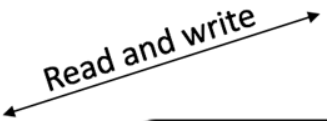


Hadoop Distributed File System



Client

Block 1



Block 2



Data Node 1



Data Node 2



Data Node 3



Data Node 4



Rack 1



Rack 2

Block 3




Rack 3



Name Node

- © Administrador para el cluster HDFS
- © Tareas principales: Balancear la carga y mantener registro de la localización de estos archivos en el cluster.

Namespace:

- © Guarda toda la metadata – la información sobre los archivos en el cluster
 - © Por ejemplo: Bloques con contenido, replication factor, ownership, client access etc. y donde se guardan
- 

Data Node

- © Trabajador esclavo
- © Almacenar data
- © Responder requests de escritura y lectura
- © Comunicarse con Name Node

DATANODE 1



DATANODE 2



DATANODE 3



DATANODE 4



RACK 1

Bloque

- © Data HDFS es almacenada en entidades de bloques
- © Un tamaño típico es de 64MB o 128MB
- © Los archivos se dividen en bloques para almacenarlos en un cluster HDFS
- © Los bloques sólo ocupan la memoria del actual espacio utilizado por el bloque



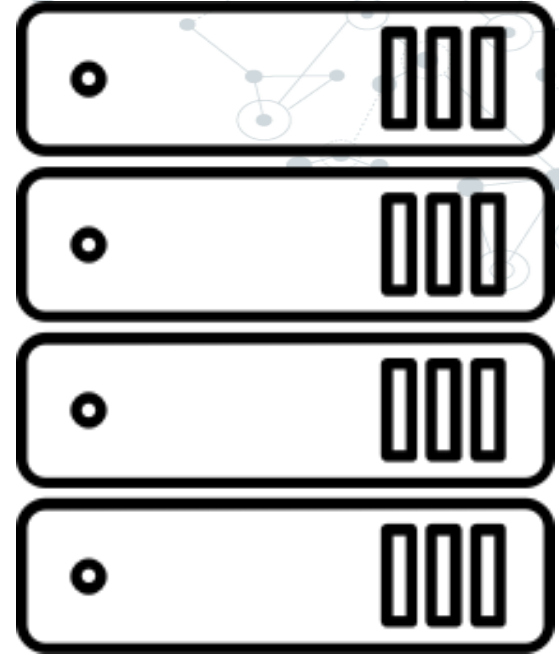
BLOCK 1

BLOCK 2

BLOCK 3

Rack

- © Un Rack consiste en una colección de Nodes, típicamente unos 30-40
- © Nodes dentro de un rack son almacenados físicamente cerca uno de otro
- © Nodes dentro de un rack están conectados al mismo network switch
- © El ancho de banda de la red entre dos nodes en un mismo Rack es más alto que en uno distinto




RACK 1

Cliente

- © El cliente es la interfaz de comunicación entre el usuario y el sistema HDFS
- © Hay dos tipos de clientes:
 - © Cliente regular se comunica con el Name Node y realiza operaciones de lectura y escritura
 - © Cliente administrador realiza tareas administrativas, por ejemplo, reiniciar el sistema o realizar upgrades

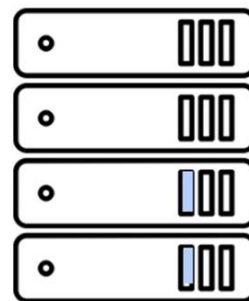
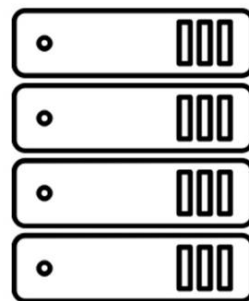
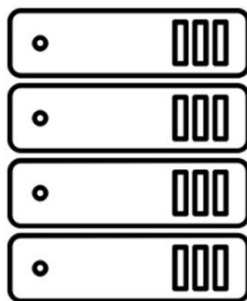
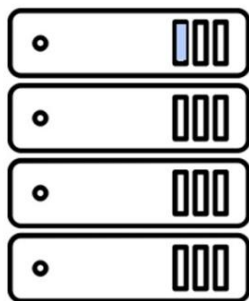
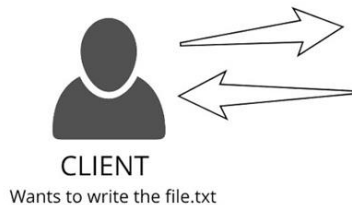


Replicación y distribución de los datos

- © Los bloques HDFS son replicados para ser tolerante a fallas
 - © El factor de replicación estándar es tres, pero es configurable
 - © La distribución de replicas en un cluster es crítica para asegurar una alta confianza y performance
- 

Lectura y escritura

- ⦿ El cliente le dice al Name Node que quiere escribir datos en el cluster, y este retorna un número de Data Nodes en que el cliente puede escribir
- ⦿ Pipelining: El cliente escribe al Data Node más cercano, que comparte los datos con el segundo Data Node, que en respuesta envía la información al último Data Node
- ⦿ Pipelining permite que los Data Nodes simultáneamente reciban y transfieran datos
- ⦿ Cuando el cliente quiere escribir, le pregunta al Name Node donde se almacenan los files blocks, y el Name Node retornará los bloques más cercanos al cliente



Rack Awareness Algorithm

- ◎ El Awareness Algorithm de un rack asegura que las replicas de un bloque estén dispersas a través del cluster y racks de cierta forma:
 - ◎ La primera replica de un bloque se almacenará en el rack local, preferiblemente en el mismo nodo en que se está escribiendo los datos
 - ◎ La segunda replica se almacenará en un rack diferente para asegurar consistencia en el caso de que el rack original falle
 - ◎ La última replica se almacenará en el segundo rack también, pero en un nodo diferente
- ◎ Por lo tanto, nunca se coloca más de una replica en un node de datos y no más de dos replicas son almacenadas en el mismo rack

Rack Awareness Algorithm

Block A :  Block B:  Block C: 

Rack - 1

1



2



3

4



Rack - 2

5

6

7

8



Rack - 3

9



10

11



12

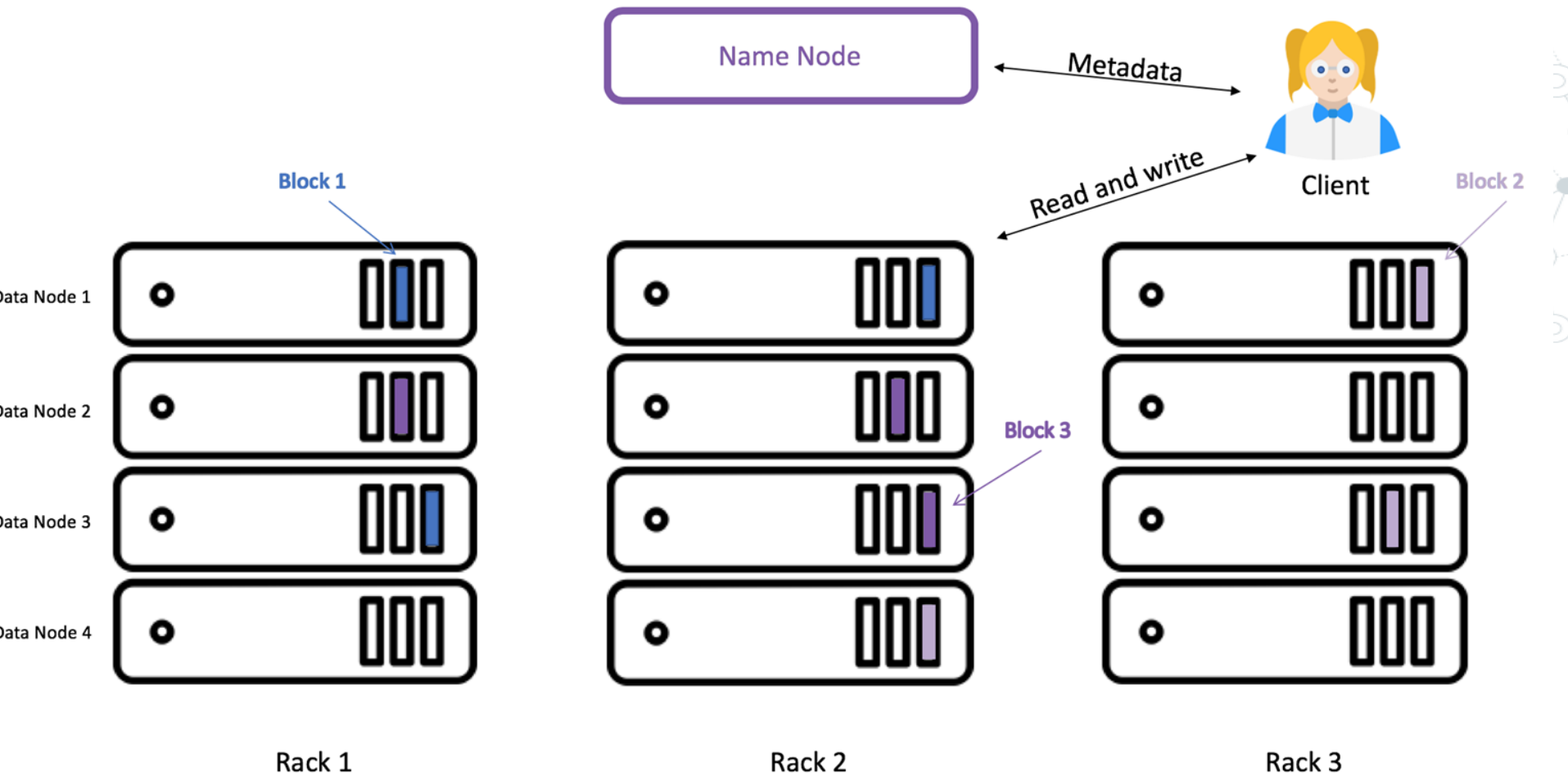


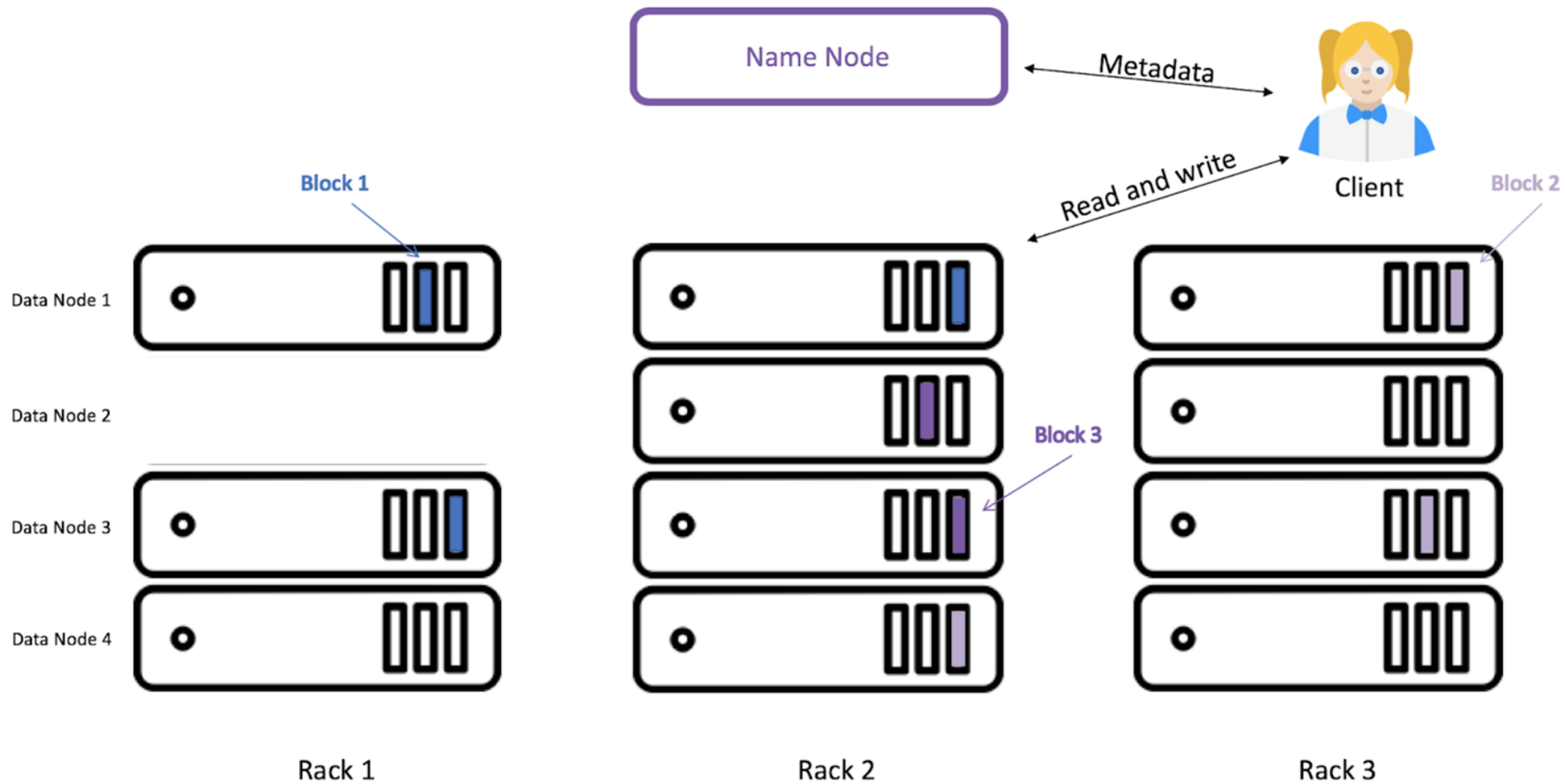
Tolerancia a fallas

- © El objetivo primario de HDFS es almacenar datos de forma segura, aún en casos de fallas, por lo que la tolerencia a fallas es necesaria
- © Los datos se almacenan en commodity hardware, por lo que habrán fallas
- © Hay tres tipos comunes de fallas:
 - © Data Node
 - © Network Partition
 - © Name Node

Data Node y Network Partitions

- © Los Data Nodes envían latidos a los Name Nodes cada tres segundos. Una ausencia de estos latidos se interpretan como la muerte de un Data Node. El Name Node detiene operaciones IO con el Data Node muerto y su contenido se copia de una replica sana a otro Data Node
- © En general, una falla de un único Data Node no es crítica debido a la replicación entre bloques. Cuando un Data Node falla, los bloques que se almacenan en un node ya están en otros dos, y la falla se descubrirá rápidamente y su replicación agendada debidamente
- © Una Network Partition puede causar que un subconjunto de nodes pierdan la conexión con el Name Node y que se prohíba que su señal de latido le llegue al Name Node





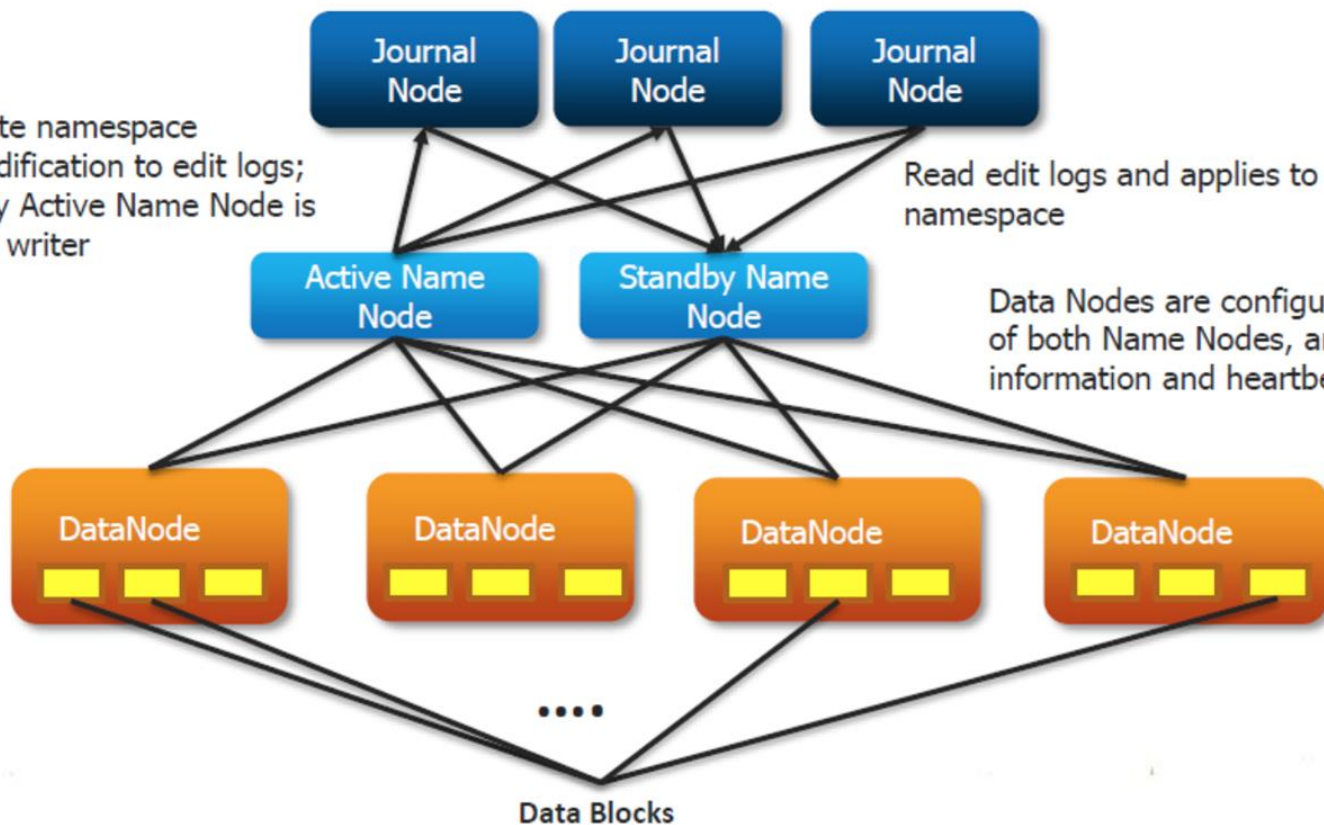
Name Node

- ⊙ Name Node es un sólo punto de falla, significando que procesos mediciones manuales se deben realizar en caso de que falle. Debido a que un Name Node es Vital para un cluster, hay muchas formas para asegurar confianza
- ⊙ Namespace image es el estado del namespace en cierto punto, que se guarda en la RAM y persistentemente en el disco (Checkpoint). Durante el reinicio, la imagen de namespace esta actualizado
- ⊙ Entre dos imagines, un edit log guarda todos los cambios hechos al cluster
- ⊙ El Name Node también puede correr un Checkpoint o un Backup Node al mismo tiempo que sirve al cliente. Ellos periódicamente juntan la actual namespace image y edit log en la RAM y lo devuelven al Name Node para actualizar el Checkpoint
- ⊙ Un Name Node secundario trabaja de forma similar al Checkpoint o Backup Node

Hadoop 2.0 - HDFS Alta Disponibilidad

- ◎ HDFS HA se lanzó con Hadoop 2.0
- ◎ El concepto de este es utilizar dos Name Nodes redundantes, uno activo y otro en modo stand-by
 - ◎ El node activo actúan como un Name Node normal
 - ◎ El Stand-by node mantiene su estado sincronizado para estar así listo para reemplazar al activo en caso de falla
- ◎ La sincronización se obtiene mediante Journal Nodes. El stand-by node continuamente vigila los Journal Nodes y copia cambios aplicados por el node activo
- ◎ Para mantener un imagen actualizada de las ubicaciones de los blocks, los Data Nodes envían la información de la ubicación de estos y latidos a ambos Name Nodes

Write namespace
modification to edit logs;
only Active Name Node
is the writer



Read edit logs and applies to its own
namespace

Data Nodes are configured with the location
of both Name Nodes, and send block location
information and heartbeats to both.

Ventajas de HDFS

1. **Archivos grandes:** El sistema maneja archivos grandes fácilmente, y de hecho, los prefiere frente a otros más chicos ya que tienen un tiempo de acceso menor
2. **Menor costo por byte:** HDFS se corre en commodity servers, por lo que el costo per byte es mucho menor que otros competidores
3. **Alto ancho de banda:** HDFS encuentra el requisito de Big Data de entrega de datos on una gigantesca razón de data
4. **Confianza de datos:** El sistema está construido en su totalidad para evitar caídas. Ha probado su confianza en numerosos casos de uso en clusters de diferentes tamaños.
5. **Escalabilidad:** Los clusters HDFS consisten en miles de nodes, y es la memoria del Name Node la que limita la escalabilidad. Al incrementar los tamaños de los archivos, los clusters se pueden escalar más

Cuando no ocupar HDFS

1. **Analíticas en tiempo real:** Si el procesamiento rápido de datos es requerido, el uso de HDFS será desventajoso. Esto es debido a que HDFS está diseñado para procesamiento de lotes por lo cual el tiempo de análisis completo será alto
2. **Múltiples datasets más pequeños y procesamiento de datasets pequeños**
La plataforma Hadoop no se recomienda para conjuntos de datos de estructura pequeña o para procesar conjuntos más pequeños.
3. **HDFS WORMT (Write Once Read Many Times)**
Si los datos se actualizan frecuentemente, HDFS no es la correcta elección, ya que es un sistema que sólo adjunta.

El sistema asume que un archivo HDFS no se modificará una vez escrito, pese a que se acceda a este múltiples veces