El modelo de computación distribuida de Hadoop
Sistemas Operativos - 2024-2

León Gómez Erick, **Martínez Jiménez Israel**

Resumen: Hadoop se ha establecido como una plataforma fundamental en el campo del cómputo distribuido, permitiendo el procesamiento y almacenamiento de grandes volúmenes de datos de manera eficiente y escalable. Este escrito explora la arquitectura integral de Hadoop, incluyendo su sistema de archivos distribuido (HDFS), el modelo de procesamiento MapReduce y el gestor de recursos YARN, que juntos facilitan una gestión robusta y flexible de recursos en entornos de clúster.

Palabras clave: Hadoop, Cómputo distribuido, HDFS (Hadoop Distributed File System), MapReduce, YARN (Yet Another Resource Negotiator), Big Data, Clúster de datos, Escalabilidad, Gestión de recursos

1. INTRODUCCIÓN

Apache Hadoop es una plataforma que facilita el manejo de enormes cantidades de datos utilizando clústeres y un modelo de programación sencillo. Escrito en Java, este framework permite el desarrollo de aplicaciones distribuidas que manejan grandes volúmenes de datos y requieren alta escalabilidad. Este proyecto está gestionado por la Apache Software Foundation y ofrece una solución ideal para programadores que no están familiarizados con el desarrollo en ambientes distribuidos, ya que abstrae la complejidad asociada con la paralelización de tareas, la gestión de procesos, el balanceo de carga y la tolerancia a fallos.

Hadoop cuenta con el apoyo de contribuyentes de grandes empresas tecnológicas y fue inicialmente adoptado por líderes de la industria como Yahoo y Facebook. Hoy en día, se utiliza ampliamente en sectores como finanzas, tecnología, telecomunicaciones, medios de comunicación y entretenimiento, gobierno y centros de investigación, entre otros, que generan grandes volúmenes de datos. Las empresas utilizan Hadoop para realizar análisis complejos y personalizados que se adaptan a sus necesidades específicas.

Doug Cutting, el creador de Apache Lucene, una popular biblioteca de búsqueda de texto, también desarrolló Hadoop, que surgió de Apache Nutch, un motor de búsqueda de código abierto que forma parte del proyecto Lucene. La idea de Hadoop fue inspirada por un artículo publicado por Google en 2004, que introdujo el concepto de MapReduce. A principios de 2005, los desarrolladores de Nutch ya habían implementado MapReduce en Nutch, y hacia mediados de ese año, todos los algoritmos principales de Nutch se ejecutaban utilizando MapReduce y NDFS (sistema de archivos de Nutch).

Estos desarrollos trascendieron el ámbito de la búsqueda, llevando a que en febrero de 2006, MapReduce y NDFS se separaran de Nutch para convertirse en un subproyecto independiente de Lucene, denominado Hadoop. Doug Cutting se unió a Yahoo! poco después, proporcionando a Hadoop los recursos y el equipo especializado necesarios para su éxito. En enero de 2008, Hadoop fue reconocido como un proyecto independiente de alto nivel en Apache, reflejando su éxito y el apoyo de una comunidad diversa y activa. Desde entonces, Hadoop ha sido adoptado por muchas otras empresas, incluyendo Last.fm, Facebook y el New York Times, demostrando su utilidad en proyectos que requieren escalabilidad y capacidad de acceso rápido y paralelo a datos distribuidos, con un robusto sistema de seguridad integrado. *(Dean, J. y Ghemawat, S., 2004)*

2. CÓMPUTO DISTRIBUIDO

La computación distribuida es un método que consiste en hacer que varias computadoras trabajen juntas para resolver un problema común. De este modo, una red de computadoras se unen en una única entidad potente que ofrece recursos a gran escala para abordar desafíos complejos. Los protocolos o reglas de comunicación crean una dependencia entre los componentes del sistema distribuido. Esta interdependencia se llama acoplamiento, y hay dos tipos principales de este.

- *Acoplamiento flexible*
  Los componentes están débilmente conectados de manera que los cambios en uno de ellos no afecten al otro. Por ejemplo, las computadoras del cliente y del servidor pueden estar débilmente acopladas temporalmente. Las peticiones del cliente se agregan a una cola del servidor, permitiendo que el cliente continúe con otras funciones hasta que el servidor responda a su mensaje.

- *Acoplamiento ajustado*
  Los sistemas distribuidos de alto rendimiento suelen utilizar un acoplamiento estrecho. Las redes de área local rápidas conectan varias computadoras, formando así un clúster. En la computación en clúster, cada computadora se configura para realizar la misma ta-

rea. Los sistemas de control central, conocidos como middleware de agrupación, supervisan y programan las tareas, además de coordinar la comunicación entre las diferentes computadoras.

2.1 Ventajas

- *Escalabilidad:*
  Los sistemas distribuidos pueden escalar según su carga de trabajo y sus necesidades. Es posible agregar nuevos nodos, es decir, más dispositivos computadoras, a la red distribuida cuando esto sea necesario.

- *Tolerancia a fallos:*
  El sistema de computación distribuida no se colapsará si una de las computadoras falla. Se tiene tolerancia a fallos, ya que puede seguir funcionando incluso si algunas computadoras individuales presentan fallas.

- *Consistencia:*
  En un sistema distribuido, las computadoras comparten información y duplican datos entre sí, pero el sistema administra automáticamente la coherencia de los datos en todas las computadoras. De esta manera, se obtiene el beneficio de la tolerancia a fallos sin comprometer la coherencia de los datos.

- *Abstracción:*
  El cómputo distribuido ofrece una separación lógica entre el usuario y los dispositivos físicos. Se puede interactuar con el sistema como si fuera una sola computadora, sin preocuparse por la instalación y configuración de las máquinas individuales. Este enfoque permite integrar hardware, software y sistemas operativos diferentes para que el sistema funcione de manera integrada y sin problemas.

- *Eficiencia:*
  El cómputo distribuido ofrece un rendimiento más rápido al utilizar de manera óptima los recursos del hardware subyacente. Como resultado, se puede gestionar cualquier carga de trabajo sin preocuparse por fallos del sistema debido a picos de volumen.

3. ARQUITECTURA DE HADOOP

El Big data, con su inmenso volumen y estructuras de datos variadas, ha abrumado a los marcos y herramientas de redes tradicionales. Utilizar hardware de alto rendimiento y servidores especializados puede ayudar, pero son inflexibles y vienen con un precio considerable.

Hadoop logra procesar y almacenar grandes cantidades de datos utilizando hardware común y asequible interconectado. Cientos o incluso miles de servidores dedicados de bajo costo trabajan juntos para almacenar y procesar datos dentro de un único ecosistema.

El Sistema de Archivos Distribuidos de Hadoop (HDFS), YARN y MapReduce están en el corazón de ese ecosistema. HDFS es un conjunto de protocolos utilizados para almacenar grandes conjuntos de datos, mientras que MapReduce procesa eficientemente los datos entrantes.

Un clúster de Hadoop consta de uno o varios nodos maestros y muchos más nodos esclavos. HDFS y MapReduce forman una base flexible que puede escalar linealmente agregando nodos adicionales.

Yet Another Resource Negotiator (YARN) fue creado para mejorar la gestión de recursos y los procesos de programación en un clúster de Hadoop. La introducción de YARN, con su interfaz genérica, abrió la puerta para que otras herramientas de procesamiento de datos se incorporaran al ecosistema de Hadoop.

Un conjunto de software ampliado, con HDFS, YARN y MapReduce en su núcleo, hace que Hadoop sea la solución ideal para procesar big data.Hadoop puede ser dividido en 4 capas.

- **Capa de Almacenamiento Distribuido**
  Cada nodo en un clúster de Hadoop tiene su propio espacio en disco, memoria, ancho de banda y procesamiento. Los datos entrantes se dividen en bloques de datos individuales, que luego se almacenan dentro de la capa de almacenamiento distribuido de HDFS. HDFS asume que cada unidad de disco y nodo esclavo dentro del clúster es poco confiable. Como precaución, HDFS almacena tres copias de cada conjunto de datos a lo largo del clúster. El nodo maestro de HDFS (NameNode) mantiene los metadatos para el bloque de datos individual y todas sus réplicas.
- **Gestión de Recursos del Clúster**
  Hadoop necesita coordinar los nodos perfectamente para que innumerables aplicaciones y usuarios compartan efectivamente sus recursos. Inicialmente, MapReduce manejaba tanto la gestión de recursos como el procesamiento de datos. YARN separa estas dos funciones. Como la herramienta de gestión de recursos de facto para Hadoop, YARN ahora puede asignar recursos a diferentes marcos escritos para Hadoop. Estos incluyen proyectos como Apache Pig, Hive, Giraph, Zookeeper, así como el propio MapReduce.
- **Capa de Marco de Procesamiento**
  La capa de procesamiento consiste en marcos que analizan y procesan los conjuntos de datos que llegan al clúster. Los conjuntos de datos estructurados y no estructurados se mapean, barajan, ordenan, fusionan y reducen a bloques de datos más pequeños y manejables. Estas operaciones se distribuyen en múltiples nodos lo más cerca posible de los servidores donde se encuentra el dato. Marcos de computación como Spark, Storm y Tez ahora permiten el procesamiento en tiempo real, el procesamiento de consultas interactivas y otras opciones de programación que ayudan al motor de MapReduce y utilizan HDFS de manera mucho más eficiente.
- **Interfaz de Programación de Aplicaciones**
  La introducción de YARN en Hadoop 2 ha llevado a la creación de nuevos marcos de procesamiento y APIs. Los big data continúan expandiéndose y la variedad de herramientas necesita seguir ese crecimiento. Proyectos que se centran en plataformas de búsqueda, transmisión de datos, interfaces amigables para el usuario, lenguajes de programación, mensajería, tolerancia a fallos y seguridad, son todos parte integral de un ecosistema de Hadoop comprensivo.

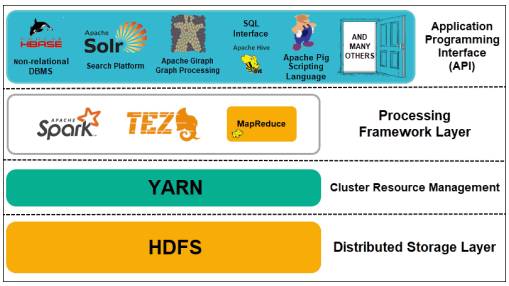

Figura 1. Ecosistema Hadoop

4. HDFS (HADOOP DISTRIBUTED FILE SYSTEM)

4.1 Introducción

El Sistema de Archivos Distribuido de Hadoop (HDFS) es un sistema de archivos distribuido diseñado para ejecutarse en hardware de tipo commodity. Tiene muchas similitudes con los sistemas de archivos distribuidos existentes. Sin embargo, las diferencias con otros sistemas de archivos distribuidos son significativas. HDFS es altamente tolerante a fallos y está diseñado para ser implementado en hardware de bajo costo. HDFS proporciona acceso de alto rendimiento a los datos de la aplicación y es adecuado para aplicaciones que manejan conjuntos de datos grandes. HDFS fue originalmente construido como infraestructura para el proyecto de motor de búsqueda web Apache Nutch y forma parte del proyecto principal de Apache Hadoop Core.

4.2 Supuestos y Objetivos de HDFS

Fallo de Hardware El fallo de hardware es la norma en lugar de la excepción. Una instancia de HDFS puede consistir en cientos o miles de máquinas servidoras, cada una almacenando parte de los datos del sistema de archivos. El hecho de que haya un gran número de componentes y que cada componente tenga una probabilidad no trivial de fallo significa que siempre habrá algún componente de HDFS que no funcione. Por lo tanto, la detección de fallos y la recuperación rápida y automática de los mismos es un objetivo arquitectónico fundamental de HDFS.

Acceso de Datos en Streaming Las aplicaciones que se ejecutan en HDFS necesitan acceso en streaming a sus conjuntos de datos. No son aplicaciones de propósito general que típicamente se ejecutan en sistemas de archivos de propósito general. HDFS está diseñado más para procesamiento por lotes que para uso interactivo por parte de los usuarios. El énfasis está en el alto rendimiento del acceso a datos en lugar de la baja latencia del acceso a datos.

Conjuntos de Datos Grandes Las aplicaciones que se ejecutan en HDFS manejan conjuntos de datos grandes. Un archivo típico en HDFS tiene un tamaño de gigabytes a terabytes. Por lo tanto, HDFS está optimizado para admitir archivos grandes. Debe proporcionar un alto ancho de banda y escalar a cientos de nodos en un solo clúster. Debe admitir decenas de millones de archivos en una sola instancia.

Modelo de Coherencia Simple Las aplicaciones de HDFS necesitan un modelo de acceso de escritura-una-vez-lectura-muchas veces para los archivos. Un archivo, una vez creado, escrito y cerrado, no necesita ser modificado excepto para añadidos y truncados. El añadido de contenido al final de los archivos está soportado pero no se puede actualizar en un punto arbitrario. Esta suposición simplifica los problemas de coherencia de datos y permite un alto rendimiento en el acceso a datos. Una aplicación de MapReduce o un rastreador web encaja perfectamente con este modelo.

Mover la computación es más barato que mover los datos Una computación solicitada por una aplicación es mucho más eficiente si se ejecuta cerca de los datos con los que opera. Esto es especialmente cierto cuando el tamaño del conjunto de datos es enorme. Esto minimiza la congestión de red y aumenta el rendimiento general del sistema. La suposición es que a menudo es mejor migrar la computación más cerca de donde están ubicados los datos que mover los datos a donde se está ejecutando la aplicación. HDFS proporciona interfaces para que las aplicaciones se muevan más cerca de donde están ubicados los datos.

Portabilidad HDFS fue diseñado para ser fácilmente portable de una plataforma a otra. Esto facilita la adopción generalizada de HDFS como plataforma de elección para un amplio conjunto de aplicaciones.

4.3 NameNodes y DataNodes

HDFS tiene una arquitectura maestro/esclavo. Un clúster de HDFS consiste en un único NameNode, un servidor maestro que gestiona el espacio de nombres del sistema de archivos y regula el acceso a los archivos por parte de los clientes. Además, hay una serie de DataNodes, generalmente uno por nodo en el clúster, que gestionan el almacenamiento adjunto a los nodos en los que se ejecutan. HDFS expone un espacio de nombres del sistema de archivos y permite que los datos de usuario se almacenen en archivos. Internamente, un archivo se divide en uno o más bloques y estos bloques se almacenan en un conjunto de DataNodes. El NameNode ejecuta operaciones de espacio de nombres del sistema de archivos como la apertura, cierre y renombrado de archivos y directorios. También determina la asignación de bloques a DataNodes. Los DataNodes son responsables de servir las solicitudes de lectura y escritura de los clientes del sistema de archivos. Los DataNodes también realizan la creación, eliminación y replicación de bloques bajo instrucciones del NameNode. La existencia de un único NameNode en un clúster simplifica enormemente la arquitectura del sistema. El NameNode es el árbitro y repositorio de todos los metadatos de HDFS. El sistema está diseñado de tal manera que los datos de usuario nunca pasan a través del NameNode.
Ver figura 2.

4.4 Replicación de Datos

HDFS está diseñado para almacenar de manera confiable archivos muy grandes en varias máquinas dentro de un clúster extenso. Almacena cada archivo como una secuencia de bloques. Los bloques de un archivo se replican para

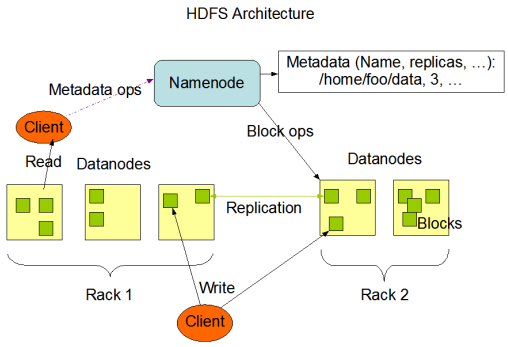

Figura 2. Arquitectura de HDFS

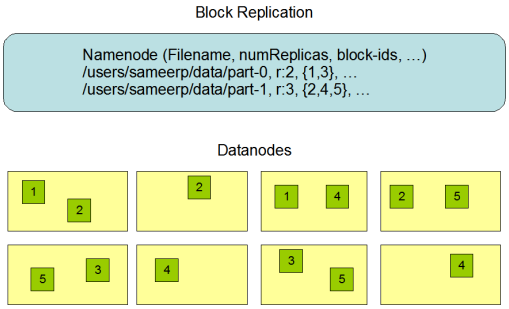

Figura 3. Replicación de bloques y DataNodes

tolerancia a fallos. El tamaño de bloque y el factor de replicación son configurables por archivo.

Todos los bloques en un archivo excepto el último bloque tienen el mismo tamaño, aunque que los usuarios pueden iniciar un nuevo bloque sin llenar el último al tamaño de bloque configurado.

Una aplicación puede especificar el número de réplicas de un archivo. El factor de replicación se puede especificar al momento de la creación del archivo y se puede cambiar más tarde. Los archivos en HDFS son de escritura única (excepto para agregados y truncados) y tienen estrictamente un único escritor en cualquier momento.

El NameNode toma todas las decisiones con respecto a la replicación de bloques. Recibe periódicamente un 'Heatbeat' y un Informe de Bloques (Blockreport) de cada uno de los DataNodes en el clúster. La recepción de un Latido implica que el DataNode está funcionando correctamente y un Blockreport contiene una lista de todos los bloques en un DataNode. Ver Figura 3.

Colocación de Réplicas La colocación de réplicas es fundamental para la confiabilidad y el rendimiento de HDFS. Optimizar la colocación de réplicas distingue a HDFS de la mayoría de los otros sistemas de archivos distribuidos. Esta es una característica que requiere mucho ajuste y experiencia. El propósito de una política de colocación de réplicas consciente de la disposición de racks es mejorar la confiabilidad de los datos, la disponibilidad y la utilización del ancho de banda de red.

Las grandes instancias de HDFS se ejecutan en un clúster de computadoras que comúnmente se distribuyen en muchos racks. La comunicación entre dos nodos en racks diferentes debe pasar por switches. En la mayoría de los casos, el ancho de banda de red entre máquinas en el mismo rack es mayor que entre máquinas en racks diferentes.

El NameNode determina el ID del rack al que pertenece cada DataNode a través de la conciencia de Racks de Hadoop. Una política simple pero no óptima es colocar réplicas en racks únicos. Esto evita la pérdida de datos cuando falla un rack completo y permite utilizar el ancho de banda de múltiples racks al leer datos. Esta política distribuye uniformemente las réplicas en el clúster, lo que facilita equilibrar la carga en caso de fallo de componentes. Sin embargo, esta política aumenta el costo de escritura porque una escritura necesita transferir bloques a múltiples racks.

Normalmente el factor de replicación es de tres, en este caso la política de colocación de HDFS es colocar una réplica en la máquina local si el escritor está en un DataNode; de lo contrario, en un DataNode aleatorio en el mismo rack que el del escritor, otra réplica en un nodo en un rack diferente (remoto), y la última en un nodo diferente en el mismo rack remoto. Esta política reduce el tráfico de escritura entre racks, lo que generalmente mejora el rendimiento de escritura. La probabilidad de falla de un rack es mucho menor que la de un nodo; esta política no afecta las garantías de confiabilidad y disponibilidad de los datos. Sin embargo, no reduce el ancho de banda de red agregado utilizado al leer datos, ya que un bloque se coloca en solo dos racks únicos en lugar de tres. Con esta política, las réplicas de un bloque no se distribuyen uniformemente entre los racks. Dos réplicas están en nodos diferentes de un rack y la réplica restante está en un nodo de uno de los otros racks. Esta política mejora el rendimiento de escritura sin comprometer la confiabilidad de los datos o el rendimiento de lectura.

Debido a que el NameNode no permite que los DataNodes tengan múltiples réplicas del mismo bloque, el número máximo de réplicas creadas es el número total de DataNodes en ese momento.

Selección de replicas Para minimizar el consumo global de ancho de banda y la latencia de lectura, HDFS intenta satisfacer una solicitud de lectura desde una réplica que esté más cerca del lector. Si existe una réplica en el mismo rack que el nodo lector, entonces esa réplica se prefiere para satisfacer la solicitud de lectura. Si el clúster de HDFS abarca varios centros de datos, entonces se prefiere una réplica que resida en el centro de datos local sobre cualquier réplica remota.

4.5 Persistencia de Metadatos del Sistema de Archivos

El espacio de nombres de HDFS es almacenado por el NameNode. El NameNode utiliza un registro de transacciones llamado EditLog para registrar de forma persistente cada cambio que ocurre en los metadatos del sistema de archivos. Por ejemplo, crear un archivo nuevo en HDFS hace que el NameNode inserte un registro en el EditLog indicando esto. De manera similar, cambiar el factor de replicación de un archivo provoca la inserción de un nuevo registro en el EditLog. El NameNode utiliza un archivo en el sistema de archivos local del host para almacenar el EditLog. Todo el espacio de nombres del sistema de archivos, incluyendo la asignación de bloques a archivos

y propiedades del sistema de archivos, se almacena en un archivo llamado FsImage. El FsImage también se guarda como un archivo en el sistema de archivos local del NameNode.

El NameNode mantiene una imagen de todo el espacio de nombres del sistema de archivos y el mapeo de bloques de archivos en la memoria. Cuando el NameNode se inicia, o se activa un punto de control por un umbral configurable, lee el FsImage y el EditLog del disco, aplica todas las transacciones del EditLog a la representación en memoria del FsImage, y guarda esta nueva versión en un nuevo FsImage en el disco. Luego puede truncar el antiguo EditLog porque sus transacciones se han aplicado al FsImage persistente. Este proceso se llama un punto de control. El propósito de un punto de control es asegurarse de que HDFS tenga una vista consistente de los metadatos del sistema de archivos al tomar una instantánea de los metadatos del sistema de archivos y guardarla en FsImage. Aunque es eficiente leer un FsImage, no es eficiente realizar ediciones incrementales directamente en un FsImage. En lugar de modificar el FsImage para cada edición, persistimos las ediciones en el EditLog. Durante el punto de control, los cambios del EditLog se aplican al FsImage.

El DataNode almacena los datos de HDFS en archivos en su sistema de archivos local. El DataNode no tiene conocimiento sobre los archivos de HDFS. Almacena cada bloque de datos de HDFS en un archivo separado en su sistema de archivos local. El DataNode no crea todos los archivos en el mismo directorio. En su lugar, utiliza una heurística para determinar el número óptimo de archivos por directorio y crea subdirectorios adecuadamente. No es óptimo crear todos los archivos locales en el mismo directorio porque es posible que el sistema de archivos local no pueda soportar eficientemente un gran número de archivos en un solo directorio. Cuando un DataNode se inicia, escanea su sistema de archivos local, genera una lista de todos los bloques de datos de HDFS que corresponden a cada uno de estos archivos locales, y envía este informe al NameNode. Este informe se llama Blockreport.

4.6 Robusticidad

El objetivo principal de HDFS es almacenar datos de manera confiable incluso en presencia de fallos. Los tres tipos comunes de fallos son fallos del NameNode, fallos del DataNode y particiones de red.

Fallo de Disco de Datos, 'Heartbeats' y Re-Replicación Cada DataNode envía periódicamente un mensaje de latido (Heartbeat) al NameNode. Una partición de red puede hacer que un subconjunto de DataNodes pierda conectividad con el NameNode. El NameNode detecta esta condición por la ausencia de un mensaje de 'Heartbeat'. El NameNode marca los DataNodes sin 'Heartbeats' recientes como muertos y no reenvía ninguna solicitud de E/S nueva a ellos. Cualquier dato registrado en un DataNode muerto ya no está disponible para HDFS. La muerte de un DataNode puede hacer que el factor de replicación de algunos bloques caiga por debajo de su valor especificado. El NameNode sigue constantemente qué bloques necesitan ser replicados e inicia la replicación cuando sea necesario. La necesidad de re-replicación puede surgir por muchas razones: un DataNode puede volverse no disponible, una réplica puede corromperse, un disco duro en un DataNode puede fallar o el factor de replicación de un archivo puede aumentar.

El tiempo de espera para marcar los DataNodes como muertos es conservadoramente largo (más de 10 minutos por defecto) para evitar una tormenta de replicación causada por el cambio de estado intermitente de los DataNodes. Los usuarios pueden establecer un intervalo más corto para marcar los DataNodes como obsoletos y evitar nodos obsoletos en lectura y/o escritura mediante configuración para cargas de trabajo sensibles al rendimiento.

Rebalanceo del Clúster La arquitectura de HDFS es compatible con esquemas de rebalanceo de datos. Un esquema podría mover automáticamente datos de un DataNode a otro si el espacio libre en un DataNode cae por debajo de un cierto umbral. En caso de una demanda repentina alta para un archivo en particular, un esquema podría crear dinámicamente réplicas adicionales y rebalancear otros datos en el clúster.

Integridad de los datos Es posible que un bloque de datos recuperado de un DataNode llegue corrupto. Esta corrupción puede ocurrir debido a fallas en un dispositivo de almacenamiento, fallas en la red o software con errores. El software cliente de HDFS implementa la verificación de sumas de comprobación (checksum) en el contenido de los archivos de HDFS. Cuando un cliente crea un archivo de HDFS, calcula una suma de comprobación para cada bloque del archivo y almacena estas sumas de comprobación en un archivo oculto separado en el mismo espacio de nombres de HDFS. Cuando un cliente recupera el contenido de un archivo, verifica que los datos recibidos de cada DataNode coincidan con la suma de comprobación almacenada en el archivo de suma de comprobación asociado. Si no coinciden, entonces el cliente puede optar por recuperar ese bloque de otro DataNode que tenga una réplica de ese bloque.

Fallo del Disco de Metadatos El FsImage y el EditLog son estructuras de datos centrales de HDFS. Una corrupción de estos archivos puede hacer que la instancia de HDFS no funcione correctamente. Por esta razón, el NameNode puede configurarse para mantener múltiples copias del FsImage y EditLog. Cualquier actualización en el FsImage o EditLog provoca que cada una de las copias de FsImage y EditLog se actualicen de manera síncrona. Esta actualización síncrona de múltiples copias del FsImage y EditLog puede degradar la tasa de transacciones por segundo que un NameNode puede admitir. Sin embargo, esta degradación es aceptable porque aunque las aplicaciones de HDFS son muy intensivas en datos, no lo son tanto en metadatos. Cuando un NameNode se reinicia, selecciona el FsImage y EditLog consistentes más recientes para usar.

5. MAPREDUCE

Hadoop MapReduce es un marco de software para escribir fácilmente aplicaciones que procesan vastas cantidades de datos (conjuntos de datos de varios terabytes) en paralelo

en grandes clústeres (miles de nodos) de hardware estándar de manera confiable y tolerante a fallos.

Un trabajo de MapReduce generalmente divide el conjunto de datos de entrada en fragmentos independientes que son procesados por las tareas de map en un proceso completamente paralelo. El marco ordena las salidas de los mapas, que luego son ingresadas a las tareas de reduce. Típicamente, tanto la entrada como la salida del trabajo se almacenan en un sistema de archivos. El marco se encarga de programar las tareas, monitorearlas y reejecutar las tareas fallidas.

Normalmente, los nodos de cómputo y los nodos de almacenamiento son los mismos, es decir, el marco de MapReduce y el Sistema de Archivos Distribuido de Hadoop se ejecutan en el mismo conjunto de nodos. Esta configuración permite al marco programar efectivamente tareas en los nodos donde los datos ya están presentes, lo que resulta en un ancho de banda agregado muy alto en todo el clúster.

El marco de MapReduce consiste en un solo maestro ResourceManager, un NodeManager por nodo de clúster, y un MRAppMaster por aplicación.

Como mínimo, las aplicaciones especifican las ubicaciones de entrada/salida y suministran funciones de map y reduce mediante implementaciones de interfaces y/o clases abstractas apropiadas. Estos, y otros parámetros del trabajo, componen la configuración del trabajo.

Luego, el cliente de trabajo de Hadoop envía el trabajo (archivo JAR/ejecutable, etc.) y la configuración al ResourceManager, que asume la responsabilidad de distribuir el software/configuración a los trabajadores, programar tareas y monitorearlas, proporcionando información de estado y diagnóstico al cliente del trabajo.

6. YARN (YET ANOTHER RESOURCE NEGOTIATOR)

6.1 Introducción

YARN (Yet Another Resource Negotiator) es el administrador de recursos de código abierto de Hadoop. Fue introducido en la versión 2.0 de Hadoop para mejorar la eficiencia y la flexibilidad en el procesamiento de datos distribuidos. YARN reemplazó el modelo de procesamiento MapReduce de Hadoop 1.x y permitió un procesamiento más diversificado y dinámico de datos en el cluster de Hadoop. Ha evolucionado para convertirse en un sistema operativo distribuido de gran escala utilizado para el procesamiento de Big Data.
La idea fundamental de YARN es dividir las funcionalidades de gestión de recursos y programación/seguimiento de trabajos en daemons separados. De esta manera se tendrá un ResourceManager (RM) global y un ApplicationMaster (AM) por aplicación. Una aplicación puede ser un único trabajo o un DAG de trabajos.

6.2 Componentes de YARN

El **ResourceManager** (RM) y el **NodeManager** (NM) forman el marco de datos y computación. El RM es la autoridad última que arbitra los recursos entre todas las

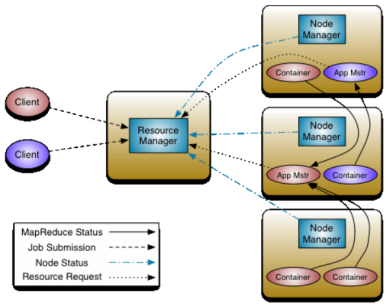

Figura 4. Arquitectura de YARN

aplicaciones en el sistema. El NM es el agente del marco por máquina responsable de los contenedores, monitorizando su uso de recursos (CPU, memoria, disco, red) y reportando lo mismo al RM.

El **ApplicationMaster** (AM) por aplicación es una biblioteca específica del marco que negocia recursos del RM y trabaja con los NM para ejecutar y monitorizar las tareas. Ver Figura 4

6.3 Funciones del ResourceManager

El ResourceManager tiene dos componentes principales:

- **Scheduler**: Asigna recursos a las aplicaciones en ejecución, sujeto a restricciones de capacidades y colas.
- **ApplicationsManager**: Acepta envíos de trabajos, negocia contenedores para ejecutar el AM y proporciona servicios de reinicio en caso de fallo.

6.4 Funcionamiento del Scheduler

El Scheduler asigna recursos a las aplicaciones según los requisitos de recursos. Utiliza una política enchufable para particionar los recursos del clúster entre las diversas colas y aplicaciones.

7. CLUSTER DE DATOS

El clúster de Hadoop se define como un grupo combinado de unidades no convencionales. Estas unidades están conectadas con un servidor dedicado que se utiliza como única fuente de organización de datos. Funciona como una unidad centralizada durante todo el proceso de trabajo. La carga de trabajo en un clúster de Hadoop se distribuye entre varios nodos que trabajan juntos para procesar datos.

Es fácil agregar nodos en el clúster para ayudar en otras áreas funcionales. Sin los nodos, no es posible analizar los datos de unidades no estructuradas.

Para el análisis de datos este tipo especial de clúster es compatible con la computación en paralelo para analizar datos.

El cluster de Hadoop:

- Es extremadamente útil para almacenar diferentes tipos de conjuntos de datos.
- Es compatible con el almacenamiento de una gran cantidad de datos diversos.

- Hadoop Cluster encaja perfectamente en situaciones de computación en paralelo para procesar datos.
- También es útil para procesos de limpieza de datos.

7.1 Principales Tareas del Clúster de Hadoop

- Es adecuado para realizar actividades de procesamiento de datos.
- Es una gran herramienta para recopilar grandes cantidades de datos.
- También agrega un gran valor en el proceso de serialización de datos.

7.2 Arquitectura del Clúster de Hadoop

Al trabajar con el Clúster de Hadoop, es importante comprender su arquitectura de la siguiente manera:

- Nodos Maestros: El nodo maestro desempeña un gran papel en la recopilación de una gran cantidad de datos en el Sistema de Archivos Distribuido de Hadoop (HDFS). Además, trabaja para almacenar datos con computación en paralelo mediante la aplicación de Map Reduce.
- Nodos Esclavos: Son responsables de la recopilación de datos. Mientras se realiza cualquier cálculo, el nodo esclavo es responsable de cualquier situación o resultado.
- Nodos Clientes: Hadoop se instala junto con la configuración. Cuando el Clúster de Hadoop necesita cargar datos, es el nodo cliente el responsable de esta tarea.

7.3 Ventajas

- Rentable: Ofrece una solución rentable para el almacenamiento y análisis de datos.
- Proceso Rápido: El sistema de almacenamiento en el clúster de Hadoop se ejecuta de manera rápida para proporcionar resultados rápidos. En el caso de que haya una gran cantidad de datos disponibles, esta herramienta es útil.
- Accesibilidad Fácil: Ayuda a acceder fácilmente a nuevas fuentes de datos. Además, se utiliza para recopilar tanto datos estructurados como no estructurados.

8. CONCLUSIONES

Hadoop, como ya vimos, representa un modelo de computación distribuido que ha revolucionado la forma en que se procesan grandes volúmenes de datos. Su capacidad para distribuir el procesamiento de datos en múltiples nodos lo hace no solo eficiente sino también altamente escalable. El modelo de Hadoop facilita el procesamiento paralelo de grandes conjuntos de datos, lo que es crucial en sistemas donde el manejo eficiente de la concurrencia y el paralelismo determina el rendimiento general de un sistema. Asimismo, la arquitectura de Hadoop y su manejo eficiente de grandes volúmenes de datos pueden inspirar nuevas formas de pensar sobre el almacenamiento de datos, la gestión de procesos y la seguridad en los sistemas.

El futuro de Hadoop parece prometedor, con continuas mejoras en su ecosistema. La integración con tecnologías emergentes como la inteligencia artificial y el aprendizaje automático, así como mejoras en la seguridad y la gestión, seguirán siendo áreas clave de desarrollo.

REFERENCIAS

Dean, J. y Ghemawat, S. (2004). Mapreduce: Simplified data processing on large clusters. In *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, 137–150. San Francisco, CA.

GeeksforGeeks (2024). Basics of hadoop cluster. `https://www.geeksforgeeks.org/basics-of-hadoop-cluster/?ref=ml_lbp`. Consultado el 6 de Mayo de 2024.

Hadoop, A. (2024a). Apache hadoop yarn – introduction. `https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html`. Consultado el 6 de Mayo de 2024.

Hadoop, A. (2024b). Hdfs design. `https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html`. Consultado el 6 de Mayo de 2024.

Hadoop, A. (2024c). Mapreduce tutorial. `https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html`. Consultado el 6 de Mayo de 2024.

PhoenixNAP (2024). Apache hadoop architecture explained. `https://phoenixnap.com/kb/apache-hadoop-architecture-explained`. Consultado el 6 de Mayo de 2024.

Services, A.W. (2024). ¿qué es la computación distribuida? `https://aws.amazon.com/es/what-is/distributed-computing/`. Consultado el 6 de Mayo de 2024.