de ejemplo

```
# 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()
```

Plan de ejecución

```
== 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

El tiempo de ejemplo es 0.087 (seg).

Spark SQL es mucho más rápido que VectorizedUDF, porque las tareas se ejecutan en cada ejecutor sin JVM un tiempo de ejecución de Python. Si puedes sustituir la tuya UDF por una función integrada, te recomendamos que lo hagas.

Código de ejemplo

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

Uso de pandas para macrodatos

Si ya estás familiarizado con los pandas y quieres usar Spark para el big data, puedes usar los pandas API de Spark. AWS Glue La versión 4.0 y las versiones posteriores lo admiten. Para empezar, puedes utilizar el Quickstart oficial del portátil: Pandas API on Spark. Para obtener más información, consulta la PySpark documentación.

Recursos

- AWS Glue
- Ajuste del rendimiento (guía SQL de Spark)
- AWS Glue Taller de optimización

Historial de documentos

En la siguiente tabla, se describen cambios significativos de esta guía. Si quiere recibir notificaciones de futuras actualizaciones, puede suscribirse a las notificaciones RSS.

Cambio	Descripción	Fecha
Publicación inicial	—	2 de enero de 2024

AWS Glosario de orientación prescriptiva

Los siguientes son términos de uso común en las estrategias, guías y patrones proporcionados por AWS Prescriptive Guidance. Para sugerir entradas, utilice el enlace Enviar comentarios al final del glosario.

Números

Las 7 R

Siete estrategias de migración comunes para trasladar aplicaciones a la nube. Estas estrategias se basan en las 5 R que Gartner identificó en 2011 y consisten en lo siguiente:

- Refactorizar/rediseñar: traslade una aplicación y modifique su arquitectura mediante el máximo aprovechamiento de las características nativas en la nube para mejorar la agilidad, el rendimiento y la escalabilidad. Por lo general, esto implica trasladar el sistema operativo y la base de datos. Ejemplo: migre su base de datos Oracle local a la edición compatible con PostgreSQL de Amazon Aurora.
- Redefinir la plataforma (transportar y redefinir): traslade una aplicación a la nube e introduzca algún nivel de optimización para aprovechar las capacidades de la nube. Ejemplo: migre su base de datos Oracle local a Amazon Relational Database Service (Amazon RDS) para Oracle en el. Nube de AWS
- Recomprar (readquirir): cambie a un producto diferente, lo cual se suele llevar a cabo al pasar de una licencia tradicional a un modelo SaaS. Ejemplo: migre su sistema de gestión de relaciones con los clientes (CRM) a Salesforce.com.
- Volver a alojar (migrar mediante lift-and-shift): traslade una aplicación a la nube sin realizar cambios para aprovechar las capacidades de la nube. Ejemplo: migre su base de datos Oracle local a Oracle en una instancia EC2 del. Nube de AWS
- Reubicar: (migrar el hipervisor mediante lift and shift): traslade la infraestructura a la nube sin comprar equipo nuevo, reescribir aplicaciones o modificar las operaciones actuales. Los servidores se migran de una plataforma local a un servicio en la nube para la misma plataforma. Ejemplo: migrar una Microsoft Hyper-V aplicación a AWS.
- Retener (revisitar): conserve las aplicaciones en el entorno de origen. Estas pueden incluir las aplicaciones que requieren una refactorización importante, que desee posponer para más adelante, y las aplicaciones heredadas que desee retener, ya que no hay ninguna justificación empresarial para migrarlas.

- Retirar: retire o elimine las aplicaciones que ya no sean necesarias en un entorno de origen.

A

ABAC

Consulte control de acceso basado en atributos.

servicios abstractos

Consulte servicios gestionados.

ACID

Consulte atomicidad, consistencia, aislamiento y durabilidad.

migración activa-activa

Método de migración de bases de datos en el que las bases de datos de origen y destino se mantienen sincronizadas (mediante una herramienta de replicación bidireccional o mediante operaciones de escritura doble) y ambas bases de datos gestionan las transacciones de las aplicaciones conectadas durante la migración. Este método permite la migración en lotes pequeños y controlados, en lugar de requerir una transición única. Es más flexible, pero requiere más trabajo que la migración activa-pasiva.

migración activa-pasiva

Método de migración de bases de datos en el que las bases de datos de origen y destino se mantienen sincronizadas, pero solo la base de datos de origen gestiona las transacciones de las aplicaciones conectadas, mientras los datos se replican en la base de datos de destino. La base de datos de destino no acepta ninguna transacción durante la migración.

función agregada

Función SQL que opera en un grupo de filas y calcula un único valor de retorno para el grupo. Entre los ejemplos de funciones agregadas se incluyen `SUM` y `MAX`.

IA

Véase inteligencia artificial.

AIOps

Consulte las operaciones de inteligencia artificial.

anonimización

El proceso de eliminar permanentemente la información personal de un conjunto de datos. La anonimización puede ayudar a proteger la privacidad personal. Los datos anonimizados ya no se consideran datos personales.

antipatrones

Una solución que se utiliza con frecuencia para un problema recurrente en el que la solución es contraproducente, ineficaz o menos eficaz que una alternativa.

control de aplicaciones

Un enfoque de seguridad que permite el uso únicamente de aplicaciones aprobadas para ayudar a proteger un sistema contra el malware.

cartera de aplicaciones

Recopilación de información detallada sobre cada aplicación que utiliza una organización, incluido el costo de creación y mantenimiento de la aplicación y su valor empresarial. Esta información es clave para el proceso de detección y análisis de la cartera y ayuda a identificar y priorizar las aplicaciones que se van a migrar, modernizar y optimizar.

inteligencia artificial (IA)

El campo de la informática que se dedica al uso de tecnologías informáticas para realizar funciones cognitivas que suelen estar asociadas a los seres humanos, como el aprendizaje, la resolución de problemas y el reconocimiento de patrones. Para más información, consulte ¿Qué es la inteligencia artificial?

operaciones de inteligencia artificial (AIOps)

El proceso de utilizar técnicas de machine learning para resolver problemas operativos, reducir los incidentes operativos y la intervención humana, y mejorar la calidad del servicio. Para obtener más información sobre cómo se utiliza AIOps en la estrategia de migración de AWS , consulte la Guía de integración de operaciones.

cifrado asimétrico

Algoritmo de cifrado que utiliza un par de claves, una clave pública para el cifrado y una clave privada para el descifrado. Puede compartir la clave pública porque no se utiliza para el descifrado, pero el acceso a la clave privada debe estar sumamente restringido.

atomicidad, consistencia, aislamiento, durabilidad (ACID)

Conjunto de propiedades de software que garantizan la validez de los datos y la fiabilidad operativa de una base de datos, incluso en caso de errores, cortes de energía u otros problemas.

control de acceso basado en atributos (ABAC)

La práctica de crear permisos detallados basados en los atributos del usuario, como el departamento, el puesto de trabajo y el nombre del equipo. Para obtener más información, consulte ABAC AWS en la documentación AWS Identity and Access Management (IAM).

origen de datos fidedigno

Ubicación en la que se almacena la versión principal de los datos, que se considera la fuente de información más fiable. Puede copiar los datos del origen de datos autorizado a otras ubicaciones con el fin de procesarlos o modificarlos, por ejemplo, anonimizarlos, redactarlos o seudonimizarlos.

Zona de disponibilidad

Una ubicación distinta dentro de una Región de AWS que está aislada de los fallos en otras zonas de disponibilidad y que proporciona una conectividad de red económica y de baja latencia a otras zonas de disponibilidad de la misma región.

AWS Marco de adopción de la nube (AWS CAF)

Un marco de directrices y mejores prácticas AWS para ayudar a las organizaciones a desarrollar un plan eficiente y eficaz para migrar con éxito a la nube. AWS CAF organiza la orientación en seis áreas de enfoque denominadas perspectivas: negocios, personas, gobierno, plataforma, seguridad y operaciones. Las perspectivas empresariales, humanas y de gobernanza se centran en las habilidades y los procesos empresariales; las perspectivas de plataforma, seguridad y operaciones se centran en las habilidades y los procesos técnicos. Por ejemplo, la perspectiva humana se dirige a las partes interesadas que se ocupan de los Recursos Humanos (RR. HH.), las funciones del personal y la administración de las personas. Desde esta perspectiva, AWS CAF proporciona orientación para el desarrollo, la formación y la comunicación de las personas a fin de ayudar a preparar a la organización para una adopción exitosa de la nube. Para obtener más información, consulte la Página web de AWS CAF y el Documento técnico de AWS CAF.

AWS Marco de calificación de la carga de trabajo (AWS WQF)

Herramienta que evalúa las cargas de trabajo de migración de bases de datos, recomienda estrategias de migración y proporciona estimaciones de trabajo. AWS WQF se incluye con AWS

Schema Conversion Tool ().AWS SCT Analiza los esquemas de bases de datos y los objetos de código, el código de las aplicaciones, las dependencias y las características de rendimiento y proporciona informes de evaluación.

B

Un bot malo

Un bot destinado a interrumpir o causar daño a personas u organizaciones.

BCP

Consulte la planificación de la continuidad del negocio.

gráfico de comportamiento

Una vista unificada e interactiva del comportamiento de los recursos y de las interacciones a lo largo del tiempo. Puede utilizar un gráfico de comportamiento con Amazon Detective para examinar los intentos de inicio de sesión fallidos, las llamadas sospechosas a la API y acciones similares. Para obtener más información, consulte Datos en un gráfico de comportamiento en la documentación de Detective.

sistema big-endian

Un sistema que almacena primero el byte más significativo. Véase también endianness.

clasificación binaria

Un proceso que predice un resultado binario (una de las dos clases posibles). Por ejemplo, es posible que su modelo de ML necesite predecir problemas como “¿Este correo electrónico es spam o no es spam?” o “¿Este producto es un libro o un automóvil?”.

filtro de floración

Estructura de datos probabilística y eficiente en términos de memoria que se utiliza para comprobar si un elemento es miembro de un conjunto.

implementación azul/verde

Una estrategia de despliegue en la que se crean dos entornos separados pero idénticos. La versión actual de la aplicación se ejecuta en un entorno (azul) y la nueva versión de la aplicación en el otro entorno (verde). Esta estrategia le ayuda a revertirla rápidamente con un impacto mínimo.

bot

Una aplicación de software que ejecuta tareas automatizadas a través de Internet y simula la actividad o interacción humana. Algunos bots son útiles o beneficiosos, como los rastreadores web que indexan información en Internet. Algunos otros bots, conocidos como bots malos, tienen como objetivo interrumpir o causar daños a personas u organizaciones.

botnet

Redes de bots que están infectadas por malware y que están bajo el control de una sola parte, conocida como pastor u operador de bots. Las botnets son el mecanismo más conocido para escalar los bots y su impacto.

rama

Área contenida de un repositorio de código. La primera rama que se crea en un repositorio es la rama principal. Puede crear una rama nueva a partir de una rama existente y, a continuación, desarrollar características o corregir errores en la rama nueva. Una rama que se genera para crear una característica se denomina comúnmente rama de característica. Cuando la característica se encuentra lista para su lanzamiento, se vuelve a combinar la rama de característica con la rama principal. Para obtener más información, consulte Acerca de las sucursales (GitHub documentación).

acceso con cristales rotos

En circunstancias excepcionales y mediante un proceso aprobado, un usuario puede acceder rápidamente a un sitio para el Cuenta de AWS que normalmente no tiene permisos de acceso. Para obtener más información, consulte el indicador Implemente procedimientos de rotura de cristales en la guía Well-Architected AWS .

estrategia de implementación sobre infraestructura existente

La infraestructura existente en su entorno. Al adoptar una estrategia de implementación sobre infraestructura existente para una arquitectura de sistemas, se diseña la arquitectura en función de las limitaciones de los sistemas y la infraestructura actuales. Si está ampliando la infraestructura existente, puede combinar las estrategias de implementación sobre infraestructuras existentes y de implementación desde cero.

caché de búfer

El área de memoria donde se almacenan los datos a los que se accede con más frecuencia.

capacidad empresarial

Lo que hace una empresa para generar valor (por ejemplo, ventas, servicio al cliente o marketing). Las arquitecturas de microservicios y las decisiones de desarrollo pueden estar impulsadas por las capacidades empresariales. Para obtener más información, consulte la sección Organizado en torno a las capacidades empresariales del documento técnico Ejecutar microservicios en contenedores en AWS.

planificación de la continuidad del negocio (BCP)

Plan que aborda el posible impacto de un evento disruptivo, como una migración a gran escala en las operaciones y permite a la empresa reanudar las operaciones rápidamente.

C

CAF

Consulte el marco AWS de adopción de la nube.

despliegue canario

El lanzamiento lento e incremental de una versión para los usuarios finales. Cuando se tiene confianza, se despliega la nueva versión y se reemplaza la versión actual en su totalidad.

CCoE

Consulte el Centro de excelencia en la nube.

CDC

Consulte la captura de datos de cambios.

captura de datos de cambio (CDC)

Proceso de seguimiento de los cambios en un origen de datos, como una tabla de base de datos, y registro de los metadatos relacionados con el cambio. Puede utilizar los CDC para diversos fines, como auditar o replicar los cambios en un sistema de destino para mantener la sincronización.

ingeniería del caos

Introducir intencionalmente fallos o eventos disruptivos para poner a prueba la resiliencia de un sistema. Puedes usar AWS Fault Injection Service (AWS FIS) para realizar experimentos que estresen tus AWS cargas de trabajo y evalúen su respuesta.

CI/CD

Consulte la integración continua y la entrega continua.

clasificación

Un proceso de categorización que permite generar predicciones. Los modelos de ML para problemas de clasificación predicen un valor discreto. Los valores discretos siempre son distintos entre sí. Por ejemplo, es posible que un modelo necesite evaluar si hay o no un automóvil en una imagen.

cifrado del cliente

Cifrado de datos localmente, antes de que el objetivo los servicio de AWS reciba.

Centro de excelencia en la nube (CCoE)

Equipo multidisciplinario que impulsa los esfuerzos de adopción de la nube en toda la organización, incluido el desarrollo de las prácticas recomendadas en la nube, la movilización de recursos, el establecimiento de plazos de migración y la dirección de la organización durante las transformaciones a gran escala. Para obtener más información, consulte las publicaciones de CCoE en el blog de estrategia Nube de AWS empresarial.

computación en la nube

La tecnología en la nube que se utiliza normalmente para la administración de dispositivos de IoT y el almacenamiento de datos de forma remota. La computación en la nube suele estar conectada a la tecnología de computación perimetral.

modelo operativo en la nube

En una organización de TI, el modelo operativo que se utiliza para crear, madurar y optimizar uno o más entornos de nube. Para obtener más información, consulte Creación de su modelo operativo de nube.

etapas de adopción de la nube

Las cuatro fases por las que suelen pasar las organizaciones cuando migran a Nube de AWS:

- Proyecto: ejecución de algunos proyectos relacionados con la nube con fines de prueba de concepto y aprendizaje
- Fundamento: realización de inversiones fundamentales para escalar la adopción de la nube (p. ej., crear una zona de aterrizaje, definir un CCoE, establecer un modelo de operaciones)
- Migración: migración de aplicaciones individuales

- Reinvención: optimización de productos y servicios e innovación en la nube

Stephen Orban definió estas etapas en la entrada del blog The Journey Toward Cloud-First & the Stages of Adoption del blog Nube de AWS Enterprise Strategy. Para obtener información sobre su relación con la estrategia de AWS migración, consulte la guía de preparación para la migración.

CMDB

Consulte la base de datos de administración de la configuración.

repositorio de código

Una ubicación donde el código fuente y otros activos, como documentación, muestras y scripts, se almacenan y actualizan mediante procesos de control de versiones. Los repositorios en la nube más comunes incluyen GitHub o AWS CodeCommit. Cada versión del código se denomina rama. En una estructura de microservicios, cada repositorio se encuentra dedicado a una única funcionalidad. Una sola canalización de CI/CD puede utilizar varios repositorios.

caché en frío

Una caché de búfer que está vacía no está bien poblada o contiene datos obsoletos o irrelevantes. Esto afecta al rendimiento, ya que la instancia de la base de datos debe leer desde la memoria principal o el disco, lo que es más lento que leer desde la memoria caché del búfer.

datos fríos

Datos a los que se accede con poca frecuencia y que suelen ser históricos. Al consultar este tipo de datos, normalmente se aceptan consultas lentas. Trasladar estos datos a niveles o clases de almacenamiento de menor rendimiento y menos costosos puede reducir los costos.

visión artificial (CV)

Campo de la IA que utiliza el aprendizaje automático para analizar y extraer información de formatos visuales, como imágenes y vídeos digitales. Por ejemplo, AWS Panorama ofrece dispositivos que añaden CV a las redes de cámaras locales, y Amazon SageMaker proporciona algoritmos de procesamiento de imágenes para CV.

desviación de configuración

En el caso de una carga de trabajo, un cambio de configuración con respecto al estado esperado. Puede provocar que la carga de trabajo deje de cumplir las normas y, por lo general, es gradual e involuntario.

base de datos de administración de configuración (CMDB)

Repositorio que almacena y administra información sobre una base de datos y su entorno de TI, incluidos los componentes de hardware y software y sus configuraciones. Por lo general, los datos de una CMDB se utilizan en la etapa de detección y análisis de la cartera de productos durante la migración.

paquete de conformidad

Conjunto de AWS Config reglas y medidas correctivas que puede reunir para personalizar sus comprobaciones de conformidad y seguridad. Puede implementar un paquete de conformidad como una entidad única en una región Cuenta de AWS y, o en una organización, mediante una plantilla YAML. Para obtener más información, consulta los paquetes de conformidad en la documentación. AWS Config

integración y entrega continuas (CI/CD)

El proceso de automatización de las etapas de origen, compilación, prueba, presentación y producción del proceso de lanzamiento del software. La CI/CD se describe comúnmente como una canalización. La CI/CD puede ayudarlo a automatizar los procesos, mejorar la productividad, mejorar la calidad del código y entregar con mayor rapidez. Para obtener más información, consulte Beneficios de la entrega continua. CD también puede significar implementación continua. Para obtener más información, consulte Entrega continua frente a implementación continua.

CV

Consulte visión artificial.

D

datos en reposo

Datos que están estacionarios en la red, como los datos que se encuentran almacenados.

clasificación de datos

Un proceso para identificar y clasificar los datos de su red en función de su importancia y sensibilidad. Es un componente fundamental de cualquier estrategia de administración de riesgos de ciberseguridad porque lo ayuda a determinar los controles de protección y retención adecuados para los datos. La clasificación de datos es un componente del pilar de seguridad

del AWS Well-Architected Framework. Para obtener más información, consulte Clasificación de datos.

desviación de datos

Una variación significativa entre los datos de producción y los datos que se utilizaron para entrenar un modelo de machine learning, o un cambio significativo en los datos de entrada a lo largo del tiempo. La desviación de los datos puede reducir la calidad, la precisión y la imparcialidad generales de las predicciones de los modelos de machine learning.

datos en tránsito

Datos que se mueven de forma activa por la red, por ejemplo, entre los recursos de la red.

malla de datos

Un marco arquitectónico que proporciona una propiedad de datos distribuida y descentralizada con administración y gobierno centralizados.

minimización de datos

El principio de recopilar y procesar solo los datos estrictamente necesarios. Practicar la minimización de los datos Nube de AWS puede reducir los riesgos de privacidad, los costos y la huella de carbono de la analítica.

perímetro de datos

Un conjunto de barreras preventivas en su AWS entorno que ayudan a garantizar que solo las identidades confiables accedan a los recursos confiables desde las redes esperadas. Para obtener más información, consulte Crear un perímetro de datos sobre. AWS

preprocesamiento de datos

Transformar los datos sin procesar en un formato que su modelo de ML pueda analizar fácilmente. El preprocesamiento de datos puede implicar eliminar determinadas columnas o filas y corregir los valores faltantes, incoherentes o duplicados.

procedencia de los datos

El proceso de rastrear el origen y el historial de los datos a lo largo de su ciclo de vida, por ejemplo, la forma en que se generaron, transmitieron y almacenaron los datos.

titular de los datos

Persona cuyos datos se recopilan y procesan.

almacenamiento de datos

Un sistema de administración de datos que respalde la inteligencia empresarial, como el análisis. Los almacenes de datos suelen contener grandes cantidades de datos históricos y, por lo general, se utilizan para consultas y análisis.

lenguaje de definición de datos (DDL)

Instrucciones o comandos para crear o modificar la estructura de tablas y objetos de una base de datos.

lenguaje de manipulación de datos (DML)

Instrucciones o comandos para modificar (insertar, actualizar y eliminar) la información de una base de datos.

DDL

Consulte el lenguaje de definición de bases de datos.

conjunto profundo

Combinar varios modelos de aprendizaje profundo para la predicción. Puede utilizar conjuntos profundos para obtener una predicción más precisa o para estimar la incertidumbre de las predicciones.

aprendizaje profundo

Un subcampo del ML que utiliza múltiples capas de redes neuronales artificiales para identificar el mapeo entre los datos de entrada y las variables objetivo de interés.

defense-in-depth

Un enfoque de seguridad de la información en el que se distribuyen cuidadosamente una serie de mecanismos y controles de seguridad en una red informática para proteger la confidencialidad, la integridad y la disponibilidad de la red y de los datos que contiene. Al adoptar esta estrategia AWS, se añaden varios controles en diferentes capas de la AWS Organizations estructura para ayudar a proteger los recursos. Por ejemplo, un defense-in-depth enfoque podría combinar la autenticación multifactorial, la segmentación de la red y el cifrado.

administrador delegado

En AWS Organizations, un servicio compatible puede registrar una cuenta de AWS miembro para administrar las cuentas de la organización y gestionar los permisos de ese servicio. Esta

cuenta se denomina administrador delegado para ese servicio. Para obtener más información y una lista de servicios compatibles, consulte Servicios que funcionan con AWS Organizations en la documentación de AWS Organizations .

Implementación

El proceso de hacer que una aplicación, características nuevas o correcciones de código se encuentren disponibles en el entorno de destino. La implementación abarca implementar cambios en una base de código y, a continuación, crear y ejecutar esa base en los entornos de la aplicación.

entorno de desarrollo

Consulte entorno.

control de detección

Un control de seguridad que se ha diseñado para detectar, registrar y alertar después de que se produzca un evento. Estos controles son una segunda línea de defensa, ya que lo advierten sobre los eventos de seguridad que han eludido los controles preventivos establecidos. Para obtener más información, consulte Controles de detección en Implementación de controles de seguridad en AWS.

asignación de flujos de valor para el desarrollo (DVSM)

Proceso que se utiliza para identificar y priorizar las restricciones que afectan negativamente a la velocidad y la calidad en el ciclo de vida del desarrollo de software. DVSM amplía el proceso de asignación del flujo de valor diseñado originalmente para las prácticas de fabricación ajustada. Se centra en los pasos y los equipos necesarios para crear y transferir valor a través del proceso de desarrollo de software.

gemelo digital

Representación virtual de un sistema del mundo real, como un edificio, una fábrica, un equipo industrial o una línea de producción. Los gemelos digitales son compatibles con el mantenimiento predictivo, la supervisión remota y la optimización de la producción.

tabla de dimensiones

En un esquema en estrella, tabla más pequeña que contiene los atributos de datos sobre los datos cuantitativos de una tabla de hechos. Los atributos de la tabla de dimensiones suelen ser campos de texto o números discretos que se comportan como texto. Estos atributos se utilizan habitualmente para restringir consultas, filtrar y etiquetar conjuntos de resultados.

desastre

Un evento que impide que una carga de trabajo o un sistema cumplan sus objetivos empresariales en su ubicación principal de implementación. Estos eventos pueden ser desastres naturales, fallos técnicos o el resultado de acciones humanas, como una configuración incorrecta involuntaria o un ataque de malware.

recuperación de desastres (DR)

La estrategia y el proceso que se utilizan para minimizar el tiempo de inactividad y la pérdida de datos ocasionados por un desastre. Para obtener más información, consulte Recuperación ante desastres de cargas de trabajo en AWS: Recovery in the Cloud in the AWS Well-Architected Framework.

DML

Consulte el lenguaje de manipulación de bases de datos.

diseño basado en el dominio

Un enfoque para desarrollar un sistema de software complejo mediante la conexión de sus componentes a dominios en evolución, o a los objetivos empresariales principales, a los que sirve cada componente. Este concepto lo introdujo Eric Evans en su libro, Diseño impulsado por el dominio: abordando la complejidad en el corazón del software (Boston: Addison-Wesley Professional, 2003). Para obtener información sobre cómo utilizar el diseño basado en dominios con el patrón de higos estranguladores, consulte Modernización gradual de los servicios web antiguos de Microsoft ASP.NET (ASMX) mediante contenedores y Amazon API Gateway.

DR

Consulte recuperación ante desastres.

detección de desviaciones

Seguimiento de las desviaciones con respecto a una configuración de referencia. Por ejemplo, puedes usarlo AWS CloudFormation para detectar desviaciones en los recursos del sistema o puedes usarlo AWS Control Tower para detectar cambios en tu landing zone que puedan afectar al cumplimiento de los requisitos de gobierno.

DVSM

Consulte el mapeo del flujo de valor del desarrollo.

E

EDA

Consulte el análisis exploratorio de datos.

computación en la periferia

La tecnología que aumenta la potencia de cálculo de los dispositivos inteligentes en la periferia de una red de IoT. En comparación con la computación en nube, la computación perimetral puede reducir la latencia de la comunicación y mejorar el tiempo de respuesta.

cifrado

Proceso informático que transforma datos de texto plano, legibles por humanos, en texto cifrado.

clave de cifrado

Cadena criptográfica de bits aleatorios que se genera mediante un algoritmo de cifrado. Las claves pueden variar en longitud y cada una se ha diseñado para ser impredecible y única.

endianidad

El orden en el que se almacenan los bytes en la memoria del ordenador. Los sistemas big-endianos almacenan primero el byte más significativo. Los sistemas Little-Endian almacenan primero el byte menos significativo.

punto de conexión

Consulte el punto final del servicio.

servicio de punto de conexión

Servicio que puede alojar en una nube privada virtual (VPC) para compartir con otros usuarios. Puede crear un servicio de punto final AWS PrivateLink y conceder permisos a otros directores Cuentas de AWS o a AWS Identity and Access Management (IAM). Estas cuentas o entidades principales pueden conectarse a su servicio de punto de conexión de forma privada mediante la creación de puntos de conexión de VPC de interfaz. Para obtener más información, consulte Creación de un servicio de punto de conexión en la documentación de Amazon Virtual Private Cloud (Amazon VPC).

planificación de recursos empresariales (ERP)

Un sistema que automatiza y gestiona los procesos empresariales clave (como la contabilidad, el MES y la gestión de proyectos) de una empresa.

cifrado de sobre

El proceso de cifrar una clave de cifrado con otra clave de cifrado. Para obtener más información, consulte el cifrado de sobres en la documentación de AWS Key Management Service (AWS KMS).

environment

Una instancia de una aplicación en ejecución. Los siguientes son los tipos de entornos más comunes en la computación en la nube:

- entorno de desarrollo: instancia de una aplicación en ejecución que solo se encuentra disponible para el equipo principal responsable del mantenimiento de la aplicación. Los entornos de desarrollo se utilizan para probar los cambios antes de promocionarlos a los entornos superiores. Este tipo de entorno a veces se denomina entorno de prueba.
- entornos inferiores: todos los entornos de desarrollo de una aplicación, como los que se utilizan para las compilaciones y pruebas iniciales.
- entorno de producción: instancia de una aplicación en ejecución a la que pueden acceder los usuarios finales. En una canalización de CI/CD, el entorno de producción es el último entorno de implementación.
- entornos superiores: todos los entornos a los que pueden acceder usuarios que no sean del equipo de desarrollo principal. Esto puede incluir un entorno de producción, entornos de preproducción y entornos para las pruebas de aceptación por parte de los usuarios.

epopeya

En las metodologías ágiles, son categorías funcionales que ayudan a organizar y priorizar el trabajo. Las epopeyas brindan una descripción detallada de los requisitos y las tareas de implementación. Por ejemplo, las epopeyas AWS de seguridad de CAF incluyen la gestión de identidades y accesos, los controles de detección, la seguridad de la infraestructura, la protección de datos y la respuesta a incidentes. Para obtener más información sobre las epopeyas en la estrategia de migración de AWS , consulte la Guía de implementación del programa.

PERP

Consulte planificación de recursos empresariales.

análisis de datos de tipo exploratorio (EDA)

El proceso de analizar un conjunto de datos para comprender sus características principales. Se recopilan o agregan datos y, a continuación, se realizan las investigaciones iniciales para

encontrar patrones, detectar anomalías y comprobar las suposiciones. El EDA se realiza mediante el cálculo de estadísticas resumidas y la creación de visualizaciones de datos.

F

tabla de datos

La tabla central de un esquema en forma de estrella. Almacena datos cuantitativos sobre las operaciones comerciales. Normalmente, una tabla de hechos contiene dos tipos de columnas: las que contienen medidas y las que contienen una clave externa para una tabla de dimensiones.

fallan rápidamente

Una filosofía que utiliza pruebas frecuentes e incrementales para reducir el ciclo de vida del desarrollo. Es una parte fundamental de un enfoque ágil.

límite de aislamiento de fallas

En el Nube de AWS, un límite, como una zona de disponibilidad Región de AWS, un plano de control o un plano de datos, que limita el efecto de una falla y ayuda a mejorar la resiliencia de las cargas de trabajo. Para obtener más información, consulte Límites de AWS aislamiento de errores.

rama de característica

Consulte la sucursal.

características

Los datos de entrada que se utilizan para hacer una predicción. Por ejemplo, en un contexto de fabricación, las características pueden ser imágenes que se capturan periódicamente desde la línea de fabricación.

importancia de las características

La importancia que tiene una característica para las predicciones de un modelo. Por lo general, esto se expresa como una puntuación numérica que se puede calcular mediante diversas técnicas, como las explicaciones aditivas de Shapley (SHAP) y los gradientes integrados. Para obtener más información, consulte Interpretabilidad del modelo de aprendizaje automático con:AWS.

transformación de funciones

Optimizar los datos para el proceso de ML, lo que incluye enriquecer los datos con fuentes adicionales, escalar los valores o extraer varios conjuntos de información de un solo campo de datos. Esto permite que el modelo de ML se beneficie de los datos. Por ejemplo, si divide la fecha del “27 de mayo de 2021 00:15:37” en “jueves”, “mayo”, “2021” y “15”, puede ayudar al algoritmo de aprendizaje a aprender patrones matizados asociados a los diferentes componentes de los datos.

FGAC

Consulte el control de acceso detallado.

control de acceso preciso (FGAC)

El uso de varias condiciones que tienen por objetivo permitir o denegar una solicitud de acceso.

migración relámpago

Método de migración de bases de datos que utiliza la replicación continua de datos mediante la captura de datos modificados para migrar los datos en el menor tiempo posible, en lugar de utilizar un enfoque gradual. El objetivo es reducir al mínimo el tiempo de inactividad.

G

bloqueo geográfico

Consulta las restricciones geográficas.

restricciones geográficas (bloqueo geográfico)

En Amazon CloudFront, una opción para impedir que los usuarios de países específicos accedan a las distribuciones de contenido. Puede utilizar una lista de permitidos o bloqueados para especificar los países aprobados y prohibidos. Para obtener más información, consulta Restringir la distribución geográfica del contenido en la CloudFront documentación.

Flujo de trabajo de Gitflow

Un enfoque en el que los entornos inferiores y superiores utilizan diferentes ramas en un repositorio de código fuente. El flujo de trabajo de Gitflow se considera heredado, y el flujo de trabajo basado en enlaces troncales es el enfoque moderno preferido.

estrategia de implementación desde cero

La ausencia de infraestructura existente en un entorno nuevo. Al adoptar una estrategia de implementación desde cero para una arquitectura de sistemas, puede seleccionar todas las tecnologías nuevas sin que estas deban ser compatibles con una infraestructura existente, lo que también se conoce como implementación sobre infraestructura existente. Si está ampliando la infraestructura existente, puede combinar las estrategias de implementación sobre infraestructuras existentes y de implementación desde cero.

barrera de protección

Una regla de alto nivel que ayuda a regular los recursos, las políticas y la conformidad en todas las unidades organizativas (OU). Las barreras de protección preventivas aplican políticas para garantizar la alineación con los estándares de conformidad. Se implementan mediante políticas de control de servicios y límites de permisos de IAM. Las barreras de protección de detección detectan las vulneraciones de las políticas y los problemas de conformidad, y generan alertas para su corrección. Se implementan mediante Amazon AWS Config AWS Security Hub GuardDuty AWS Trusted Advisor, Amazon Inspector y AWS Lambda cheques personalizados.

H

JA

Consulte alta disponibilidad.

migración heterogénea de bases de datos

Migración de la base de datos de origen a una base de datos de destino que utilice un motor de base de datos diferente (por ejemplo, de Oracle a Amazon Aurora). La migración heterogénea suele ser parte de un esfuerzo de rediseño de la arquitectura y convertir el esquema puede ser una tarea compleja. AWS ofrece AWS SCT, lo cual ayuda con las conversiones de esquemas.

alta disponibilidad (HA)

La capacidad de una carga de trabajo para funcionar de forma continua, sin intervención, en caso de desafíos o desastres. Los sistemas de alta disponibilidad están diseñados para realizar una conmutación por error automática, ofrecer un rendimiento de alta calidad de forma constante y gestionar diferentes cargas y fallos con un impacto mínimo en el rendimiento.

modernización histórica

Un enfoque utilizado para modernizar y actualizar los sistemas de tecnología operativa (TO) a fin de satisfacer mejor las necesidades de la industria manufacturera. Un histórico es un tipo de base de datos que se utiliza para recopilar y almacenar datos de diversas fuentes en una fábrica.

migración homogénea de bases de datos

Migración de la base de datos de origen a una base de datos de destino que comparte el mismo motor de base de datos (por ejemplo, Microsoft SQL Server a Amazon RDS para SQL Server). La migración homogénea suele formar parte de un esfuerzo para volver a alojar o redefinir la plataforma. Puede utilizar las utilidades de bases de datos nativas para migrar el esquema.

datos recientes

Datos a los que se accede con frecuencia, como datos en tiempo real o datos traslacionales recientes. Por lo general, estos datos requieren un nivel o una clase de almacenamiento de alto rendimiento para proporcionar respuestas rápidas a las consultas.

hotfix

Una solución urgente para un problema crítico en un entorno de producción. Debido a su urgencia, las revisiones suelen realizarse fuera del flujo de trabajo habitual de las DevOps versiones.

periodo de hiperatención

Periodo, inmediatamente después de la transición, durante el cual un equipo de migración administra y monitorea las aplicaciones migradas en la nube para solucionar cualquier problema. Por lo general, este periodo dura de 1 a 4 días. Al final del periodo de hiperatención, el equipo de migración suele transferir la responsabilidad de las aplicaciones al equipo de operaciones en la nube.

I

IaC

Vea la infraestructura como código.

políticas basadas en identidad

Política asociada a uno o más directores de IAM que define sus permisos en el Nube de AWS entorno.

aplicación inactiva

Aplicación que utiliza un promedio de CPU y memoria de entre 5 y 20 por ciento durante un periodo de 90 días. En un proyecto de migración, es habitual retirar estas aplicaciones o mantenerlas en las instalaciones.

IIoT

Consulte Internet de las cosas industrial.

infraestructura inmutable

Un modelo que implementa una nueva infraestructura para las cargas de trabajo de producción en lugar de actualizar, parchear o modificar la infraestructura existente. Las infraestructuras inmutables son intrínsecamente más consistentes, fiables y predecibles que las infraestructuras mutables. Para obtener más información, consulte las prácticas recomendadas para implementar con una infraestructura inmutable en Well-Architected Framework AWS .

VPC entrante (de entrada)

En una arquitectura de AWS cuentas múltiples, una VPC que acepta, inspecciona y enruta las conexiones de red desde fuera de una aplicación. La Arquitectura de referencia de seguridad de AWS recomienda configurar su cuenta de red con VPC entrantes, salientes y de inspección para proteger la interfaz bidireccional entre su aplicación e Internet en general.

migración gradual

Estrategia de transición en la que se migra la aplicación en partes pequeñas en lugar de realizar una transición única y completa. Por ejemplo, puede trasladar inicialmente solo unos pocos microservicios o usuarios al nuevo sistema. Tras comprobar que todo funciona correctamente, puede trasladar microservicios o usuarios adicionales de forma gradual hasta que pueda retirar su sistema heredado. Esta estrategia reduce los riesgos asociados a las grandes migraciones.

Industria 4.0

Un término que Klaus Schwab introdujo en 2016 para referirse a la modernización de los procesos de fabricación mediante avances en la conectividad, los datos en tiempo real, la automatización, el análisis y la inteligencia artificial/aprendizaje automático.

infraestructura

Todos los recursos y activos que se encuentran en el entorno de una aplicación.

infraestructura como código (IaC)

Proceso de aprovisionamiento y administración de la infraestructura de una aplicación mediante un conjunto de archivos de configuración. La IaC se ha diseñado para ayudarlo a centralizar la administración de la infraestructura, estandarizar los recursos y escalar con rapidez a fin de que los entornos nuevos sean repetibles, fiables y consistentes.

Internet de las cosas industrial (IIoT)

El uso de sensores y dispositivos conectados a Internet en los sectores industriales, como el productivo, el eléctrico, el automotriz, el sanitario, el de las ciencias de la vida y el de la agricultura. Para obtener más información, consulte Creación de una estrategia de transformación digital del Internet de las cosas industrial (IIoT).

VPC de inspección

En una arquitectura de AWS cuentas múltiples, una VPC centralizada que gestiona las inspecciones del tráfico de red entre las VPC (iguales o Regiones de AWS diferentes), Internet y las redes locales. La Arquitectura de referencia de seguridad de AWS recomienda configurar su cuenta de red con VPC entrantes, salientes y de inspección para proteger la interfaz bidireccional entre su aplicación e Internet en general.

Internet de las cosas (IoT)

Red de objetos físicos conectados con sensores o procesadores integrados que se comunican con otros dispositivos y sistemas a través de Internet o de una red de comunicación local. Para obtener más información, consulte ¿Qué es IoT?.

interpretabilidad

Característica de un modelo de machine learning que describe el grado en que un ser humano puede entender cómo las predicciones del modelo dependen de sus entradas. Para más información, consulte Interpretabilidad del modelo de machine learning con AWS.

IoT

Consulte Internet de las cosas.

biblioteca de información de TI (ITIL)

Conjunto de prácticas recomendadas para ofrecer servicios de TI y alinearlos con los requisitos empresariales. La ITIL proporciona la base para la ITSM.

administración de servicios de TI (ITSM)

Actividades asociadas con el diseño, la implementación, la administración y el soporte de los servicios de TI para una organización. Para obtener información sobre la integración de las operaciones en la nube con las herramientas de ITSM, consulte la Guía de integración de operaciones.

ITIL

Consulte la biblioteca de información de TI.

ITSM

Consulte Administración de servicios de TI.

L

control de acceso basado en etiquetas (LBAC)

Una implementación del control de acceso obligatorio (MAC) en la que a los usuarios y a los propios datos se les asigna explícitamente un valor de etiqueta de seguridad. La intersección entre la etiqueta de seguridad del usuario y la etiqueta de seguridad de los datos determina qué filas y columnas puede ver el usuario.

zona de aterrizaje

Una landing zone es un AWS entorno multicuenta bien diseñado, escalable y seguro. Este es un punto de partida desde el cual las empresas pueden lanzar e implementar rápidamente cargas de trabajo y aplicaciones con confianza en su entorno de seguridad e infraestructura. Para obtener más información sobre las zonas de aterrizaje, consulte Configuración de un entorno de AWS seguro y escalable con varias cuentas.

migración grande

Migración de 300 servidores o más.

LBAC

Consulte el control de acceso basado en etiquetas.

privilegio mínimo

La práctica recomendada de seguridad que consiste en conceder los permisos mínimos necesarios para realizar una tarea. Para obtener más información, consulte Aplicar permisos de privilegio mínimo en la documentación de IAM.

migrar mediante lift-and-shift

Ver 7 Rs.

sistema little-endian

Un sistema que almacena primero el byte menos significativo. Véase también endianness.

entornos inferiores

Véase entorno.

M

machine learning (ML)

Un tipo de inteligencia artificial que utiliza algoritmos y técnicas para el reconocimiento y el aprendizaje de patrones. El ML analiza y aprende de los datos registrados, como los datos del Internet de las cosas (IoT), para generar un modelo estadístico basado en patrones. Para más información, consulte Machine learning.

rama principal

Ver sucursal.

malware

Software diseñado para comprometer la seguridad o la privacidad de la computadora. El malware puede interrumpir los sistemas informáticos, filtrar información confidencial u obtener acceso no autorizado. Algunos ejemplos de malware son los virus, los gusanos, el ransomware, los troyanos, el spyware y los registradores de pulsaciones de teclas.

servicios gestionados

servicios de AWS para los que AWS opera la capa de infraestructura, el sistema operativo y las plataformas, y usted accede a los puntos finales para almacenar y recuperar datos. Amazon Simple Storage Service (Amazon S3) y Amazon DynamoDB son ejemplos de servicios gestionados. También se conocen como servicios abstractos.

sistema de ejecución de fabricación (MES)

Un sistema de software para rastrear, monitorear, documentar y controlar los procesos de producción que convierten las materias primas en productos terminados en el taller.

MAP

Consulte Migration Acceleration Program.

mecanismo

Un proceso completo en el que se crea una herramienta, se impulsa su adopción y, a continuación, se inspeccionan los resultados para realizar ajustes. Un mecanismo es un ciclo que se refuerza y mejora a sí mismo a medida que funciona. Para obtener más información, consulte Creación de mecanismos en el AWS Well-Architected Framework.

cuenta de miembro

Todas las Cuentas de AWS demás cuentas, excepto la de administración, que forman parte de una organización. AWS Organizations Una cuenta no puede pertenecer a más de una organización a la vez.

MES

Consulte el sistema de ejecución de la fabricación.

Transporte telemétrico de Message Queue Queue (MQTT)

Un protocolo de comunicación ligero machine-to-machine (M2M), basado en el patrón de publicación/suscripción, para dispositivos de IoT con recursos limitados.

microservicio

Un servicio pequeño e independiente que se comunica a través de API bien definidas y que, por lo general, es propiedad de equipos pequeños e independientes. Por ejemplo, un sistema de seguros puede incluir microservicios que se adapten a las capacidades empresariales, como las de ventas o marketing, o a subdominios, como las de compras, reclamaciones o análisis. Los beneficios de los microservicios incluyen la agilidad, la escalabilidad flexible, la facilidad de implementación, el código reutilizable y la resiliencia. Para obtener más información, consulte Integrar microservicios mediante servicios sin servidor. AWS

arquitectura de microservicios

Un enfoque para crear una aplicación con componentes independientes que ejecutan cada proceso de la aplicación como un microservicio. Estos microservicios se comunican a través de

una interfaz bien definida mediante API ligeras. Cada microservicio de esta arquitectura se puede actualizar, implementar y escalar para satisfacer la demanda de funciones específicas de una aplicación. Para obtener más información, consulte Implementación de microservicios en. AWS

Programa de aceleración de la migración (MAP)

Un AWS programa que proporciona soporte de consultoría, formación y servicios para ayudar a las organizaciones a crear una base operativa sólida para migrar a la nube y para ayudar a compensar el costo inicial de las migraciones. El MAP incluye una metodología de migración para ejecutar las migraciones antiguas de forma metódica y un conjunto de herramientas para automatizar y acelerar los escenarios de migración más comunes.

migración a escala

Proceso de transferencia de la mayoría de la cartera de aplicaciones a la nube en oleadas, con más aplicaciones desplazadas a un ritmo más rápido en cada oleada. En esta fase, se utilizan las prácticas recomendadas y las lecciones aprendidas en las fases anteriores para implementar una fábrica de migración de equipos, herramientas y procesos con el fin de agilizar la migración de las cargas de trabajo mediante la automatización y la entrega ágil. Esta es la tercera fase de la estrategia de migración de AWS.

fábrica de migración

Equipos multifuncionales que agilizan la migración de las cargas de trabajo mediante enfoques automatizados y ágiles. Los equipos de las fábricas de migración suelen incluir a analistas y propietarios de operaciones, empresas, ingenieros de migración, desarrolladores y DevOps profesionales que trabajan a pasos agigantados. Entre el 20 y el 50 por ciento de la cartera de aplicaciones empresariales se compone de patrones repetidos que pueden optimizarse mediante un enfoque de fábrica. Para obtener más información, consulte la discusión sobre las fábricas de migración y la Guía de fábricas de migración a la nube en este contenido.

metadatos de migración

Información sobre la aplicación y el servidor que se necesita para completar la migración. Cada patrón de migración requiere un conjunto diferente de metadatos de migración. Algunos ejemplos de metadatos de migración son la subred de destino, el grupo de seguridad y AWS la cuenta.

patrón de migración

Tarea de migración repetible que detalla la estrategia de migración, el destino de la migración y la aplicación o el servicio de migración utilizados. Ejemplo: rehospede la migración a Amazon EC2 AWS con Application Migration Service.

Migration Portfolio Assessment (MPA)

Una herramienta en línea que proporciona información para validar el modelo de negocio para migrar a. Nube de AWS La MPA ofrece una evaluación detallada de la cartera (adecuación del tamaño de los servidores, precios, comparaciones del costo total de propiedad, análisis de los costos de migración), así como una planificación de la migración (análisis y recopilación de datos de aplicaciones, agrupación de aplicaciones, priorización de la migración y planificación de oleadas). La herramienta MPA (requiere iniciar sesión) está disponible de forma gratuita para todos los AWS consultores y consultores asociados de APN.

Evaluación de la preparación para la migración (MRA)

Proceso que consiste en obtener información sobre el estado de preparación de una organización para la nube, identificar sus puntos fuertes y débiles y elaborar un plan de acción para cerrar las brechas identificadas mediante el AWS CAF. Para obtener más información, consulte la Guía de preparación para la migración. La MRA es la primera fase de la estrategia de migración de AWS.

estrategia de migración

El enfoque utilizado para migrar una carga de trabajo a. Nube de AWS Para obtener más información, consulte la entrada de las 7 R de este glosario y consulte Movilice a su organización para acelerar las migraciones a gran escala.

ML

Consulte el aprendizaje automático.

modernización

Transformar una aplicación obsoleta (antigua o monolítica) y su infraestructura en un sistema ágil, elástico y de alta disponibilidad en la nube para reducir los gastos, aumentar la eficiencia y aprovechar las innovaciones. Para obtener más información, consulte Estrategia para modernizar las aplicaciones en el Nube de AWS.

evaluación de la preparación para la modernización

Evaluación que ayuda a determinar la preparación para la modernización de las aplicaciones de una organización; identifica los beneficios, los riesgos y las dependencias; y determina qué tan bien la organización puede soportar el estado futuro de esas aplicaciones. El resultado de la evaluación es un esquema de la arquitectura objetivo, una hoja de ruta que detalla las fases de desarrollo y los hitos del proceso de modernización y un plan de acción para abordar las brechas identificadas. Para obtener más información, consulte Evaluación de la preparación para la modernización de las aplicaciones en el Nube de AWS.

aplicaciones monolíticas (monolitos)

Aplicaciones que se ejecutan como un único servicio con procesos estrechamente acoplados. Las aplicaciones monolíticas presentan varios inconvenientes. Si una característica de la aplicación experimenta un aumento en la demanda, se debe escalar toda la arquitectura. Agregar o mejorar las características de una aplicación monolítica también se vuelve más complejo a medida que crece la base de código. Para solucionar problemas con la aplicación, puede utilizar una arquitectura de microservicios. Para obtener más información, consulte Descomposición de monolitos en microservicios.

MAPA

Consulte la evaluación de la cartera de migración.

MQTT

Consulte Message Queue Queue Telemetría y Transporte.

clasificación multiclase

Un proceso que ayuda a generar predicciones para varias clases (predice uno de más de dos resultados). Por ejemplo, un modelo de ML podría preguntar "¿Este producto es un libro, un automóvil o un teléfono?" o "¿Qué categoría de productos es más interesante para este cliente?".

infraestructura mutable

Un modelo que actualiza y modifica la infraestructura existente para las cargas de trabajo de producción. Para mejorar la coherencia, la fiabilidad y la previsibilidad, el AWS Well-Architected Framework recomienda el uso de una infraestructura inmutable como práctica recomendada.

O

OAC

Consulte el control de acceso de origen.

OAI

Consulte la identidad de acceso de origen.

OCM

Consulte gestión del cambio organizacional.

migración fuera de línea

Método de migración en el que la carga de trabajo de origen se elimina durante el proceso de migración. Este método implica un tiempo de inactividad prolongado y, por lo general, se utiliza para cargas de trabajo pequeñas y no críticas.

OI

Consulte integración de operaciones.

OLA

Véase el acuerdo a nivel operativo.

migración en línea

Método de migración en el que la carga de trabajo de origen se copia al sistema de destino sin que se desconecte. Las aplicaciones que están conectadas a la carga de trabajo pueden seguir funcionando durante la migración. Este método implica un tiempo de inactividad nulo o mínimo y, por lo general, se utiliza para cargas de trabajo de producción críticas.

OPC-UA

Consulte Open Process Communications: arquitectura unificada.

Comunicaciones de proceso abierto: arquitectura unificada (OPC-UA)

Un protocolo de comunicación machine-to-machine (M2M) para la automatización industrial. El OPC-UA proporciona un estándar de interoperabilidad con esquemas de cifrado, autenticación y autorización de datos.

acuerdo de nivel operativo (OLA)

Acuerdo que aclara lo que los grupos de TI operativos se comprometen a ofrecerse entre sí, para respaldar un acuerdo de nivel de servicio (SLA).

revisión de la preparación operativa (ORR)

Una lista de preguntas y las mejores prácticas asociadas que le ayudan a comprender, evaluar, prevenir o reducir el alcance de los incidentes y posibles fallos. Para obtener más información, consulte Operational Readiness Reviews (ORR) en AWS Well-Architected Framework.

tecnología operativa (OT)

Sistemas de hardware y software que funcionan con el entorno físico para controlar las operaciones, los equipos y la infraestructura industriales. En la industria manufacturera, la

integración de los sistemas de TO y tecnología de la información (TI) es un enfoque clave para las transformaciones de la industria 4.0.

integración de operaciones (OI)

Proceso de modernización de las operaciones en la nube, que implica la planificación de la preparación, la automatización y la integración. Para obtener más información, consulte la Guía de integración de las operaciones.

registro de seguimiento organizativo

Un registro creado por el AWS CloudTrail que se registran todos los eventos para todos Cuentas de AWS los miembros de una organización AWS Organizations. Este registro de seguimiento se crea en cada Cuenta de AWS que forma parte de la organización y realiza un seguimiento de la actividad en cada cuenta. Para obtener más información, consulte Crear un registro para una organización en la CloudTrail documentación.

administración del cambio organizacional (OCM)

Marco para administrar las transformaciones empresariales importantes y disruptivas desde la perspectiva de las personas, la cultura y el liderazgo. La OCM ayuda a las empresas a prepararse para nuevos sistemas y estrategias y a realizar la transición a ellos, al acelerar la adopción de cambios, abordar los problemas de transición e impulsar cambios culturales y organizacionales. En la estrategia de AWS migración, este marco se denomina aceleración de personal, debido a la velocidad de cambio que requieren los proyectos de adopción de la nube. Para obtener más información, consulte la Guía de OCM.

control de acceso de origen (OAC)

En CloudFront, una opción mejorada para restringir el acceso y proteger el contenido del Amazon Simple Storage Service (Amazon S3). El OAC admite todos los buckets de S3 Regiones de AWS, el cifrado del lado del servidor AWS KMS (SSE-KMS) y las solicitudes dinámicas `PUT` y `DELETE` dirigidas al bucket de S3.

identidad de acceso de origen (OAI)

En CloudFront, una opción para restringir el acceso y proteger el contenido de Amazon S3. Cuando utiliza OAI, CloudFront crea un principal con el que Amazon S3 puede autenticarse. Los directores autenticados solo pueden acceder al contenido de un bucket de S3 a través de una distribución específica. CloudFront Consulte también el OAC, que proporciona un control de acceso más detallado y mejorado.

O

Consulte la revisión de la preparación operativa.

NO

Consulte tecnología operativa.

VPC saliente (de salida)

En una arquitectura de AWS cuentas múltiples, una VPC que gestiona las conexiones de red que se inician desde una aplicación. La Arquitectura de referencia de seguridad de AWS recomienda configurar su cuenta de red con VPC entrantes, salientes y de inspección para proteger la interfaz bidireccional entre su aplicación e Internet en general.

P

límite de permisos

Una política de administración de IAM que se adjunta a las entidades principales de IAM para establecer los permisos máximos que puede tener el usuario o el rol. Para obtener más información, consulte Límites de permisos en la documentación de IAM.

información de identificación personal (PII)

Información que, vista directamente o combinada con otros datos relacionados, puede utilizarse para deducir de manera razonable la identidad de una persona. Algunos ejemplos de información de identificación personal son los nombres, las direcciones y la información de contacto.

PII

Consulte la información de identificación personal.

manual de estrategias

Conjunto de pasos predefinidos que capturan el trabajo asociado a las migraciones, como la entrega de las funciones de operaciones principales en la nube. Un manual puede adoptar la forma de scripts, manuales de procedimientos automatizados o resúmenes de los procesos o pasos necesarios para operar un entorno modernizado.

PLC

Consulte controlador lógico programable.

PLM

Consulte la gestión del ciclo de vida del producto.

política

Un objeto que puede definir los permisos (consulte la política basada en la identidad), especifique las condiciones de acceso (consulte la política basada en los recursos) o defina los permisos máximos para todas las cuentas de una organización AWS Organizations (consulte la política de control de servicios).

persistencia políglota

Elegir de forma independiente la tecnología de almacenamiento de datos de un microservicio en función de los patrones de acceso a los datos y otros requisitos. Si sus microservicios tienen la misma tecnología de almacenamiento de datos, pueden enfrentarse a desafíos de implementación o experimentar un rendimiento deficiente. Los microservicios se implementan más fácilmente y logran un mejor rendimiento y escalabilidad si utilizan el almacén de datos que mejor se adapte a sus necesidades. Para obtener más información, consulte Habilitación de la persistencia de datos en los microservicios.

evaluación de cartera

Proceso de detección, análisis y priorización de la cartera de aplicaciones para planificar la migración. Para obtener más información, consulte la Evaluación de la preparación para la migración.

predicate

Una condición de consulta que devuelve `true` o`false`, por lo general, se encuentra en una cláusula. `WHERE`

pulsar un predicado

Técnica de optimización de consultas de bases de datos que filtra los datos de la consulta antes de transferirlos. Esto reduce la cantidad de datos que se deben recuperar y procesar de la base de datos relacional y mejora el rendimiento de las consultas.

control preventivo

Un control de seguridad diseñado para evitar que ocurra un evento. Estos controles son la primera línea de defensa para evitar el acceso no autorizado o los cambios no deseados en la red. Para obtener más información, consulte Controles preventivos en Implementación de controles de seguridad en AWS.

entidad principal

Una entidad AWS que puede realizar acciones y acceder a los recursos. Esta entidad suele ser un usuario raíz para un Cuenta de AWS rol de IAM o un usuario. Para obtener más información, consulte Entidad principal en Términos y conceptos de roles en la documentación de IAM.

Privacidad desde el diseño

Un enfoque de ingeniería de sistemas que tiene en cuenta la privacidad durante todo el proceso de ingeniería.

zonas alojadas privadas

Contenedor que aloja información acerca de cómo desea que responda Amazon Route 53 a las consultas de DNS de un dominio y sus subdominios en una o varias VPC. Para obtener más información, consulte Uso de zonas alojadas privadas en la documentación de Route 53.

control proactivo

Un control de seguridad diseñado para evitar el despliegue de recursos no conformes. Estos controles escanean los recursos antes de aprovisionarlos. Si el recurso no cumple con el control, significa que no está aprovisionado. Para obtener más información, consulte la guía de referencia de controles en la AWS Control Tower documentación y consulte Controles proactivos en Implementación de controles de seguridad en AWS.

gestión del ciclo de vida del producto (PLM)

La gestión de los datos y los procesos de un producto a lo largo de todo su ciclo de vida, desde el diseño, el desarrollo y el lanzamiento, pasando por el crecimiento y la madurez, hasta el rechazo y la retirada.

entorno de producción

Consulte el entorno.

controlador lógico programable (PLC)

En la fabricación, una computadora adaptable y altamente confiable que monitorea las máquinas y automatiza los procesos de fabricación.

seudonimización

El proceso de reemplazar los identificadores personales de un conjunto de datos por valores de marcadores de posición. La seudonimización puede ayudar a proteger la privacidad personal. Los datos seudonimizados siguen considerándose datos personales.

publicar/suscribirse (pub/sub)

Un patrón que permite las comunicaciones asíncronas entre microservicios para mejorar la escalabilidad y la capacidad de respuesta. Por ejemplo, en un MES basado en microservicios, un microservicio puede publicar mensajes de eventos en un canal al que se puedan suscribir otros microservicios. El sistema puede añadir nuevos microservicios sin cambiar el servicio de publicación.

Q

plan de consulta

Serie de pasos, como instrucciones, que se utilizan para acceder a los datos de un sistema de base de datos relacional SQL.

regresión del plan de consulta

El optimizador de servicios de la base de datos elige un plan menos óptimo que antes de un cambio determinado en el entorno de la base de datos. Los cambios en estadísticas, restricciones, configuración del entorno, enlaces de parámetros de consultas y actualizaciones del motor de base de datos PostgreSQL pueden provocar una regresión del plan.

R

Matriz RACI

Véase responsable, responsable, consultado, informado (RACI).

ransomware

Software malicioso que se ha diseñado para bloquear el acceso a un sistema informático o a los datos hasta que se efectúe un pago.

Matriz RASCI

Véase responsable, responsable, consultado, informado (RACI).

RCAC

Consulte control de acceso por filas y columnas.

read replica

Una copia de una base de datos que se utiliza con fines de solo lectura. Puede enrutar las consultas a la réplica de lectura para reducir la carga en la base de datos principal.

rediseñar

Ver 7 Rs.

objetivo de punto de recuperación (RPO)

La cantidad de tiempo máximo aceptable desde el último punto de recuperación de datos. Esto determina qué se considera una pérdida de datos aceptable entre el último punto de recuperación y la interrupción del servicio.

objetivo de tiempo de recuperación (RTO)

La demora máxima aceptable entre la interrupción del servicio y el restablecimiento del servicio.

refactorizar

Ver 7 Rs.

Región

Una colección de AWS recursos en un área geográfica. Cada uno Región de AWS está aislado e independiente de los demás para proporcionar tolerancia a las fallas, estabilidad y resiliencia. Para obtener más información, consulte Regiones de AWS Especificar qué cuenta puede usar.

regresión

Una técnica de ML que predice un valor numérico. Por ejemplo, para resolver el problema de "¿A qué precio se venderá esta casa?", un modelo de ML podría utilizar un modelo de regresión lineal para predecir el precio de venta de una vivienda en función de datos conocidos sobre ella (por ejemplo, los metros cuadrados).

volver a alojar

Consulte 7 Rs.

versión

En un proceso de implementación, el acto de promover cambios en un entorno de producción.

trasladarse

Ver 7 Rs.

redefinir la plataforma

Ver 7 Rs.

recompra

Ver 7 Rs.

resilencia

La capacidad de una aplicación para resistir las interrupciones o recuperarse de ellas. La alta disponibilidad y la recuperación ante desastres son consideraciones comunes a la hora de planificar la resiliencia en el. Nube de AWS Para obtener más información, consulte Nube de AWS Resiliencia.

política basada en recursos

Una política asociada a un recurso, como un bucket de Amazon S3, un punto de conexión o una clave de cifrado. Este tipo de política especifica a qué entidades principales se les permite el acceso, las acciones compatibles y cualquier otra condición que deba cumplirse.

matriz responsable, confiable, consultada e informada (RACI)

Una matriz que define las funciones y responsabilidades de todas las partes involucradas en las actividades de migración y las operaciones de la nube. El nombre de la matriz se deriva de los tipos de responsabilidad definidos en la matriz: responsable (R), contable (A), consultado (C) e informado (I). El tipo de soporte (S) es opcional. Si incluye el soporte, la matriz se denomina matriz RASCI y, si la excluye, se denomina matriz RACI.

control receptivo

Un control de seguridad que se ha diseñado para corregir los eventos adversos o las desviaciones con respecto a su base de seguridad. Para obtener más información, consulte Controles receptivos en Implementación de controles de seguridad en AWS.

retain

Consulte 7 Rs.

jubilarse

Ver 7 Rs.

rotación

Proceso de actualizar periódicamente un secreto para dificultar el acceso de un atacante a las credenciales.

control de acceso por filas y columnas (RCAC)

El uso de expresiones SQL básicas y flexibles que tienen reglas de acceso definidas. El RCAC consta de permisos de fila y máscaras de columnas.

RPO

Consulte el objetivo del punto de recuperación.

RTO

Consulte el objetivo de tiempo de recuperación.

manual de procedimientos

Conjunto de procedimientos manuales o automatizados necesarios para realizar una tarea específica. Por lo general, se diseñan para agilizar las operaciones o los procedimientos repetitivos con altas tasas de error.

S

SAML 2.0

Un estándar abierto que utilizan muchos proveedores de identidad (IdPs). Esta función permite el inicio de sesión único (SSO) federado, de modo que los usuarios pueden iniciar sesión AWS Management Console o llamar a las operaciones de la AWS API sin tener que crear un usuario en IAM para todos los miembros de la organización. Para obtener más información sobre la federación basada en SAML 2.0, consulte Acerca de la federación basada en SAML 2.0 en la documentación de IAM.

SCADA

Consulte el control de supervisión y la adquisición de datos.

SCP

Consulte la política de control de servicios.

secreta

Información confidencial o restringida, como una contraseña o credenciales de usuario, que almacene de forma cifrada. AWS Secrets Manager Se compone del valor secreto y sus

metadatos. El valor secreto puede ser binario, una sola cadena o varias cadenas. Para obtener más información, consulta ¿Qué hay en un secreto de Secrets Manager? en la documentación de Secrets Manager.

control de seguridad

Barrera de protección técnica o administrativa que impide, detecta o reduce la capacidad de un agente de amenazas para aprovechar una vulnerabilidad de seguridad. Hay cuatro tipos principales de controles de seguridad: preventivos, de detección, de respuesta y proactivos.

refuerzo de la seguridad

Proceso de reducir la superficie expuesta a ataques para hacerla más resistente a los ataques. Esto puede incluir acciones, como la eliminación de los recursos que ya no se necesitan, la implementación de prácticas recomendadas de seguridad consistente en conceder privilegios mínimos o la desactivación de características innecesarias en los archivos de configuración.

sistema de información sobre seguridad y administración de eventos (SIEM)

Herramientas y servicios que combinan sistemas de administración de información sobre seguridad (SIM) y de administración de eventos de seguridad (SEM). Un sistema de SIEM recopila, monitorea y analiza los datos de servidores, redes, dispositivos y otras fuentes para detectar amenazas y brechas de seguridad y generar alertas.

automatización de la respuesta de seguridad

Una acción predefinida y programada que está diseñada para responder automáticamente a un evento de seguridad o remediarlo. Estas automatizaciones sirven como controles de seguridad detectables o adaptables que le ayudan a implementar las mejores prácticas AWS de seguridad. Algunos ejemplos de acciones de respuesta automatizadas incluyen la modificación de un grupo de seguridad de VPC, la aplicación de parches a una instancia de Amazon EC2 o la rotación de credenciales.

cifrado del servidor

Cifrado de los datos en su destino, por parte de quien servicio de AWS los recibe.

política de control de servicio (SCP)

Una política que proporciona un control centralizado de los permisos de todas las cuentas de una organización en AWS Organizations. Las SCP definen barreras de protección o establecen límites a las acciones que un administrador puede delegar en los usuarios o roles. Puede utilizar las SCP como listas de permitidos o rechazados, para especificar qué servicios o acciones se

encuentra permitidos o prohibidos. Para obtener más información, consulte las políticas de control de servicios en la AWS Organizations documentación.

punto de enlace de servicio

La URL del punto de entrada de un servicio de AWS. Para conectarse mediante programación a un servicio de destino, puede utilizar un punto de conexión. Para obtener más información, consulte Puntos de conexión de servicio de AWS en Referencia general de AWS.

acuerdo de nivel de servicio (SLA)

Acuerdo que aclara lo que un equipo de TI se compromete a ofrecer a los clientes, como el tiempo de actividad y el rendimiento del servicio.

indicador de nivel de servicio (SLI)

Medición de un aspecto del rendimiento de un servicio, como la tasa de errores, la disponibilidad o el rendimiento.

objetivo de nivel de servicio (SLO)

Una métrica objetivo que representa el estado de un servicio, medido mediante un indicador de nivel de servicio.

modelo de responsabilidad compartida

Un modelo que describe la responsabilidad que compartes con respecto a la seguridad y AWS el cumplimiento de la nube. AWS es responsable de la seguridad de la nube, mientras que usted es responsable de la seguridad en la nube. Para obtener más información, consulte el Modelo de responsabilidad compartida.

SIEM

Consulte la información de seguridad y el sistema de gestión de eventos.

punto único de fallo (SPOF)

Una falla en un único componente crítico de una aplicación que puede interrumpir el sistema.

SLA

Consulte el acuerdo de nivel de servicio.

SLI

Consulte el indicador de nivel de servicio.

ASÍ QUE

Consulte el objetivo de nivel de servicio.

split-and-seed modelo

Un patrón para escalar y acelerar los proyectos de modernización. A medida que se definen las nuevas funciones y los lanzamientos de los productos, el equipo principal se divide para crear nuevos equipos de productos. Esto ayuda a ampliar las capacidades y los servicios de su organización, mejora la productividad de los desarrolladores y apoya la innovación rápida. Para obtener más información, consulte Enfoque gradual para modernizar las aplicaciones en el. Nube de AWS

SPOT

Consulte el punto único de falla.

esquema en forma de estrella

Estructura organizativa de una base de datos que utiliza una tabla de datos grande para almacenar datos transaccionales o medidos y una o más tablas dimensionales más pequeñas para almacenar los atributos de los datos. Esta estructura está diseñada para usarse en un almacén de datos o con fines de inteligencia empresarial.

patrón de higo estrangulador

Un enfoque para modernizar los sistemas monolíticos mediante la reescritura y el reemplazo gradual de las funciones del sistema hasta que se pueda desmantelar el sistema heredado. Este patrón utiliza la analogía de una higuera que crece hasta convertirse en un árbol estable y, finalmente, se apodera y reemplaza a su host. El patrón fue presentado por Martin Fowler como una forma de gestionar el riesgo al reescribir sistemas monolíticos. Para ver un ejemplo con la aplicación de este patrón, consulte Modernización gradual de los servicios web antiguos de Microsoft ASP.NET (ASMX) mediante contenedores y Amazon API Gateway.

subred

Un intervalo de direcciones IP en la VPC. Una subred debe residir en una sola zona de disponibilidad.

supervisión, control y adquisición de datos (SCADA)

En la industria manufacturera, un sistema que utiliza hardware y software para monitorear los activos físicos y las operaciones de producción.

cifrado simétrico

Un algoritmo de cifrado que utiliza la misma clave para cifrar y descifrar los datos.

pruebas sintéticas

Probar un sistema de manera que simule las interacciones de los usuarios para detectar posibles problemas o monitorear el rendimiento. Puede usar Amazon CloudWatch Synthetics para crear estas pruebas.

T

etiquetas

Pares clave-valor que actúan como metadatos para organizar los recursos. AWS Las etiquetas pueden ayudarle a administrar, identificar, organizar, buscar y filtrar recursos. Para obtener más información, consulte Etiquetado de los recursos de AWS.

variable de destino

El valor que intenta predecir en el ML supervisado. Esto también se conoce como variable de resultado. Por ejemplo, en un entorno de fabricación, la variable objetivo podría ser un defecto del producto.

lista de tareas

Herramienta que se utiliza para hacer un seguimiento del progreso mediante un manual de procedimientos. La lista de tareas contiene una descripción general del manual de procedimientos y una lista de las tareas generales que deben completarse. Para cada tarea general, se incluye la cantidad estimada de tiempo necesario, el propietario y el progreso.

entorno de prueba

Consulte entorno.

entrenamiento

Proporcionar datos de los que pueda aprender su modelo de ML. Los datos de entrenamiento deben contener la respuesta correcta. El algoritmo de aprendizaje encuentra patrones en los datos de entrenamiento que asignan los atributos de los datos de entrada al destino (la respuesta que desea predecir). Genera un modelo de ML que captura estos patrones. Luego, el modelo de ML se puede utilizar para obtener predicciones sobre datos nuevos para los que no se conoce el destino.

puerta de enlace de tránsito

Centro de tránsito de red que puede utilizar para interconectar las VPC y las redes en las instalaciones. Para obtener más información, consulte Qué es una pasarela de tránsito en la AWS Transit Gateway documentación.

flujo de trabajo basado en enlaces troncales

Un enfoque en el que los desarrolladores crean y prueban características de forma local en una rama de característica y, a continuación, combinan esos cambios en la rama principal. Luego, la rama principal se adapta a los entornos de desarrollo, preproducción y producción, de forma secuencial.

acceso de confianza

Otorgar permisos a un servicio que especifique para realizar tareas en su organización AWS Organizations y en sus cuentas en su nombre. El servicio de confianza crea un rol vinculado al servicio en cada cuenta, cuando ese rol es necesario, para realizar las tareas de administración por usted. Para obtener más información, consulte AWS Organizations Utilización con otros AWS servicios en la AWS Organizations documentación.

ajuste

Cambiar aspectos de su proceso de formación a fin de mejorar la precisión del modelo de ML. Por ejemplo, puede entrenar el modelo de ML al generar un conjunto de etiquetas, incorporar etiquetas y, luego, repetir estos pasos varias veces con diferentes ajustes para optimizar el modelo.

equipo de dos pizzas

Un DevOps equipo pequeño al que puedes alimentar con dos pizzas. Un equipo formado por dos integrantes garantiza la mejor oportunidad posible de colaboración en el desarrollo de software.

U

incertidumbre

Un concepto que hace referencia a información imprecisa, incompleta o desconocida que puede socavar la fiabilidad de los modelos predictivos de ML. Hay dos tipos de incertidumbre: la incertidumbre epistémica se debe a datos limitados e incompletos, mientras que la incertidumbre aleatoria se debe al ruido y la aleatoriedad inherentes a los datos. Para más información, consulte la guía Cuantificación de la incertidumbre en los sistemas de aprendizaje profundo.

tareas indiferenciadas

También conocido como tareas arduas, es el trabajo que es necesario para crear y operar una aplicación, pero que no proporciona un valor directo al usuario final ni proporciona una ventaja competitiva. Algunos ejemplos de tareas indiferenciadas son la adquisición, el mantenimiento y la planificación de la capacidad.

entornos superiores

Ver entorno.

V

succión

Una operación de mantenimiento de bases de datos que implica limpiar después de las actualizaciones incrementales para recuperar espacio de almacenamiento y mejorar el rendimiento.

control de versión

Procesos y herramientas que realizan un seguimiento de los cambios, como los cambios en el código fuente de un repositorio.

Emparejamiento de VPC

Conexión entre dos VPC que permite enrutar el tráfico mediante direcciones IP privadas. Para obtener más información, consulte ¿Qué es una interconexión de VPC? en la documentación de Amazon VPC.

vulnerabilidad

Defecto de software o hardware que pone en peligro la seguridad del sistema.

W

caché caliente

Un búfer caché que contiene datos actuales y relevantes a los que se accede con frecuencia. La instancia de base de datos puede leer desde la caché del búfer, lo que es más rápido que leer desde la memoria principal o el disco.

datos templados

Datos a los que el acceso es infrecuente. Al consultar este tipo de datos, normalmente se aceptan consultas moderadamente lentas.

función de ventana

Función SQL que realiza un cálculo en un grupo de filas que se relacionan de alguna manera con el registro actual. Las funciones de ventana son útiles para procesar tareas, como calcular una media móvil o acceder al valor de las filas en función de la posición relativa de la fila actual.

carga de trabajo

Conjunto de recursos y código que ofrece valor comercial, como una aplicación orientada al cliente o un proceso de backend.

flujo de trabajo

Grupos funcionales de un proyecto de migración que son responsables de un conjunto específico de tareas. Cada flujo de trabajo es independiente, pero respalda a los demás flujos de trabajo del proyecto. Por ejemplo, el flujo de trabajo de la cartera es responsable de priorizar las aplicaciones, planificar las oleadas y recopilar los metadatos de migración. El flujo de trabajo de la cartera entrega estos recursos al flujo de trabajo de migración, que luego migra los servidores y las aplicaciones.

GUSANO

Mira, escribe una vez, lee muchas.

WQF

Consulte el marco de calificación de cargas de trabajo de AWS.

escribe una vez, lee muchas (WORM)

Un modelo de almacenamiento que escribe los datos una sola vez y evita que los datos se eliminen o modifiquen. Los usuarios autorizados pueden leer los datos tantas veces como sea necesario, pero no pueden cambiarlos. Esta infraestructura de almacenamiento de datos se considera inmutable.

Z

ataque de día cero

Un ataque, normalmente de malware, que aprovecha una vulnerabilidad de día cero.

vulnerabilidad de día cero

Un defecto o una vulnerabilidad sin mitigación en un sistema de producción. Los agentes de amenazas pueden usar este tipo de vulnerabilidad para atacar el sistema. Los desarrolladores suelen darse cuenta de la vulnerabilidad a raíz del ataque.

aplicación zombi

Aplicación que utiliza un promedio de CPU y memoria menor al 5 por ciento. En un proyecto de migración, es habitual retirar estas aplicaciones.

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.