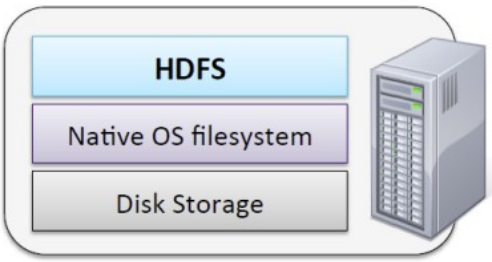
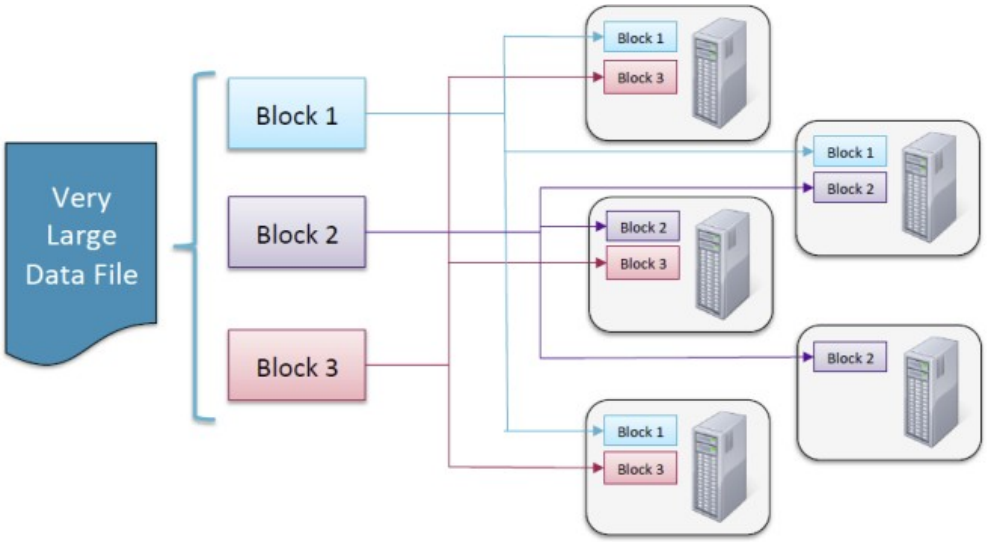
HDFS

HDFS es un sistema de archivos nativo de Google escrito en Java
Está desplegado sobre el sistema de archivos nativo de cada servidor
  ◦ Típicamente Línux: EXT3, EXT4 o XFS
Su característica más importante es que provee de almacenamiento redundante de grandes volúmenes de datos transparente al usuario
  ◦ El usuario no tiene que indicar que quiere replicar la información almacenada

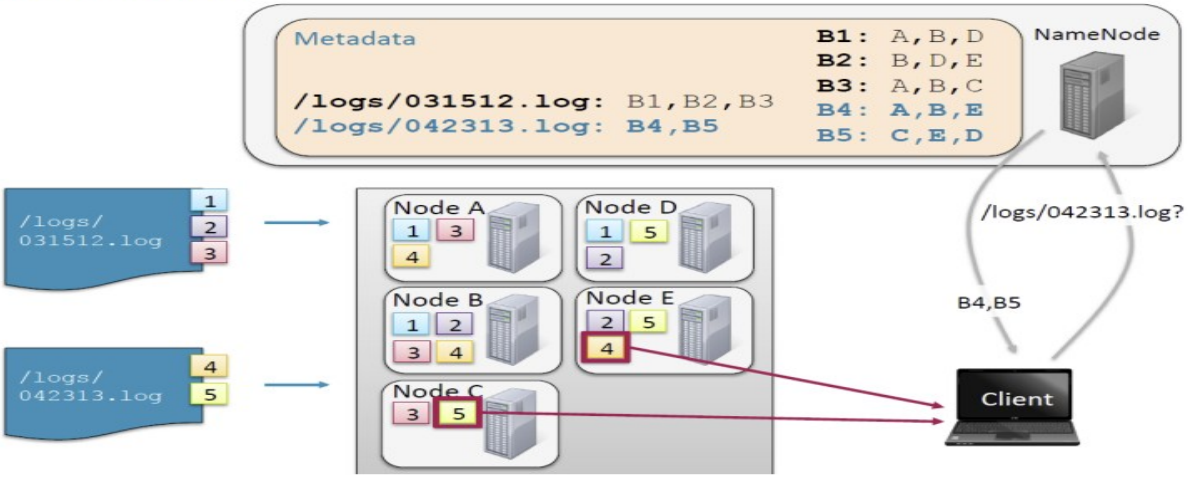


HDFS es óptimo cuando trabaja con un cantidad moderada de archivos de gran tamaño
  ◦ Millones mejor que Billones
  ◦ Cada archivo de 100 MB o más
  ◦ Disco óptico
HDFS está pensado para escribir una vez y leer muchas
  ◦ Se pueden modificar archivos escritos, pero es extremadamente costoso, por lo que se prefiere eliminar el archivo y copiarlo nuevamente actualizado.
HDFS está pensado para leer óptimamente archivos de gran tamaño, no para lecturas aleatorias

Funcionamiento 1
◦ Cada archivo se divide en bloques y distribuido entre los nodos del cluster en tiempo de escritura
◦ Cada archivo se replica 3 veces, por defecto, en el cluster, aunque este número es modificable



Funcionamiento 2

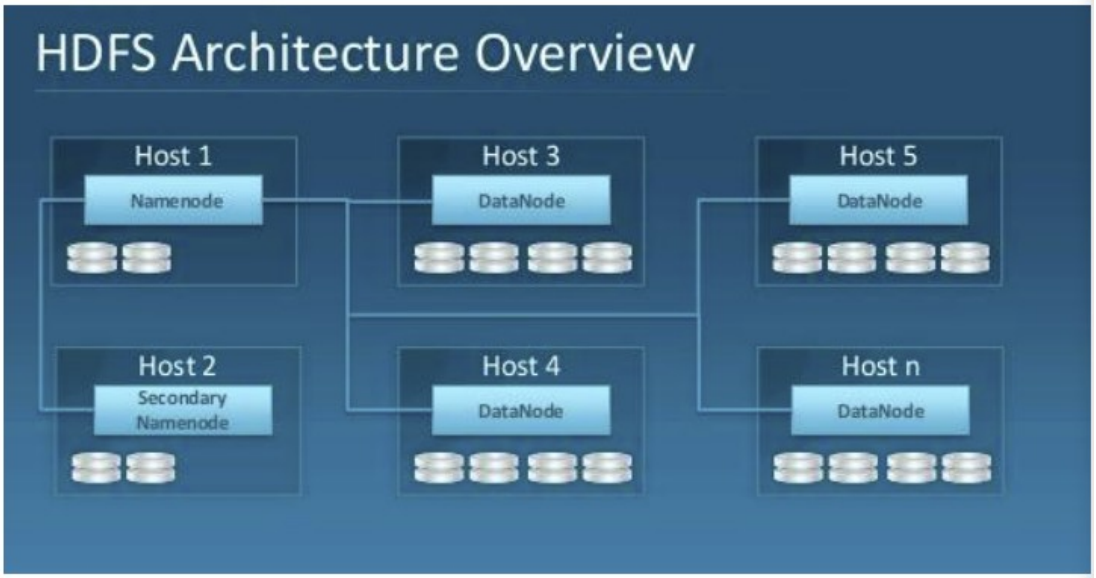


Demonios de HDFS

◦ Nodo Maestro. Sus demonios son
  ◦ NameNode
    ◦ Se encarga del almacenamiento y gestión de los metadatos
    ◦ Información de permisos y usuarios
    ◦ Qué bloques componen un archivo
    ◦ Dónde está cada bloque que compone un archivo
    ◦ Los metadatos son guardados en disco y cargados en memoria cuando el cluster arranca en un fichero llamado fsimage
    ◦ Los cambios realizados sobre los metadatos son almacenados en un fichero llamado edits en memoria
  ◦ Secondary Namenode
    ◦ Se encarga de realizar las labores de mantenimiento (edits - fsimage)
  ◦ Standby Namenode
    ◦ Para arquitecturas con Alta Disponibilidad
    ◦ Se encarga de realizar las labores de mantenimiento (edits - fsimage)
    ◦ Automáticamente toma el relevo

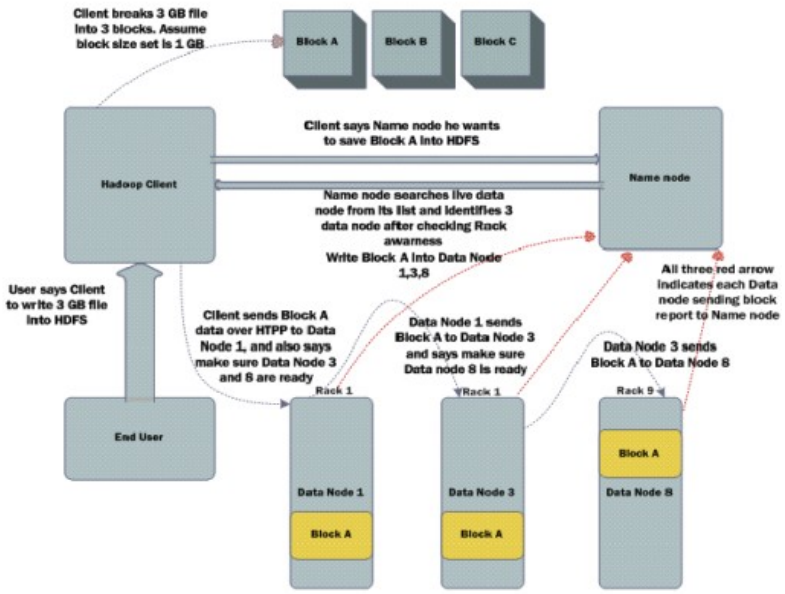
◦ Nodos esclavo. Su demonio es
  ◦ DataNode
    ◦ Los nodos que gobierna almacenan los bloques de datos, no información referente al bloque.
    ◦ Es el encargado de acceder a dichos bloques (blk_xxxxxxx)
    ◦ Cada bloque se almacena varias veces, dependiendo del factor de replicación
    ◦ Cuando un fichero entra en el cluster y se divide en bloques, cada nodo esclavo se encarga de hacer una copia al siguiente nodo esclavo hasta llegar al factor de replicación
    ◦ Se encargan de gestionar las tasks que componen un Job

Checkpoint del Secondary Namenode
◦ Cada hora o cada 1M de transacciones, el SNN realiza el checkpoint de los metadatos:
◦ 1. Llama al NameNode para obtener el fichero edits
◦ 2. Obtiene los ficheros fsimage y edits del NameNode
◦ 3. Carga el fichero fsimage en memory y agrega los cambios del fichero edits
◦ 4. Crea el nuevo fichero consolidado de fsimage
◦ 5. Envía el nuevo fichero consolidado al NameNode
◦ 6. El NameNode reemplaza el antiguo fsimage por el nuevo



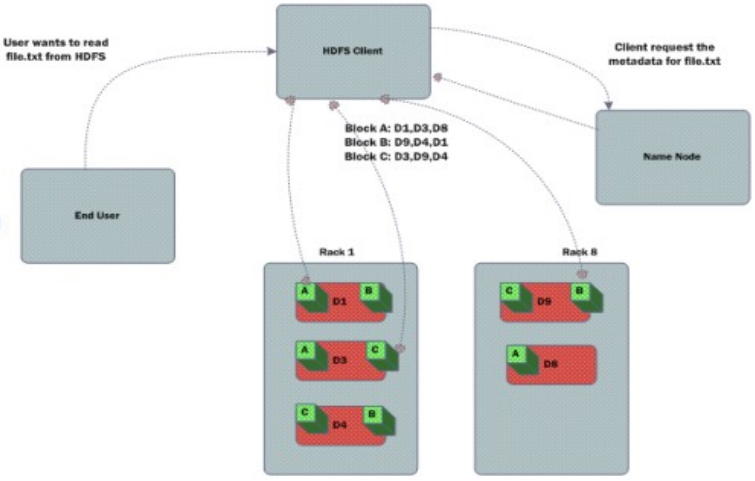
Procedimiento de escritura en HDFS

1. Cliente **conecta** con el NameNode
2. El NameNode busca sus metadatos y **devuelve el nombre del bloque y la lista de los DataNode**.
3. El Cliente conecta con el primer DataNode de la lista y **empieza el envío de los datos**
4. Se conecta con el segundo DataNode para realizar envío y lo mismo con el tercero.
5. Finaliza el envío
6. El cliente indica al NameNode donde ha realizado la escritura



Procedimiento de lectura en HDFS

1. Cliente **conecta** con el NameNode
2. El NameNode devuelve una **lista con los DataNode** que contienen ese bloque
3. El cliente conecta con el primer ode los DataNode y comienza la lectura del bloque



Fiabilidad y recuperación de datos en HDFS

- Los DataNode envían heartbeats al NameNode cada 3 segundos para indicar su estado
- Si pasado un tiempo (por defecto 5 minutos) el NameNode no recibe el heartbeat de alguno de los nodos, da por perdido ese nodo
  - El NameNode averigua que bloques había en el nodo perdido
  - El NameNode busca otros DataNode donde realizar la copia de los bloques perdidos y así mantener el número de replicación. Los DataNode serán los encargados de realizarse la copia de los bloques "perdidos"
- En caso de recuperar el nodo perdido, el sistema automáticamente decidirá que bloques eliminar para mantener el número de replicación de bloques.
- Speculative Execution:
  - Si el master detecta que un nodo esclavo está ejecutando las tasks más lento de lo normal, ordena lanzar los trabajos de ese nodo esclavo en otro nodo. El primer nodo que termina entrega el resultado. Los procesos que quedan en el otro nodo, e más lento, se matan.

WEB UI

- A través de la web UI del NameNode seremos capaces de ver el estado de nuestro cluster:
  - Espacio total del HDFS
  - Espacio ocupado del HDFS
  - Espacio disponible del HDFS
  - Estado de los Nodos
  - Navegación por el sistema de ficheros
  - …...
- Por defecto se encuentra en el puerto 50070

Acceso a datos en HDFS a través de la línea de comandos

- El sistema de archivos es muy similar al de Linux y sus comandos también:
  - Copiar un fichero de disco local a HDFS
    - $ hadoop fs -put /…/foo.txt /…/foo.txt
  - Listar el directorio Home del usuario
    - $ hadoop fs -ls
  - Listar el directorio root del usuario
    - $ hadoop fs –ls /
  - Mostrar el contenido de un fichero, por ej: /user/fred/bar.txt
    - $ hadoop fs -cat /user/fred/bar.txt
  - Copiar un fichero de HDFS al local, por ejemplo baz.txt
    - $ hadoop fs -get /user/fred/bar.txt baz.txt
  - Crear un directorio en el home de HDFS del usuario llamado input
    - $ hadoop fs -mkdir input
  - Borrar un directorio y todo su contenido
    - $ hadoop fs -rm -r input_old