CTIC UNI
CENTRO DE TECNOLOGÍAS DE INFORMACIÓN Y COMUNICACIONES
UNIVERSIDAD NACIONAL DE INGENIERÍA

CIENCIA DE DATOS:

APRENDE LOS FUNDAMENTOS DE MANERA PRÁCTICA

SESION 03

MapReduce

Juan Chipoco
mindquasar@gmail.com

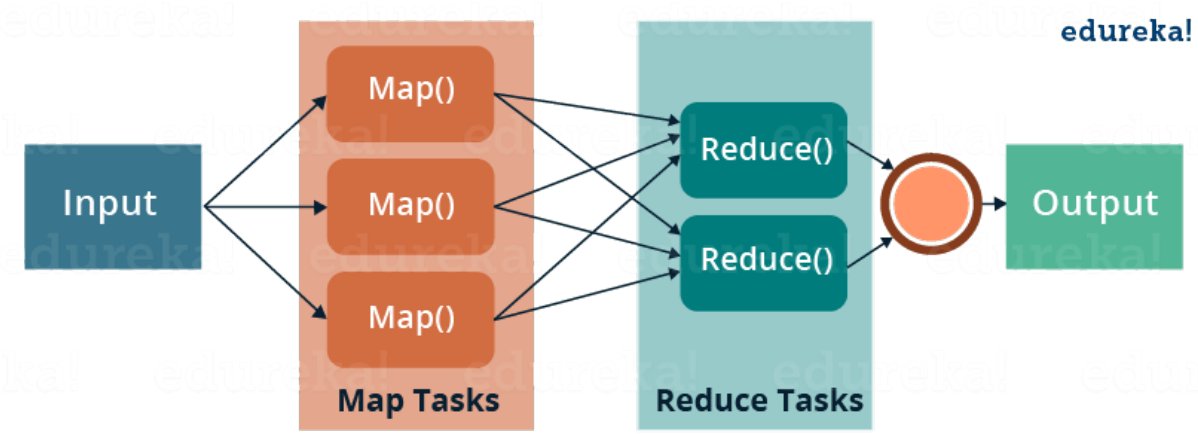
*Las imágenes y graficos no me pertenecen y son usados solo con fines educativos

ÍNDICE

OBJETIVO

MapReduce se define como un modelo de análisis de big data que procesa conjuntos de datos utilizando un algoritmo paralelo en clústeres de computadoras, generalmente clústeres de Apache Hadoop o sistemas en la nube como clústeres de Amazon Elastic MapReduce (EMR). Esta sesion tiene como objetivo explicar el significado de MapReduce, cómo funciona, sus funciones y sus aplicaciones.

MapReduce



Se utiliza un framework y un modelo de programación llamado MapReduce para procesar enormes volúmenes de datos. Map y Reduce son las dos etapas del funcionamiento del programa MapReduce. Se generan grandes volúmenes de datos en el mercado actual basado en datos debido a algoritmos y aplicaciones que recopilan constantemente información sobre personas, empresas, sistemas y organizaciones. La parte difícil es descubrir cómo digerir de manera rápida y efectiva este gran volumen de datos sin perder conclusiones perspicaces.

Solía ocurrir que la única forma de acceder a los datos almacenados en el sistema de archivos distribuidos de Hadoop (HDFS) era usando MapReduce. Ahora se utilizan otros métodos basados en consultas para obtener datos del HDFS mediante comandos similares al lenguaje de consulta estructurado (SQL), como Hive y Pig. Estos, sin embargo, normalmente se ejecutan junto con las tareas creadas con el enfoque de MapReduce.

Esto es así porque MapReduce tiene beneficios únicos. Para acelerar el procesamiento, MapReduce ejecuta la lógica (ilustrada arriba) en el servidor donde ya se encuentran los datos, en lugar de transferir los datos a la ubicación de la aplicación o la lógica.

MapReduce apareció por primera vez como una herramienta para que Google analizara sus resultados de búsqueda. Sin embargo, creció rápidamente en popularidad gracias a su capacidad para dividir y procesar terabytes de datos en paralelo, produciendo resultados más rápidos.

MapReduce es esencial para el funcionamiento del marco Hadoop y un componente central. Mientras que las "tareas de reducción" mezclan y reducen los datos, las "tareas de mapa" se ocupan de separar y mapear los datos. MapReduce facilita el procesamiento simultáneo al dividir petabytes de datos en fragmentos más pequeños y procesarlos en paralelo en los servidores básicos de Hadoop. Al final, recopila toda la información de varios servidores y le da a la aplicación una salida consolidada.

Por ejemplo, consideremos un clúster de Hadoop que consta de 20 000 servidores básicos asequibles que contienen bloques de datos de 256 MB cada uno. Podrá procesar alrededor de cinco terabytes de datos simultáneamente. En comparación con el procesamiento secuencial de

un conjunto de datos tan grande, el uso de MapReduce reduce la cantidad de tiempo necesario para el procesamiento.

Para acelerar el procesamiento, MapReduce elimina la necesidad de transportar datos a la ubicación donde se aloja la aplicación o la lógica. En su lugar, ejecuta la lógica directamente en el servidor que alberga los datos en sí. Tanto el acceso a los datos como su almacenamiento se realizan mediante discos de servidor. Además, los datos de entrada normalmente se guardan en archivos que pueden incluir información organizada, semiestructurada o no estructurada. Finalmente, los datos de salida se guardan de manera similar en forma de archivos.

El principal beneficio de MapReduce es que los usuarios pueden escalar fácilmente el procesamiento de datos en varios nodos informáticos. Las primitivas de procesamiento de datos utilizadas en el modelo MapReduce son mapeadores y reductores. A veces es difícil dividir una aplicación de procesamiento de datos en mapeadores y reductores. Sin embargo, escalar una aplicación para que se ejecute en cientos, miles o decenas de miles de servidores en un clúster es solo una modificación de la configuración después de haberla escrito de la manera MapReduce.

¿Cómo funciona MapReduce?

MapReduce generalmente divide los datos de entrada en partes y las distribuye entre otras computadoras. Los datos de entrada se dividen en pares clave-valor. En los equipos de un clúster, los trabajos de mapas paralelos procesan los datos fragmentados. El trabajo de reducción combina el resultado en una salida de par clave-valor específica, y luego los datos se escriben en el sistema de archivos distribuidos de Hadoop (HDFS).

Por lo general, el programa MapReduce funciona en la misma colección de computadoras que el sistema de archivos distribuidos de Hadoop. El tiempo que lleva realizar una tarea disminuye drásticamente cuando el marco ejecuta un trabajo en los nodos que almacenan los datos. Se utilizaron varios demonios de componentes en la primera iteración de MapReduce, incluidos TaskTrackers y JobTracker.

Los TaskTrackers son agentes instalados en cada máquina del clúster para realizar el mapa y reducir tareas. JobHistory Server es un componente que realiza un seguimiento de los trabajos completados y normalmente se implementa como una función separada o con JobTracker. JobTracker es el nodo maestro que administra todos los trabajos y recursos en un clúster.

Los demonios JobTracker y TaskTracker de versiones anteriores de MapReduce y Hadoop son reemplazados por los componentes ResourceManager y NodeManager de "Yet Another Resource Negotiator" o YARN, que se introdujeron con el lanzamiento de MapReduce y Hadoop versión 2. El proceso de envío de trabajos y la programación en el clúster es manejado por ResourceManager, que está instalado en un nodo maestro. También gestiona la asignación de recursos y el seguimiento de tareas.

NodeManager se ejecuta en nodos esclavos junto con Resource Manager para ejecutar actividades y monitorear la utilización de recursos. NodeManager puede usar otros demonios para ayudar en la ejecución de tareas en el nodo esclavo. MapReduce utiliza tamaños de clúster enormes para ejecutar operaciones paralelas para distribuir datos de entrada y compilar salidas.

Uno puede distribuir trabajos entre prácticamente cualquier cantidad de servidores porque el tamaño del clúster tiene poco impacto en cómo resulta un trabajo de procesamiento.

En consecuencia, la arquitectura de Hadoop en su conjunto y MapReduce simplifican el desarrollo de programas. Muchos lenguajes admiten MapReduce, incluidos C, C++, Java, Ruby, Perl y Python. Para generar tareas sin preocuparse por la coordinación o la comunicación entre nodos, los programadores pueden utilizar las bibliotecas de MapReduce.

Además, MapReduce tolerante a fallas informa regularmente el estado de cada nodo a un nodo maestro. El nodo maestro redistribuye esa tarea a otros nodos de clúster disponibles si un nodo no reacciona como se esperaba. Esto permite la resiliencia y hace viable que MapReduce se ejecute en servidores básicos asequibles.

El programa MapReduce se ejecuta en tres fases: map stage, shuffle stage y la reduce stage.

1. Map Stage

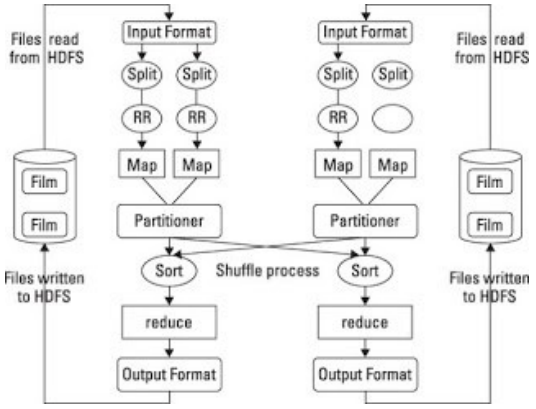
La tarea del mapa o mapeador es procesar los datos de entrada en este nivel. En la mayoría de los casos, los datos de entrada se almacenan en el sistema de archivos de Hadoop como un archivo o directorio (HDFS). La función de mapeador recibe el archivo de entrada línea por línea. El mapeador procesa los datos y produce varios pequeños fragmentos de datos.

2. Reduce Stage (incluye Shuffle y Reduce)

Las etapas de reproducción aleatoria y reducción se combinan para crear la etapa de reducción. El procesamiento de los datos que llegan del mapeador es responsabilidad del reductor. Después del procesamiento, genera un nuevo conjunto de salidas que se mantendrán en el HDFS.

Hadoop asigna las tareas Map y Reduce a las computadoras del clúster adecuadas durante un trabajo de MapReduce. El marco controla todos los aspectos del paso de datos, incluida la asignación de tareas, la confirmación de su finalización y la transferencia de datos entre nodos dentro de un clúster. La mayor parte de la informática se realiza en nodos con datos almacenados localmente en unidades, lo que reduce el tráfico de red. Una vez finalizadas las tareas asignadas, el clúster recopila y reduce los datos para crear los resultados necesarios y luego los devuelve al servidor de Hadoop.

Ejemplo



La unidad de trabajo de nivel superior en MapReduce es un trabajo. Un trabajo normalmente tiene un mapa y una fase de reducción, aunque la fase de reducción se puede omitir. Por ejemplo, considere un trabajo de MapReduce que cuenta la cantidad de veces que se usa cada palabra en un conjunto de documentos. La fase de mapa cuenta las palabras en cada documento, luego la fase de reducción agrega los datos por documento en recuentos de palabras que abarcan toda la colección.

Durante la fase de mapa, los datos de entrada se dividen en divisiones de entrada para el análisis mediante tareas de mapa que se ejecutan en paralelo en el clúster de Hadoop. De forma predeterminada, el marco MapReduce obtiene datos de entrada del sistema de archivos distribuidos de Hadoop (HDFS).

La fase de reducción utiliza los resultados de las tareas de mapa como entrada para un conjunto de tareas de reducción paralelas. Las tareas de reducción consolidan los datos en resultados finales. De forma predeterminada, el marco MapReduce almacena los resultados en HDFS.

Aunque la fase de reducción depende de la salida de la fase de mapa, el procesamiento de mapa y reducción no es necesariamente secuencial. Es decir, las tareas de reducción pueden comenzar tan pronto como se complete cualquier tarea de mapa. No es necesario que se completen todas las tareas del mapa antes de que pueda comenzar cualquier tarea de reducción.

MapReduce opera en pares clave-valor. Conceptualmente, un trabajo de MapReduce toma un conjunto de pares clave-valor de entrada y produce un conjunto de pares clave-valor de salida al pasar los datos a través de funciones map y reduce. Las tareas de mapa producen un conjunto intermedio de pares clave-valor que las tareas de reducción utilizan como entrada. El siguiente diagrama ilustra la progresión de los pares clave-valor de entrada a los pares clave-valor de salida en un nivel alto:

Aunque cada conjunto de pares clave-valor es homogéneo, los pares clave-valor en cada paso no necesitan ser del mismo tipo. Por ejemplo, los pares clave-valor en el conjunto de entrada

(KV1) pueden ser pares (cadena, cadena), con la fase de mapa produciendo pares (cadena, entero) como resultados intermedios (KV2), y la fase de reducción produciendo (entero, cadena) pares para los resultados finales (KV3).

Las claves en los pares de salida del mapa no necesitan ser únicas. Entre el procesamiento del mapa y el procesamiento de reducción, un paso aleatorio ordena todos los valores de salida del mapa con la misma clave en un solo par de entrada de reducción (clave, lista de valores), donde el 'valor' es una lista de todos los valores que comparten la misma clave . Por lo tanto, la entrada a una tarea de reducción es en realidad un conjunto de pares (clave, lista de valores).

Los tipos de clave y valor en cada etapa determinan las interfaces con su mapa y reducen las funciones. Por lo tanto, antes de codificar un trabajo, determine los tipos de datos necesarios en cada etapa del proceso de reducción de mapas. Por ejemplo:

Elija la clave de salida de reducción y los tipos de valores que mejor representen el resultado deseado.

Elija la clave de entrada del mapa y los tipos de valor más adecuados para representar los datos de entrada de los que derivar el resultado final.

Determine la transformación necesaria para pasar de la entrada de mapa a la salida de reducción y elija el tipo de valor de clave de entrada de reducción/salida de mapa intermedio para que coincida.

Controle las características del trabajo de MapReduce a través de las propiedades de configuración. La configuración del trabajo especifica:

Cómo recopilar información

Los tipos de pares clave-valor de entrada y salida para cada etapa

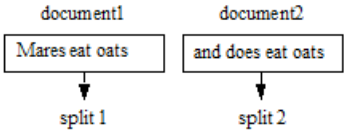
El mapa y reducir funciones

Cómo y dónde almacenar los resultados finales

Ejemplo: Cálculo de ocurrencias de palabras

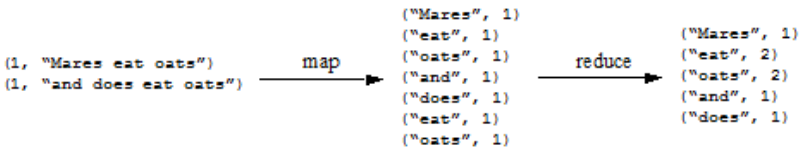
Este ejemplo demuestra el concepto básico de MapReduce calculando el número de ocurrencias de cada palabra en un conjunto de archivos de texto.

Recuerde que los datos de entrada de MapReduce se dividen en divisiones de entrada y las divisiones se dividen a su vez en pares clave-valor de entrada. En este ejemplo, el conjunto de datos de entrada son los dos documentos, documento1 y documento2. La subclase InputFormat divide el conjunto de datos en una división por documento, para un total de 2 divisiones:

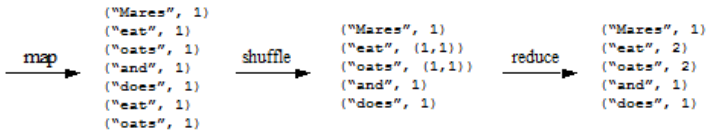


Se genera un par clave-valor (número de línea, texto) para cada línea en un documento de entrada. La función map descarta el número de línea y produce un par por línea (palabra, conteo) para cada palabra en la línea de entrada. La fase de reducción produce pares (palabra, recuento) que representan recuentos de palabras agregados en todos los documentos de entrada.

Dados los datos de entrada que se muestran arriba, la progresión de reducción del mapa para el trabajo de ejemplo es:



El resultado de la fase del mapa contiene varios pares clave-valor con la misma clave: las claves "avena" y "comer" aparecen dos veces. Recuerde que el marco MapReduce consolida todos los valores con la misma clave antes de ingresar a la fase de reducción, por lo que la entrada para reducir es en realidad pares (clave, valores). Por lo tanto, la progresión completa desde la salida del mapa, pasando por la reducción, hasta los resultados finales es:



Comprender el ciclo de vida del trabajo de MapReduce

Ahora describiremos brevemente el ciclo de vida de un trabajo de MapReduce y las funciones de los principales actores en el ciclo de vida. El ciclo de vida completo es mucho más complejo. Para obtener más información, consulte la documentación de su distribución de Hadoop o la documentación de Apache Hadoop MapReduce.

Aunque son posibles otras configuraciones, una configuración de clúster de Hadoop común es un solo nodo maestro donde se ejecuta Job Tracker y varios nodos trabajadores, cada uno de los cuales ejecuta un Task Tracker. El nodo Rastreador de trabajos también puede ser un nodo trabajador.

Cuando el usuario envía un trabajo de MapReduce a Hadoop:

- El Job Client local prepara el trabajo para su envío y lo entrega al Job Tracker.
- El rastreador de trabajos programa el trabajo y distribuye el trabajo del mapa entre los rastreadores de tareas para el procesamiento paralelo.
- Cada rastreador de tareas genera una tarea de mapa. El rastreador de trabajos recibe información de progreso de los rastreadores de tareas.
- A medida que los resultados del mapa están disponibles, Job Tracker distribuye el trabajo reducido entre los Task Trackers para el procesamiento paralelo.

- Cada rastreador de tareas genera una tarea de reducción para realizar el trabajo. El rastreador de trabajos recibe información de progreso de los rastreadores de tareas.
- Todas las tareas de mapa no tienen que completarse antes de que las tareas de reducción comiencen a ejecutarse. Las tareas reducidas pueden comenzar tan pronto como las tareas del mapa comiencen a completarse. Por lo tanto, el mapa y los pasos de reducción a menudo se superponen.

Job Client

Job Client prepara un trabajo para su ejecución. Cuando envía un trabajo de MapReduce a Hadoop, el JobClient local:

Valida la configuración del trabajo.

Genera las divisiones de entrada.

Copia los recursos del trabajo (configuración, archivo JAR del trabajo, divisiones de entrada) en una ubicación compartida, como un directorio HDFS, donde está accesible para el rastreador de trabajos y los rastreadores de tareas.

Envía el trabajo al Job Tracker.

Job Tracker

Job Tracker es responsable de programar trabajos, dividir un trabajo en mapear y reducir tareas, distribuir mapear y reducir tareas entre los nodos trabajadores, la recuperación de fallas de tareas y el seguimiento del estado del trabajo. La programación de trabajos y la recuperación de fallas no se tratan aquí; consulte la documentación de su distribución de Hadoop o la documentación de Apache Hadoop MapReduce.

Al prepararse para ejecutar un trabajo, Job Tracker:

- Obtiene divisiones de entrada de la ubicación compartida donde Job Client colocó la información.
- Crea una tarea de mapa para cada división.
- Asigna cada tarea de mapa a un Rastreador de tareas (nodo trabajador).

El rastreador de trabajos supervisa el estado de los rastreadores de tareas y el progreso del trabajo. A medida que se completan las tareas del mapa y los resultados están disponibles, Job Tracker:

- Crea tareas reducidas hasta el máximo permitido por la configuración del trabajo.

- Asigna cada partición de resultado de mapa a una tarea de reducción.
- Asigna cada tarea de reducción a un rastreador de tareas.

Un trabajo está completo cuando todas las tareas de asignación y reducción se completan con éxito o, si no hay un paso de reducción, cuando todas las tareas de asignación se completan con éxito.

Rastreador de tareas

Un rastreador de tareas administra las tareas de un nodo trabajador e informa el estado al rastreador de trabajos. A menudo, el Rastreador de tareas se ejecuta en el nodo trabajador asociado, pero no es necesario que esté en el mismo host.

Cuando el Rastreador de tareas asigna un mapa o reduce una tarea a un Rastreador de tareas, el Task Tracker:

- Obtiene recursos de trabajo localmente.
- Genera una JVM secundaria en el nodo trabajador para ejecutar el mapa o reducir la tarea.
- Informa el estado al Job Tracker.

La tarea generada por el Rastreador de tareas ejecuta el mapa del trabajo o reduce las funciones.

Map Tracker

El framework Hadoop MapReduce crea una tarea de mapa para procesar cada división de entrada. El map tracker:

- Utiliza InputFormat para obtener los datos de entrada localmente y crear pares clave-valor de entrada.
- Aplica la función de asignación proporcionada por el trabajo a cada par clave-valor.
- Realiza la clasificación local y la agregación de los resultados.
- Si el trabajo incluye un combinador, ejecuta el combinador para una mayor agregación.
- Almacena los resultados localmente, en la memoria y en el sistema de archivos local.
- Comunica el progreso y el estado al Rastreador de tareas.

Los resultados de la tarea Map se someten a una ordenación local por clave para preparar los datos para el consumo mediante tareas reducidas. Si se configura un combinador para el trabajo, también se ejecuta en la tarea de mapa. ACombiner consolida los datos de una manera específica de la aplicación, lo que reduce la cantidad de datos que deben transferirse para reducir las tareas. Por ejemplo, un combinador podría calcular un valor máximo local para una clave y descartar el resto de los valores. Los detalles de cómo las tareas de mapas administran, ordenan

y mezclan los resultados no se tratan aquí. Consulte la documentación de su distribución de Hadoop o la documentación de Apache Hadoop MapReduce.

Cuando una tarea de mapa notifica al Rastreador de tareas que se completó, el Rastreador de tareas notifica al Rastreador de trabajos. El Job Tracker luego pone a disposición los resultados para reducir las tareas.

Reduce Task

La fase de reducción agrega los resultados de la fase de mapa en resultados finales. Por lo general, el conjunto de resultados final es más pequeño que el conjunto de entrada, pero esto depende de la aplicación. La reducción se lleva a cabo mediante tareas de reducción paralelas. No es necesario que las claves y los valores de entrada de reducción tengan el mismo tipo que las claves y los valores de salida.

La fase de reducción es opcional. Puede configurar un trabajo para que se detenga después de que se complete la fase del mapa.

Reducir se lleva a cabo en tres fases, copiar, ordenar y fusionar. Una tarea de reducción:

- Obtiene recursos de trabajo localmente.
- Entra en la fase de copia para obtener copias locales de todos los resultados de mapas asignados de los nodos trabajadores de mapas.
- Cuando se completa la fase de copia, ejecuta la fase de clasificación para fusionar los resultados copiados en un solo conjunto ordenado de pares (clave, lista de valores).
- Cuando se completa la fase de clasificación, ejecuta la fase de reducción, invocando la función de reducción proporcionada por el trabajo en cada par (clave, lista de valores).
- Guarda los resultados finales en el destino de salida, como HDFS.

La entrada a una función de reducción son pares clave-valor donde el valor es una lista de valores que comparten la misma clave. Por ejemplo, si una tarea de mapa produce un par clave-valor ('comer', 2) y otra tarea de mapa produce el par ('comer', 1), estos pares se consolidan en ('comer', (2, 1)) para la entrada a la función de reducción. Si el propósito de la fase de reducción es calcular una suma de todos los valores para cada clave, entonces el par clave-valor de salida final para esta entrada es ('comer', 3).

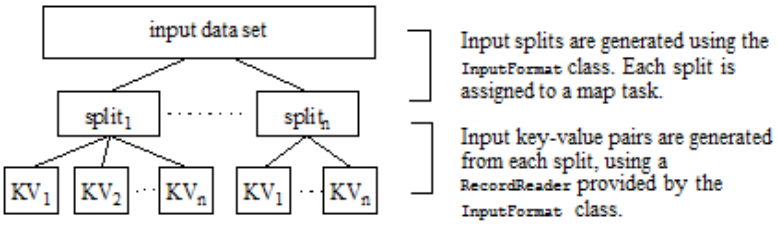
Las tareas de reducción utilizan una subclase OutputFormat para registrar los resultados. La API de Hadoop proporciona subclases de OutputFormat para usar HDFS como destino de salida.

Cómo particiones Hadoop mapean datos de entrada

Cuando envía un trabajo, el marco MapReduce divide el conjunto de datos de entrada en fragmentos llamados divisiones utilizando la subclase

org.apache.hadoop.mapreduce.InputFormat proporcionada en la configuración del trabajo. Las divisiones son creadas por el Job Client local y se incluyen en la información del trabajo disponible para Job Tracker.

JobTracker crea una tarea de mapa para cada división. Cada tarea de mapa utiliza un RecordReader proporcionado por la subclase InputFormat para transformar la división en pares clave-valor de entrada. El siguiente diagrama muestra cómo se desglosan los datos de entrada para su análisis durante la fase del mapa:



La API de Hadoop proporciona subclases de InputFormat para usar HDFS como fuente de entrada.

Caracteristicas de MapReduce

Las siguientes características avanzadas caracterizan a MapReduce:

1. Altamente escalable:

Un marco con excelente escalabilidad es Apache Hadoop MapReduce. Esto se debe a su capacidad para distribuir y almacenar grandes cantidades de datos en numerosos servidores. Todos estos servidores pueden ejecutarse simultáneamente y tienen un precio razonable.

Al agregar servidores al clúster, simplemente podemos aumentar la cantidad de almacenamiento y poder de cómputo. Podemos mejorar la capacidad de los nodos o agregar cualquier número de nodos (escalabilidad horizontal) para lograr un alto poder de cómputo. Las organizaciones pueden ejecutar aplicaciones desde conjuntos masivos de nodos, utilizando potencialmente miles de terabytes de datos, gracias a la programación de Hadoop MapReduce.

2. Versátil

Las empresas pueden utilizar la programación de MapReduce para acceder a nuevas fuentes de datos. Hace posible que las empresas trabajen con muchas formas de datos. Las empresas pueden acceder a datos organizados y no estructurados con este método y adquirir información valiosa de las diversas fuentes de datos.

Dado que Hadoop es un proyecto de código abierto, su código fuente es de libre acceso para revisión, modificaciones y análisis. Esto permite a las empresas modificar el código para satisfacer sus necesidades específicas. El marco MapReduce admite datos de fuentes que incluyen correo electrónico, redes sociales y secuencias de clics en diferentes idiomas.

3. Seguro

El modelo de programación MapReduce utiliza los enfoques de seguridad HBase y HDFS, y solo los usuarios autenticados pueden ver y manipular los datos. HDFS utiliza una técnica de replicación en Hadoop 2 para proporcionar tolerancia a fallas. Según el factor de replicación, realiza un clon de cada bloque en las distintas máquinas. Por lo tanto, se puede acceder a los datos de los otros dispositivos que albergan una réplica de los mismos datos si alguna máquina en un clúster deja de funcionar. La codificación de borrado ha asumido el papel de esta técnica de replicación en Hadoop 3. La codificación de borrado ofrece el mismo nivel de tolerancia a fallas con menos área. La sobrecarga de almacenamiento con la codificación de borrado es inferior al 50 %.

4. Asequibilidad

Con la ayuda del marco de programación MapReduce y el diseño escalable de Hadoop, los grandes volúmenes de datos se pueden almacenar y procesar de manera muy económica. Dicho sistema es particularmente rentable y altamente escalable, lo que lo hace ideal para modelos comerciales que deben almacenar datos en constante expansión para satisfacer las demandas del presente.

En términos de escalabilidad, el procesamiento de datos con sistemas de gestión de bases de datos relacionales convencionales más antiguos no era tan sencillo como con el sistema Hadoop. En estas situaciones, la empresa tuvo que minimizar los datos y ejecutar una clasificación basada en presunciones sobre cómo los datos específicos podrían ser relevantes para la organización, por lo que eliminó los datos sin procesar. El modelo de programación MapReduce en la arquitectura de escalamiento horizontal de Hadoop ayuda en esta situación.

5. De ritmo rápido

El sistema de archivos distribuidos de Hadoop, una técnica de almacenamiento distribuido utilizada por MapReduce, es un sistema de mapeo para encontrar datos en un clúster. Las tecnologías de procesamiento de datos, como la programación de MapReduce, generalmente se colocan en los mismos servidores que permiten un procesamiento de datos más rápido.

Gracias al almacenamiento de datos distribuido de Hadoop, los usuarios pueden procesar datos de manera distribuida en un grupo de nodos. Como resultado, le da a la arquitectura Hadoop la capacidad de procesar datos con una rapidez excepcional. Hadoop MapReduce puede procesar datos no estructurados o semiestructurados en grandes cantidades en menos tiempo.

6. Basado en un modelo de programación simple

Hadoop MapReduce se basa en un modelo de programación sencillo y es una de las muchas características notables de la tecnología. Esto permite a los programadores crear aplicaciones MapReduce que pueden manejar tareas de forma rápida y eficaz. Java es un lenguaje de programación muy apreciado y fácil de aprender que se utiliza para desarrollar el modelo de programación MapReduce.

La programación Java es fácil de aprender y cualquiera puede crear un modelo de procesamiento de datos que funcione para su empresa. Hadoop es fácil de usar porque los clientes no necesitan preocuparse por la distribución informática. El marco mismo hace el procesamiento.

7. Compatible con procesamiento paralelo

El procesamiento paralelo involucrado en la programación de MapReduce es uno de sus componentes clave. Las tareas se dividen en el paradigma de programación para permitir la ejecución simultánea de actividades independientes. Como resultado, el programa se ejecuta más rápido debido al procesamiento en paralelo, lo que facilita que los procesos manejen cada trabajo. Múltiples procesadores pueden llevar a cabo estas tareas desglosadas gracias al procesamiento paralelo. En consecuencia, todo el software se ejecuta más rápido.

8. Confiable

El mismo conjunto de datos se transfiere a algunos otros nodos en un clúster cada vez que se envía una recopilación de información a un solo nodo. Por lo tanto, incluso si falla un nodo, las copias de seguridad siempre están disponibles en otros nodos que aún pueden recuperarse cuando sea necesario. Esto asegura una alta disponibilidad de datos.

El marco ofrece una forma de garantizar la confiabilidad de los datos mediante el uso de los módulos Block Scanner, Volume Scanner, Disk Checker y Directory Scanner. Sus datos se guardan de forma segura en el clúster y se puede acceder a ellos desde otra máquina que tenga una copia de los datos si su dispositivo falla o los datos se corrompen.

9. Altamente disponible

La función de tolerancia a fallas de Hadoop garantiza que incluso si uno de los DataNodes falla, el usuario aún puede acceder a los datos de otros DataNodes que tienen copias de él. Además, el clúster de Hadoop de alta accesibilidad consta de dos o más NameNodes activos y pasivos que se ejecutan en espera activa. El NameNode activo es el nodo activo. Un nodo pasivo es un nodo de respaldo que aplica los cambios realizados en los registros de edición de NameNode activos a su espacio de nombres.

Uso de MapReduce

1. Entretenimiento

Hadoop MapReduce ayuda a los usuarios finales a encontrar las películas más populares según sus preferencias y su historial de visualización anterior. Se concentra principalmente en sus clics y registros.

Varios servicios OTT, incluido Netflix, lanzan regularmente muchas series web y películas. Es posible que te haya pasado que no podías elegir qué película ver, así que miraste las recomendaciones de Netflix y decidiste ver una de las series o películas sugeridas. Netflix usa Hadoop y MapReduce para indicarle al usuario algunas películas conocidas en función de lo que ha visto y las películas que disfruta. MapReduce puede examinar los registros y los clics de los usuarios para saber cómo ven películas.

2. Comercio electrónico

Varias empresas de comercio electrónico, incluidas Flipkart, Amazon y eBay, emplean MapReduce para evaluar los patrones de compra de los consumidores en función de los intereses de los clientes o los patrones de compra históricos. Para varias empresas de comercio electrónico, proporciona métodos de sugerencia de productos mediante el análisis de datos, el historial de compras y los registros de interacción del usuario.

Muchos proveedores de comercio electrónico utilizan el modelo de programación MapReduce para identificar productos populares según las preferencias de los clientes o el comportamiento de compra. Hacer propuestas de artículos para el inventario de comercio electrónico es parte de esto, al igual que mirar los registros del sitio web, los historiales de compras, los registros de interacción del usuario, etc., para obtener recomendaciones de productos.

3. Redes sociales

Cerca de 500 millones de tweets, o alrededor de 3000 por segundo, se envían diariamente en la plataforma de microblogging Twitter. MapReduce procesa datos de Twitter, realizando operaciones como tokenización, filtrado, conteo y agregación de contadores.

> *Tokenización*: crea pares clave-valor a partir de los tweets tokenizados al mapear los tweets como mapas de tokens.
> *Filtrado*: Los términos que no se desean se eliminan de los mapas de tokens.
> *Conteo*: crea un contador de fichas para cada palabra en el conteo.
> *Contadores agregados*: una agrupación de valores de contadores comparables se prepara en piezas pequeñas y manejables utilizando contadores agregados.

4. Almacén de datos

Los sistemas que manejan enormes volúmenes de información se conocen como sistemas de almacenamiento de datos. El esquema en estrella, que consta de una tabla de hechos y varias tablas de dimensiones, es el modelo de almacén de datos más popular. En una arquitectura de nada compartido, es imposible almacenar todos los datos necesarios en un solo nodo, por lo que es esencial recuperar datos de otros nodos.

Esto da como resultado la congestión de la red y velocidades lentas de ejecución de consultas. Si las dimensiones no son demasiado grandes, los usuarios pueden replicarlas en los nodos para solucionar este problema y maximizar el paralelismo. Con MapReduce, podemos crear una lógica empresarial especializada para la comprensión de los datos mientras analizamos enormes volúmenes de datos en los almacenes de datos.

5. Detección de fraude

Los métodos convencionales de prevención del fraude no siempre son muy efectivos. Por ejemplo, los analistas de datos generalmente administran pagos inexactos auditando una pequeña muestra de reclamos y solicitando registros médicos de remitentes específicos. Hadoop es un sistema muy adecuado para manejar grandes volúmenes de datos necesarios para crear algoritmos de detección de fraude. Las empresas financieras, incluidos los bancos, las compañías de seguros y los lugares de pago, utilizan Hadoop y MapReduce para la detección de fraudes, la evidencia de reconocimiento de patrones y el análisis comercial a través del análisis de transacciones.