EDUCACIÓN
EN LÍNEA

COMPUTACION PARALELA Y DISTRIBUIDA

HDFS

Integrantes: Lucero Anderson
Quilumbaquin Cristian
Vaca Diego

Nivel:6to

EL HDFS

- Es un sistema de archivos distribuido diseñado para ejecutarse en hardware básico. 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 fallas y está diseñado para implementarse en hardware de bajo costo
- HDFS proporciona acceso de alto rendimiento a los datos de la aplicación y es adecuado para aplicaciones que tienen grandes conjuntos de datos
- DFS relaja algunos requisitos POSIX para permitir el acceso de transmisión a los datos del sistema de archivos
- HDFS se creó originalmente como infraestructura para el proyecto del motor de búsqueda web Apache Nutch
- HDFS es ahora un subproyecto de Apache Hadoop

Fallo de hadware

- Una instancia de HDFS puede constar de cientos o miles de máquinas servidor, cada una de las cuales almacena parte de los datos del sistema de archivos.
- El hecho de que haya una gran cantidad de componentes y que cada componente tenga una probabilidad de falla no trivial significa que algún componente de HDFS siempre no es funcional. Por lo tanto, la detección de fallas y la recuperación rápida y automática de ellas es un objetivo arquitectónico central de HDFS.

Grandes conjuntos de datos

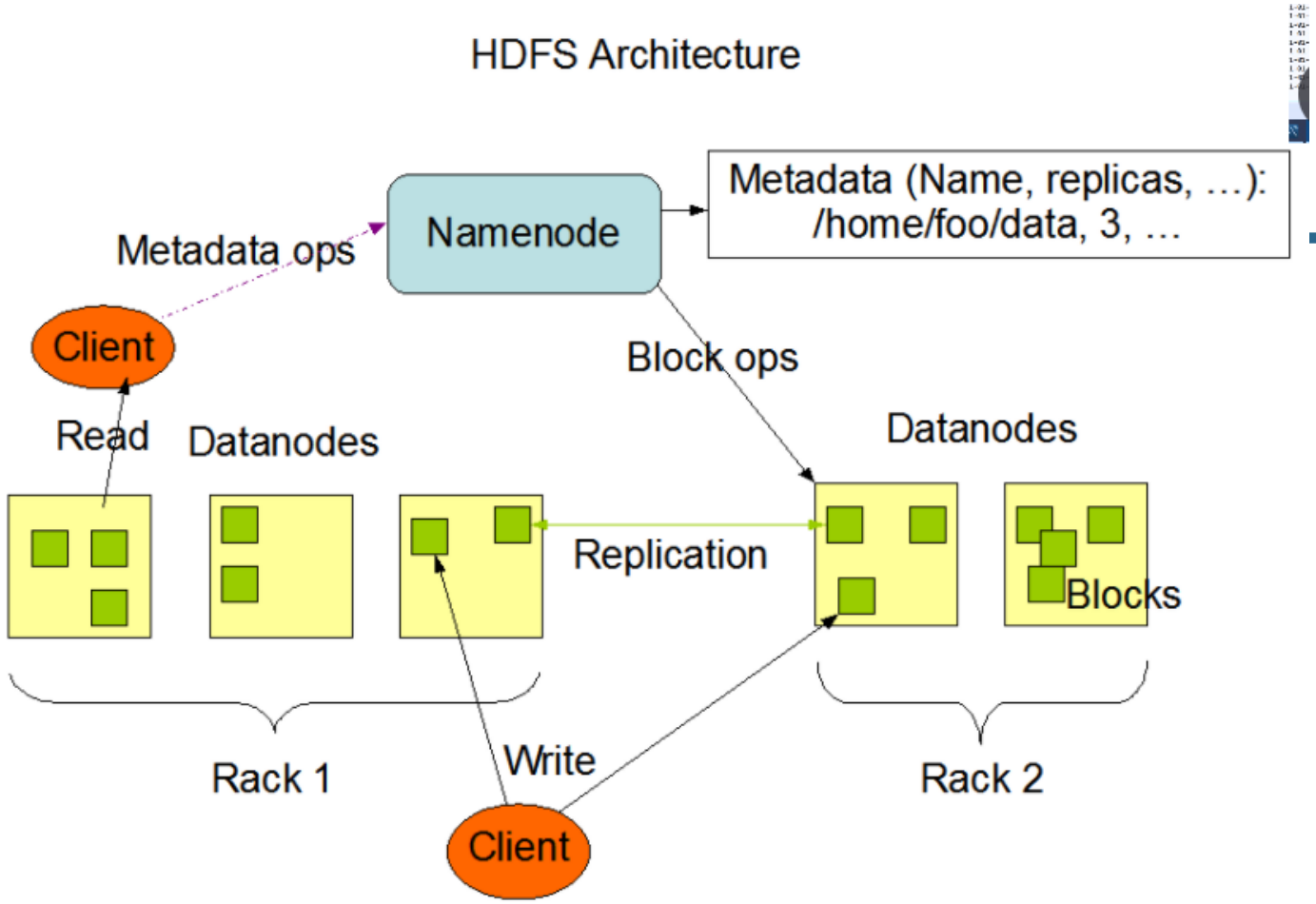
- Las aplicaciones que se ejecutan en HDFS tienen grandes conjuntos de datos.
- Un archivo típico en HDFS tiene un tamaño de gigabytes a terabytes. Por lo tanto, HDFS está optimizado para admitir archivos grandes.
- Proporciona un gran ancho de banda de datos agregados y escalar a cientos de nodos en un solo clúster.
- Además debería admitir decenas de millones de archivos en una sola instancia.

Modelo de coherencia simple

- Las aplicaciones HDFS necesitan un modelo de acceso de escritura única y lectura múltiple para los archivos.
- Un archivo una vez creado, escrito y cerrado no necesita ser modificado.
- Esta suposición simplifica los problemas de coherencia de los datos y permite un acceso a los datos de alto rendimiento.
- Una aplicación MapReduce o una aplicación de rastreo web encaja perfectamente con este modelo.
- Existe un plan para admitir la adición de escrituras a archivos en el futuro.

NameNodes y DataNodes

- Un clúster HDFS consta de un solo NameNode, un servidor maestro que administra el espacio de nombres del sistema de archivos y regula el acceso a los archivos por parte de los clientes.
- Además, hay varios DataNodes, generalmente uno por nodo en el clúster, que administran 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 del 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.
- NameNode ejecuta operaciones de espacio de nombres del sistema de archivos, como abrir, cerrar y cambiar el nombre de archivos y directorios.
- También determina la asignación de bloques a DataNodes.
- Los DataNodes son responsables de atender las solicitudes de lectura y escritura de los clientes del sistema de archivos.



- La existencia de un solo NameNode en un clúster simplifica enormemente la arquitectura del sistema. NameNode es el árbitro y el repositorio de todos los metadatos de HDFS. El sistema está diseñado de tal manera que los datos del usuario nunca fluyen a través del NameNode.

HDFS y el nombre del Sistema de archivos

- HDFS admite una organización de archivos jerárquica tradicional.
- Un usuario o una aplicación pueden crear directorios y almacenar archivos dentro de estos directorios.
- La jerarquía del espacio de nombres del sistema de archivos es similar a la mayoría de los demás sistemas de archivos existentes; uno puede crear y eliminar archivos, mover un archivo de un directorio a otro o cambiar el nombre de un archivo.
- HDFS aún no implementa cuotas de usuario.
- HDFS no admite enlaces físicos ni enlaces flexibles. Sin embargo, la arquitectura HDFS no excluye la implementación de estas características

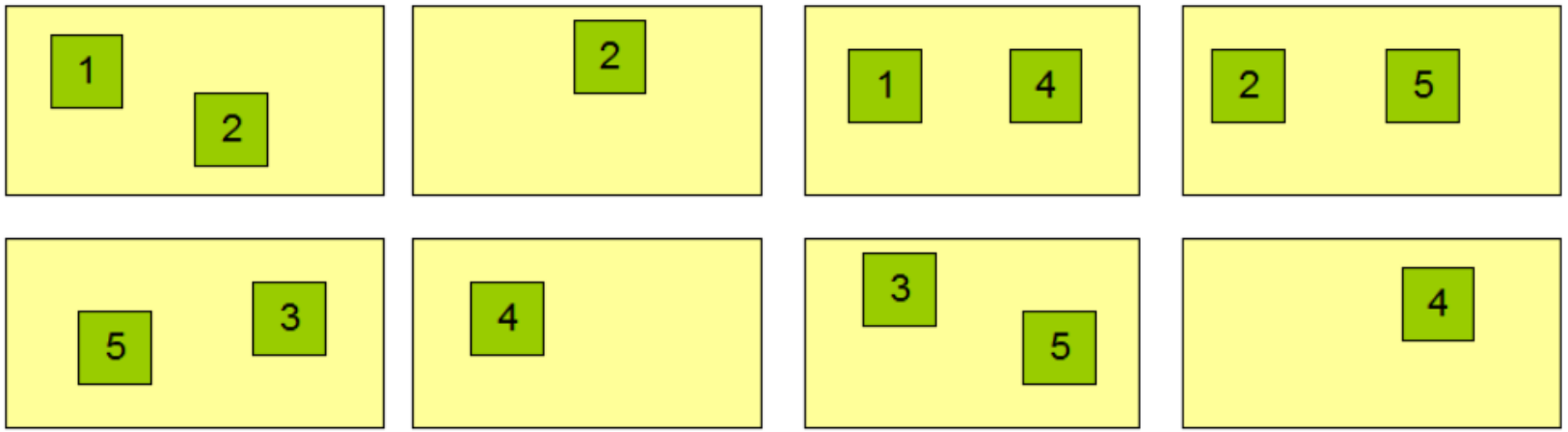
Replicacion de datos

- HDFS está diseñado para almacenar de manera confiable archivos muy grandes en máquinas en un clúster grande.
- Almacena cada archivo como una secuencia de bloques; todos los bloques de un archivo, excepto el último bloque, tienen el mismo tamaño.
- Los bloques de un archivo se replican para tolerancia a errores. El tamaño del bloque y el factor de replicación se pueden configurar por archivo.
- Una aplicación puede especificar el número de réplicas de un archivo.
- El factor de replicación se puede especificar en el momento de la creación del archivo y se puede cambiar más tarde.
- Los archivos en HDFS son de una sola escritura y tienen estrictamente un solo escritor en cualquier momento.

Block Replication

Namenode (Filename, numReplicas, block-ids, …)
/users/sameerp/data/part-0, r:2, {1,3}, …
/users/sameerp/data/part-1, r:3, {2,4,5}, …

Datanodes



Protocolos de comunicacion

- Todos los protocolos de comunicación HDFS están superpuestos al protocolo TCP / IP.
- Un cliente establece una conexión a un puerto TCP configurable en la máquina NameNode. Habla el ClientProtocol con el NameNode.
- Los DataNodes se comunican con el NameNode mediante el protocolo DataNode.
- Una abstracción de llamada a procedimiento remoto (RPC) envuelve tanto el protocolo de cliente como el protocolo de nodo de datos.
- Por diseño, NameNode nunca inicia ningún RPC. En cambio, solo responde a las solicitudes de RPC emitidas por DataNodes o clientes.

Fallo del disco de datos

- Cada DataNode envía un mensaje Heartbeat al NameNode periódicamente.
- Una partición de red puede hacer que un subconjunto de DataNodes pierda conectividad con NameNode.
- NameNode detecta esta condición por la ausencia de un mensaje Heartbeat. El NameNode marca los DataNodes sin Heartbeats recientes como muertos y no reenvía ningún IO nuevosolicitudes a ellos.
- Los datos que se registraron en un DataNode inactivo ya no están disponibles para HDFS.
- La muerte de DataNode puede hacer que el factor de replicación de algunos bloques caiga por debajo de su valor especificado.
- NameNode realiza un seguimiento constante de los bloques que deben replicarse e inicia la replicación siempre que sea necesario.
- La necesidad de volver a replicar puede surgir debido a muchas razones: un DataNode puede dejar de estar disponible, una réplica puede dañarse, un disco duro en un DataNode puede fallar o el factor de replicación de un archivo puede aumentar.

Equilibrio de clusters

- La arquitectura HDFS es compatible con los esquemas de reequilibrio de datos.
- Un esquema puede mover datos automáticamente de un DataNode a otro si el espacio libre en un DataNode cae por debajo de un cierto umbral.
- En el caso de una gran demanda repentina de un archivo en particular, un esquema puede crear dinámicamente réplicas adicionales y reequilibrar otros datos en el clúster.
- Estos tipos de esquemas de reequilibrio de datos aún no se han implementado.

Integridad de los datos

- Es posible que un bloque de datos obtenido de un DataNode llegue dañado. Esta corrupción puede ocurrir debido a fallas en un dispositivo de almacenamiento, fallas de red o software defectuoso. El software de cliente HDFS implementa la verificación de suma de comprobación en el contenido de los archivos HDFS.
- Cuando un cliente crea un archivo HDFS, calcula una suma de comprobación de cada bloque del archivo y almacena estas sumas de comprobación en un archivo oculto separado en el mismo espacio de nombres HDFS.
- Cuando un cliente recupera el contenido del archivo, verifica que los datos que recibió de cada DataNode coincidan con la suma de comprobación almacenada en el archivo de suma de comprobación asociado. De lo contrario, el cliente puede optar por recuperar ese bloque de otro DataNode que tenga una réplica de ese bloque.

Fallo del disco de metadatos

- FsImage y EditLog son estructuras de datos centrales de HDFS. Una corrupción de estos archivos puede hacer que la instancia de HDFS no funcione. Por este motivo, NameNode se puede configurar para admitir el mantenimiento de varias copias de FsImage y EditLog.
- Cualquier actualización de FsImage o EditLog hace que cada FsImages y EdjtLogs se actualicen sincrónicamente. Esta actualización sincrónica de múltiples copias de FsImage y EditLog puede degradar la tasa de transacciones de espacio de nombres por segundo que un NameNode puede admitir.
- Sin embargo, esta degradación es aceptable porque, aunque las aplicaciones HDFS son de naturaleza muy intensiva en datos, no son intensivas en metadatos. Cuando se reinicia un NameNode, selecciona el último FsImage y EditLog coherente para usar.

Instantaneas

- Las instantáneas admiten el almacenamiento de una copia de datos en un momento determinado. Un uso de la función de instantánea puede ser revertir una instancia de HDFS dañada a un buen momento conocido previamente. Actualmente, HDFS no admite instantáneas, pero lo hará en una versión futura.

Organizacion de datos

- **Bloques de datos**
- HDFS está diseñado para admitir archivos muy grandes. Las aplicaciones que son compatibles con HDFS son aquellas que se ocupan de grandes conjuntos de datos. Estas aplicaciones escriben sus datos solo una vez, pero los leen una o más veces y requieren que estas lecturas se satisfagan a velocidades de transmisión.
- HDFS admite la semántica de escribir una vez, leer muchas en archivos. Un tamaño de bloque típico utilizado por HDFS es de 64 MB. Por lo tanto, un archivo HDFS se divide en trozos de 64 MB y, si es posible, cada trozo residirá en un DataNode diferente.

Organizacion de datos

- **Puesta en escena**
- La solicitud de un cliente para crear un archivo no llega al NameNode inmediatamente. De hecho, inicialmente el cliente HDFS almacena en caché los datos del archivo en un archivo local temporal. Las escrituras de la aplicación se redirigen de forma transparente a este archivo local temporal.
- Cuando el archivo local acumula datos por valor de más de un tamaño de bloque HDFS, el cliente se pone en contacto con NameNode. NameNode inserta el nombre del archivo en la jerarquía del sistema de archivos y le asigna un bloque de datos. El NameNode responde a la solicitud del cliente con la identidad del DataNode y el bloque de datos de destino.

Organizacion de datos

Canalización de replicación

- Cuando un cliente escribe datos en un archivo HDFS, sus datos se escriben primero en un archivo local como se explica en la sección anterior. Suponga que el archivo HDFS tiene un factor de replicación de tres.
- Cuando el archivo local acumula un bloque completo de datos de usuario, el cliente recupera una lista de DataNodes del NameNode. Esta lista contiene los DataNodes que albergarán una réplica de ese bloque. A continuación, el cliente descarga el bloque de datos en el primer DataNode

Organizacion de datos

Accesibilidad

- Se puede acceder a HDFS desde aplicaciones de muchas formas diferentes. De forma nativa, HDFS proporciona una API de Java para que la utilicen las aplicaciones.
- El contenedor de lenguaje AC para esta API de Java también está disponible. Además, también se puede utilizar un navegador HTTP para explorar los archivos de una instancia HDFS. Se está trabajando para exponer HDFS a través del protocolo WebDAV .

Organizacion de datos

FS Shell

- HDFS permite organizar los datos del usuario en forma de archivos y directorios. Proporciona una interfaz de línea de comandos llamada FS shell que permite al usuario interactuar con los datos en HDFS. La sintaxis de este conjunto de comandos es similar a otros shells (por ejemplo, bash, csh) con los que los usuarios ya están familiarizados. A continuación, se muestran algunos ejemplos de pares de acción / comando:

Acción	Mando
Crea un directorio llamado / foodir	bin / hadoop dfs -mkdir / foodir
Eliminar un directorio llamado / foodir	bin / hadoop dfs -rmr / foodir
Ver el contenido de un archivo llamado /foodir/myfile.txt	bin / hadoop dfs -cat /foodir/myfile.txt

FS shell está destinado a aplicaciones que necesitan un lenguaje de secuencias de comandos para interactuar con los datos almacenados.

Organizacion de datos

DFSAdmin

- El conjunto de comandos DFSAdmin se utiliza para administrar un clúster HDFS.
- Estos son comandos que solo usa un administrador de HDFS. A continuación, se muestran algunos ejemplos de pares de acción / comando:

Acción	Mando
Pon el clúster en modo seguro	`bin / hadoop dfsadmin -safemode enter`
Generar una lista de DataNodes	`bin / hadoop dfsadmin -report`
Nodo (s) de datos de reinicio o desmantelamiento	`bin / hadoop dfsadmin -refreshNodes`

Interfaz del navegadoe

Una instalación típica de HDFS configura un servidor web para exponer el espacio de nombres HDFS a través de un puerto TCP configurable. Esto permite a un usuario navegar por el espacio de nombres HDFS y ver el contenido de sus archivos usando un navegador web.

Recuperacion de espacio

Archivos eliminados y recuperados

- Cuando un usuario o una aplicación elimina un archivo, no se elimina inmediatamente de HDFS. En su lugar, HDFS primero lo renombra a un archivo en el directorio `/ trash` . El archivo se puede restaurar rápidamente siempre que permanezca en `/ trash` .
- Un archivo permanece en `/ trash` durante un período de tiempo configurable. Después de que expire su vida en `/ trash` , NameNode elimina el archivo del espacio de nombres HDFS.
- La eliminación de un archivo hace que se liberen los bloques asociados con el archivo. Tenga en cuenta que podría haber una demora apreciable entre el momento en que un usuario elimina un archivo y el momento en que se produce el aumento correspondiente en el espacio libre en HDFS.

Recuperacion de espacio

Reducir el factor de replicación

- Cuando se reduce el factor de replicación de un archivo, NameNode selecciona el exceso de réplicas que se pueden eliminar. El siguiente Heartbeat transfiere esta información al DataNode.
- El DataNode luego elimina los bloques correspondientes y el espacio libre correspondiente aparece en el clúster. Una vez más, puede haber un retraso de tiempo entre la finalización de la `llamada a la` API `setReplication` y la aparición de espacio libre en el clúster.

UTE EC

¡GRACIAS!

TRAS CENDE MOS