HDFS –Hadoop distributed File system

Prof. H.L. Phalachandra

LEVERAGING SLIDES of Prof. K V Subramaniam

Why the need?

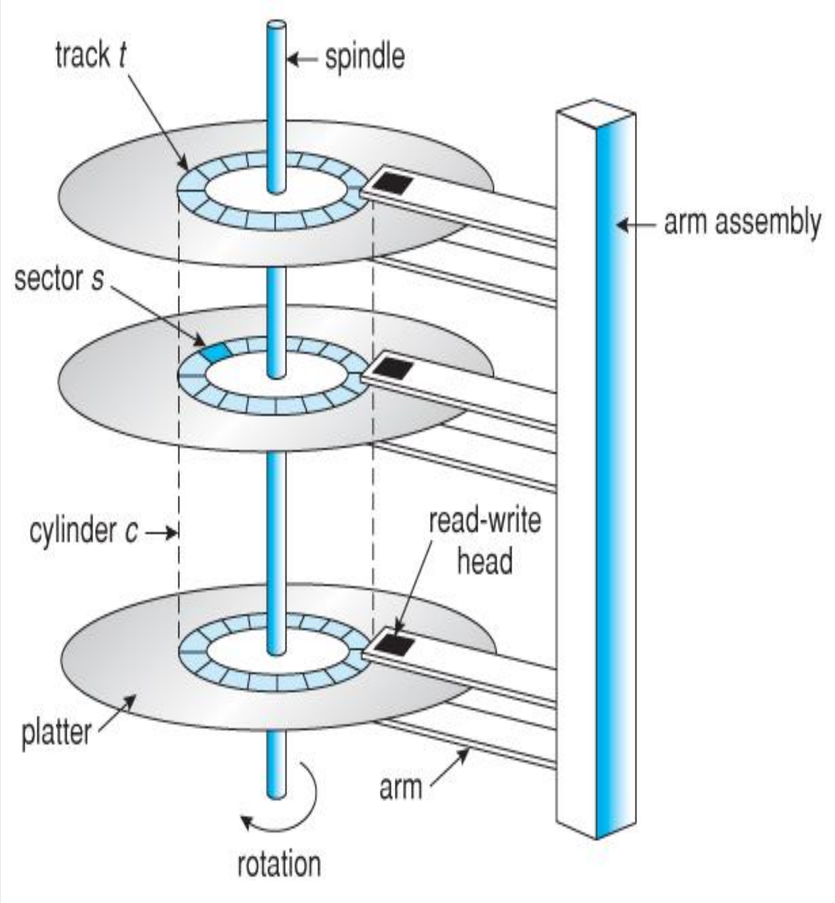
- As per RBI in May 2019,
  - #credit/debit card transactions~ 1.3 Billion (https://rbidocs.rbi.org.in/rdocs/ATM/PDFs/ATM052019E96EC259708C4ED9AD9E0C6B5E8B6DD5.PDF)
  - If each transaction requires about 10K of data

    13 TB of data

  - That's a lot of data and this is
  - There are other transactions also
  - Suppose you want to look for fraudulent transactions
- How to store and process this data?

Persistent Storage - Disks

- Block oriented devices
  - unit of data transfer is a block – e.g. 4KB on some Unix systems
- Can we store the persistent data directly on these blocks?
- Concerns:
  - Which block contains the data
  - Who will maintain the meta-data?
- Typically handled by Filesystems



File System Terminologies

- **File system** is used to control how data is stored and retrieved. Without a file system, information placed in a storage medium would be one large body of data with no way to tell where one piece of information stops and the next begins.
- Data is separated and grouped into pieces and given a name called a **file**
- **Files** thus are named collection of related information on disks
- Desirable properties of files
  - Long term existence (stored on disk or other secondary storage and do not disappear when a user logs off)
  - Sharable between processes (have names and can have associated access permissions that permit controlled sharing)
  - Structure (files can be organized into hierarchical or more complex structure to reflect the relationships among files)
- The ***structure and logic rules*** used ***to manage the groups of information (files) and their names*** is what forms a file system. It also has a collection of functions that can be performed on files.
- File system also maintains a set of attributes associated with the file
- Typical operations include: Create, Delete, Open, Close, Read, Write
- Operating system can have multiple file systems
- File systems
  - Can be used on numerous different types of storage devices that uses different kinds of media
  - There are also many different kinds of file systems.
  - Each one has different structure and logic, properties of speed, flexibility, security, size and more.

File System –design principles

- Separates information into data and meta-data
- Mapping between these and block level abstraction provided by disks
- Block size

File system meta-data

Size of the filesystem

Free and used blocks of the disk

File Meta-Data

Name of the file

Size of the file

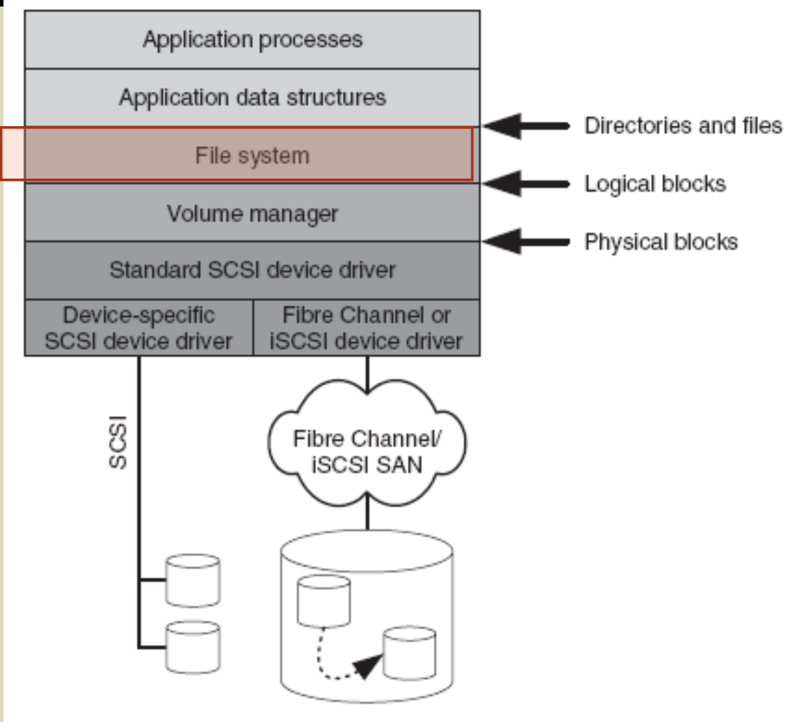
Permissions

Data

Actual contents of the file

File Systems : Local File Systems

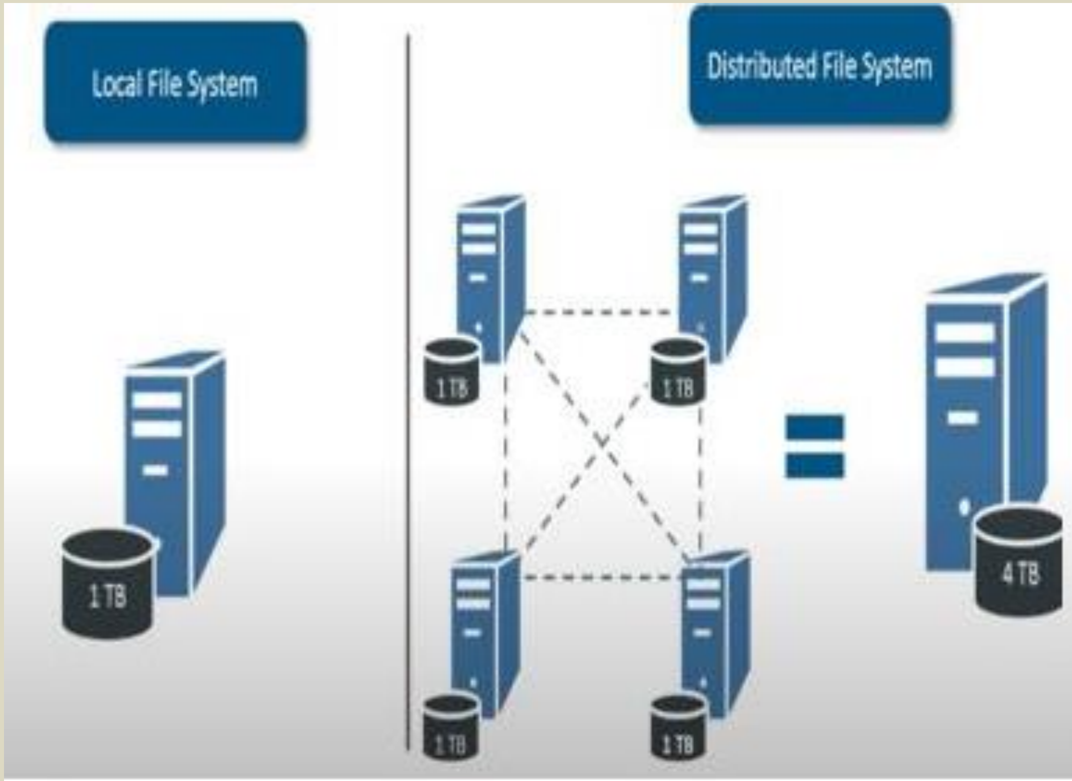
- For storing data
  - An operating system will engage and schedule a storing level process
  - The filesystem manages where within the address space the data needs to be stored
  - Applications & users, use the storage capacity of the disks via directories and files.


Application processes
Application data structures
File system
Volume manager
Standard SCSI device driver
Device-specific SCSI device driver
Fibre Channel or iSCSI device driver
Directories and files
Logical blocks
Physical blocks
SCSI
Fibre Channel/ iSCSI SAN

- File systems form an intermediate layer between applications & block oriented disks (with layers of volume management software which includes RAID controllers, virtualization systems) coming into play
- File systems and Volume managers are generic, and support all kinds of applications, and are not tuned for performance of any particular type of Applications

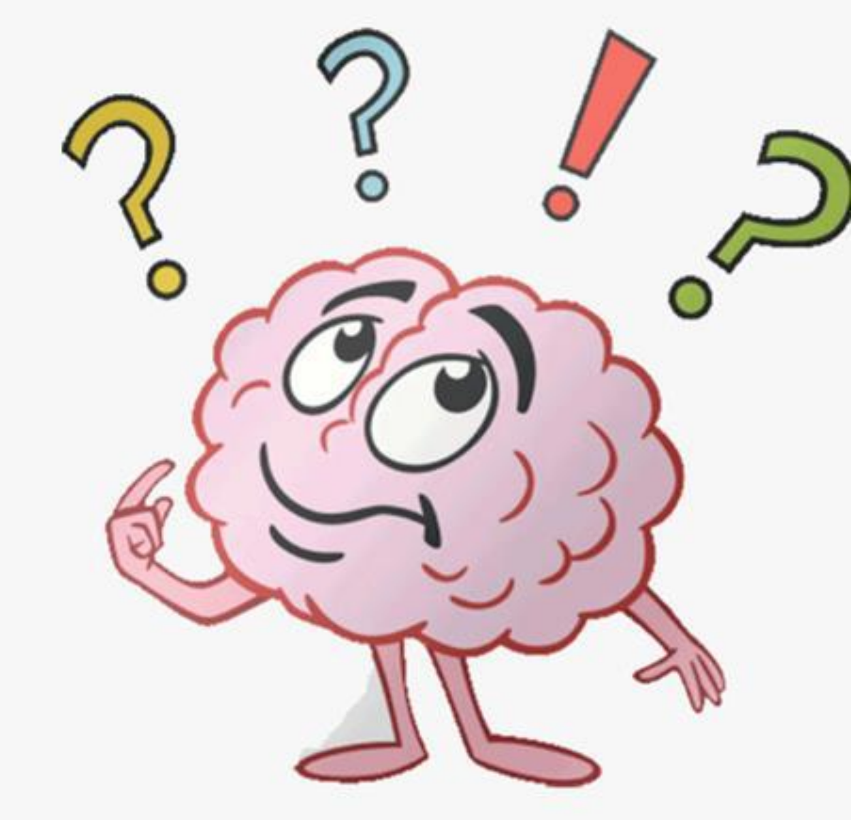
Distributed File System

- Consider the case when data is so large that it cannot fit on a single disk
  - DFS manages files and folders across multiple computers
- DFS can organize and display files as if they are stored on one computer
  - It serves the same purpose as a tradition al file system
- Designed to provide file storage and controlled access to files over local and wide area networks



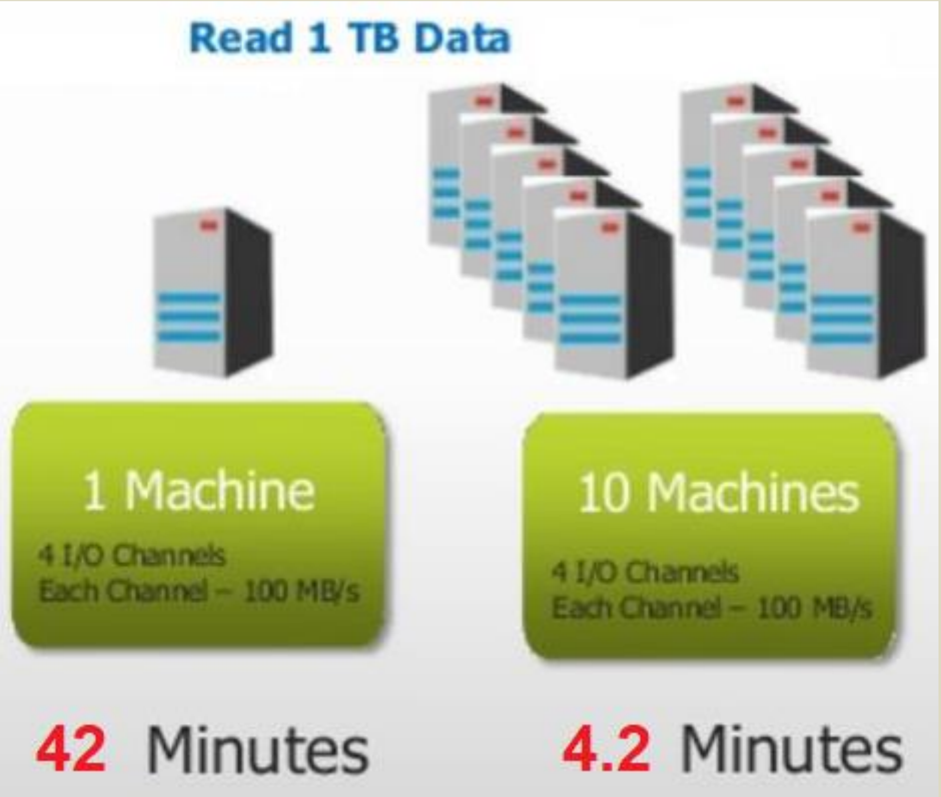
Exercise

- Consider that you have 1 TB of data?
- Compare the time taken to read data in both the cases below
  - Single machine (4 I/O channels each channel 100mb/S)
  - 10 machines (each having 4 I/O channels each channel 100mb/s)



Exercise

- Consider that you have 1 TB of data?
- Compare the time taken to read data in both the cases below
    - Single machine (4 I/O channels each channel 100mb/S)
    - 10 machines (each having 4 I/O channels each channel 100mb/s)

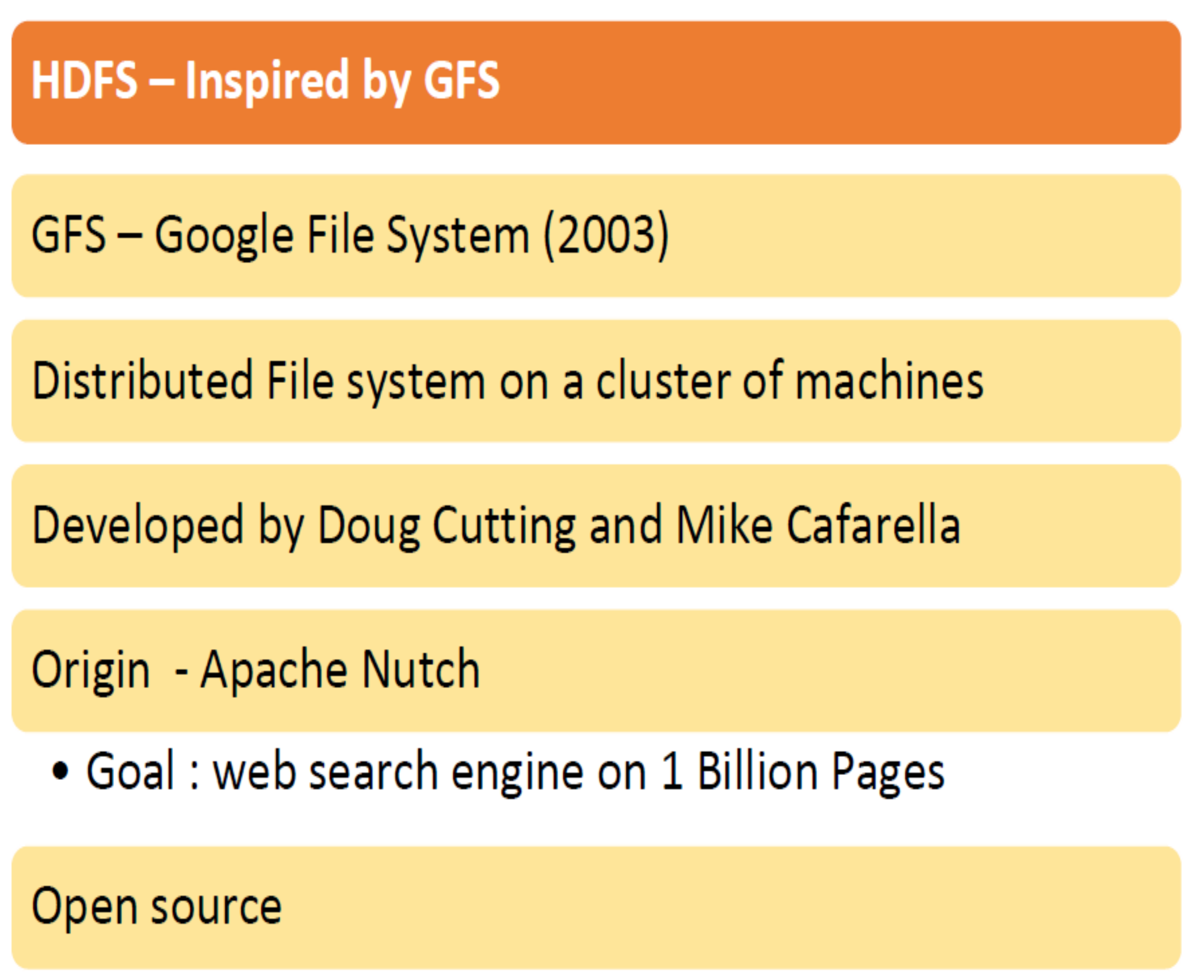


What about existing FS?

- NFS –Network File System
  - Small block sizes 8KB-16KB → large meta-data
  - Designed for large number of small files
- Handling scaling
  - Extending size of filesystem requires growing the volume
  - Can mainly handle scaling up.
- Scaling out –the need
  - A single machine cannot handle the load. Need a cluster
  - Originally not by design.
    - Was added later on need (Clustered Shared Volume in NTFS).
  - Need to support distributed operations

HDFS is based on the opensource version of GFS

What about existing FS?



HDFS – Inspired by GFS

GFS – Google File System (2003)

Distributed File system on a cluster of machines

Developed by Doug Cutting and Mike Cafarella

Origin - Apache Nutch

- Goal : web search engine on 1 Billion Pages

Open source

HDFS –Hadoop Distributed File System

“HDFS is a filesystem designed for storing very large files with streaming data access patterns, running on clusters of commodity hardware.”

Very large

- Files can be MB/GB/TB in size
- Hadoop clusters that are PB are currently operational

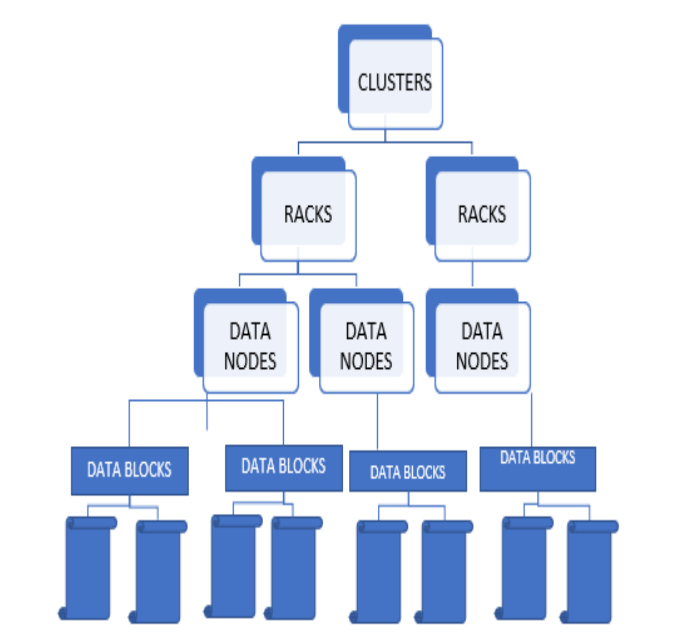
Read Mostly data

- most efficient data processing pattern is a write-once, read-many-times pattern.
- Each analysis will involve a large proportion of the dataset
- time to read the whole dataset is more important than the latency in reading the first record.

Commodity hardware

- Hadoop doesn’t require expe
- Designed to run on clusters of commodity hardware

Hadoop Distributed File System



Exercise

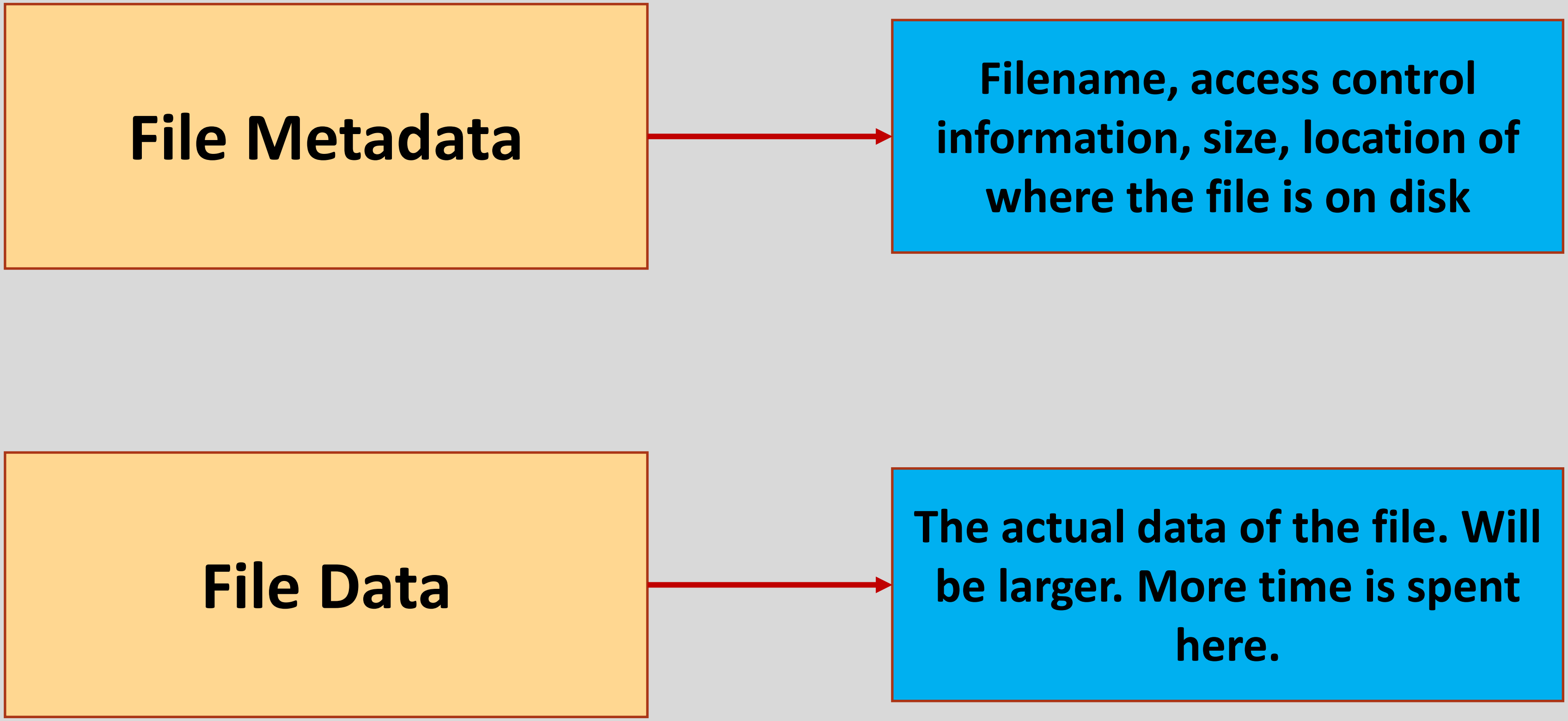
- If you want to store a file on disk what constitutes
  - Data
  - Metadata
- What are their access patterns
  - How often do you think each one will be accessed during a normal file read
- How large are they (comparatively)? Wh this important?

Data: much larger in size. Occupies multiple blocks

Metadata: smaller compared to data. Only information on filenames and blocks it occupies.

Since data is much larger. Most time spent in fetching data

HDFS Motivation



File Metadata

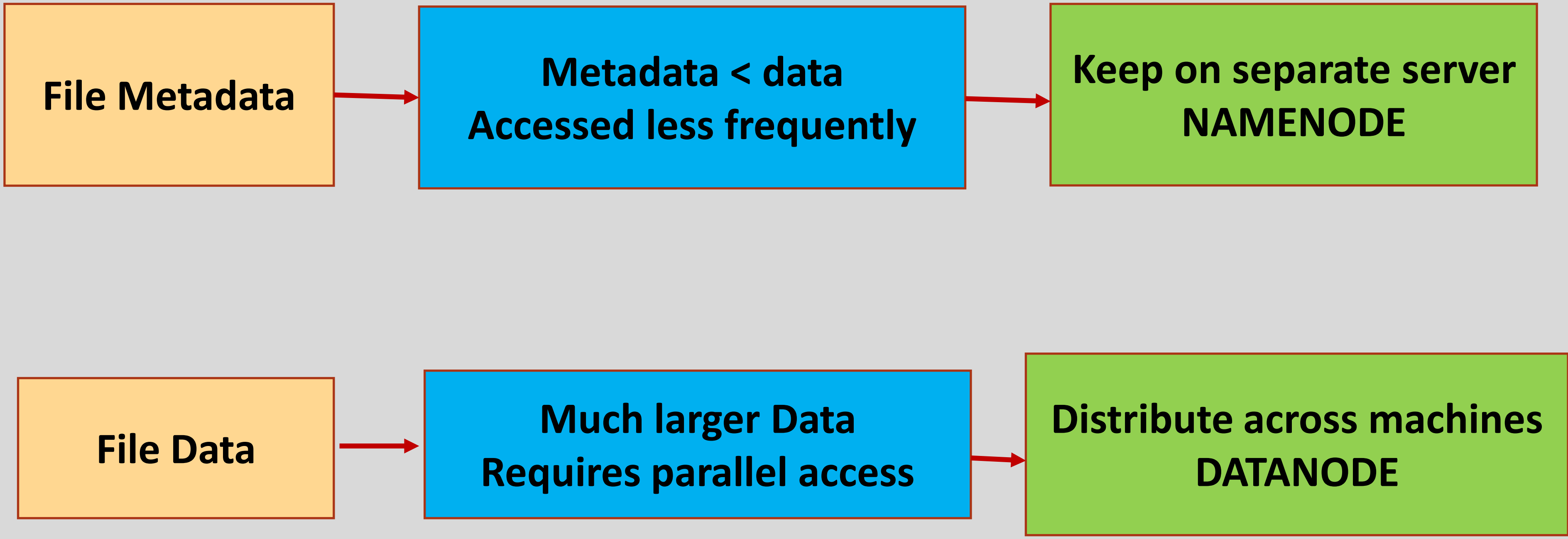
Filename, access control information, size, location of where the file is on disk

File Data

The actual data of the file. Will be larger. More time is spent here.

HDFS Motivation

Solution



File Metadata

Metadata < data
Accessed less frequently

Keep on separate server
NAMENODE

File Data

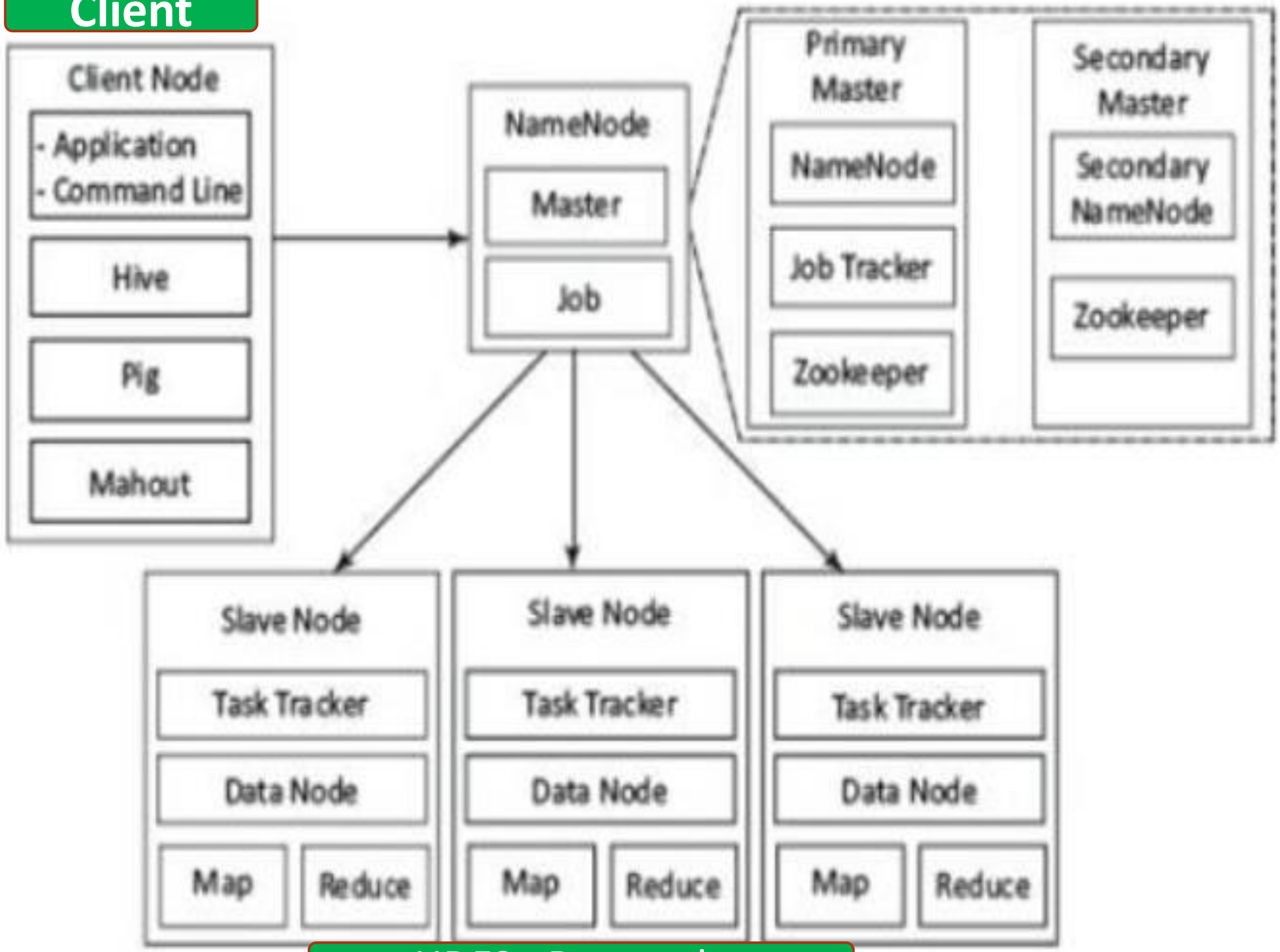
Much larger Data
Requires parallel access

Distribute across machines
DATANODE

HDFS –Master Slave Architecture

HDFS Master :Namenode

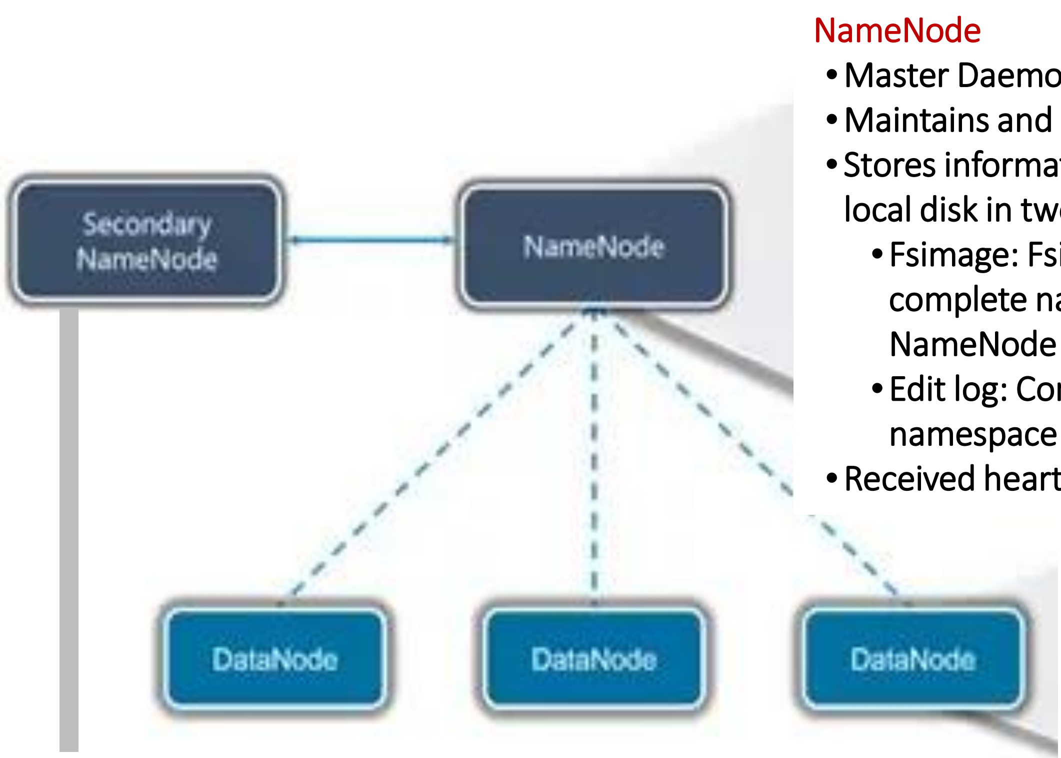
Client



HDFS : Datanodes

The client, master NameNode, MasterNodes and slave nodes

HDFS - Architecture



NameNode

- Master Daemon
- Maintains and Manages DataNodes
- Stores information about blocks locations, permissions, etc. on the local disk in two files:
  - Fsimage: Fsimage stands for File System image. It contains the complete namespace of the Hadoop file system since the NameNode creation.
  - Edit log: Contains all recent changes performed to the file system namespace to the most recent Fsimage
- Received heartbeat and block report from all the DataNodes

DataNodes

DataNodes are the slave nodes and the workhorses of HDFS

These are inexpensive commodity hardware storing blocks of a file

These are responsible for serving the client read/write requests

Based on the instruction from the NameNode, DataNodes performs block creation, replication, and deletion.

DataNodes send a heartbeat to NameNode to report the health of HDFS.

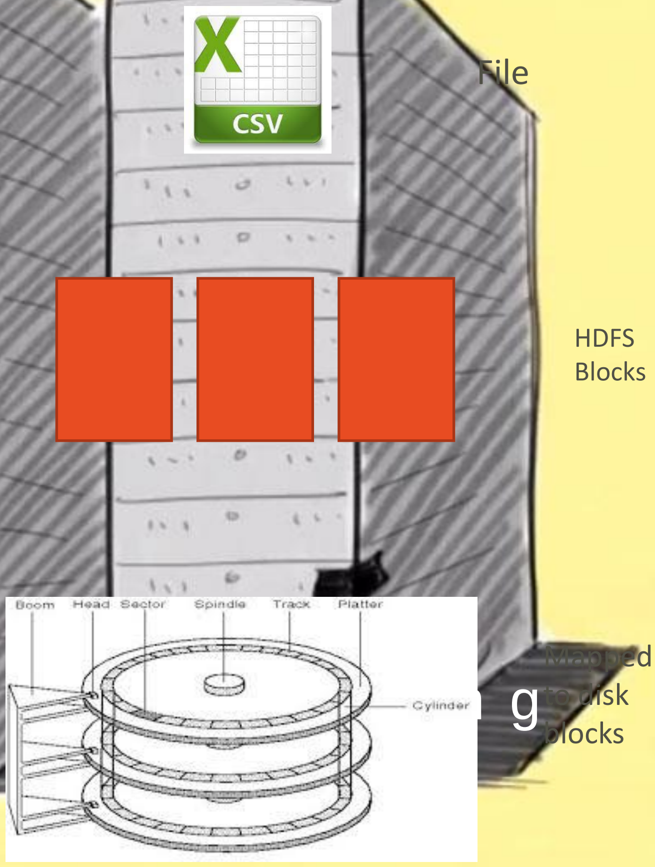
DataNodes also sends block reports to NameNode to report the list of blocks it contains.

Secondary NameNode

- Helper Node to Primary NameNode
- Housekeeping backup (not hot standby)
- Supports primary NameNode by downloading and merging the Fsimage file and edit logs file, updates the Fsimage and refreshes the edit logs
- It then sends this updated image to NameNode and enables NameNode to start faster

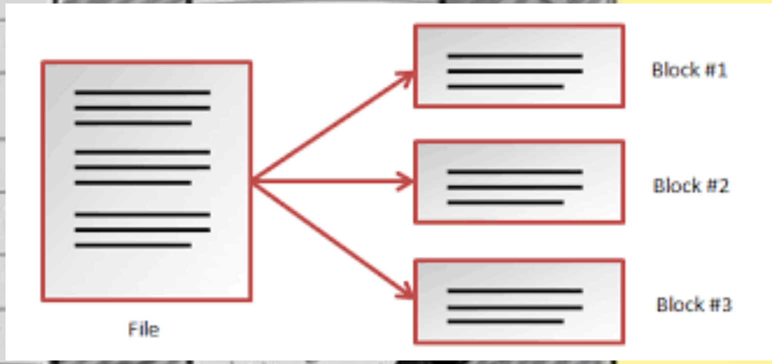
HDFS Blocks: What

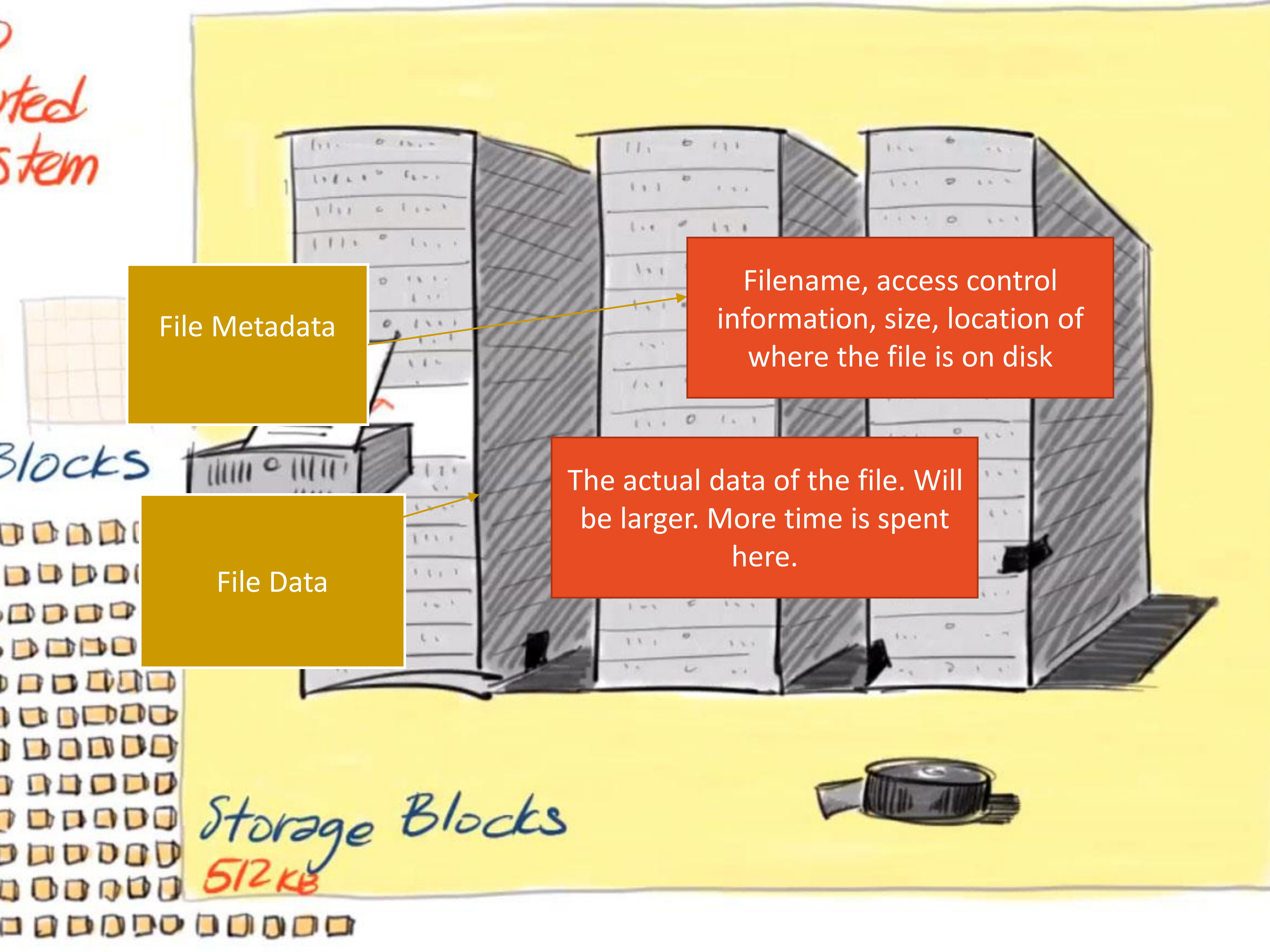
- Disk Blocks:
  - Minimum data that can be read written.
  - Typically 512 bytes.
- HDFS blocks
  - Much larger unit
    - 128MB (in v2)
- Files in HDFS are broken into block-sized chunks, which are stored as independent units.
- A file in HDFS that is smaller than a single block does not occupy a full block's worth of storage.
  - Use as many disk blocks as necessary.



HDFS Blocks: Why

- Benefits of block abstraction.
  - A file can be larger than any single disk in the network.
    - Files can be distributed across disks
  - Simplifies the storage subsystem
  - Blocks fit well with replication for providing fault tolerance and availability.
- % **`hadoop fsck -files –blocks`**
  - will list the blocks that make up each file in the filesystem



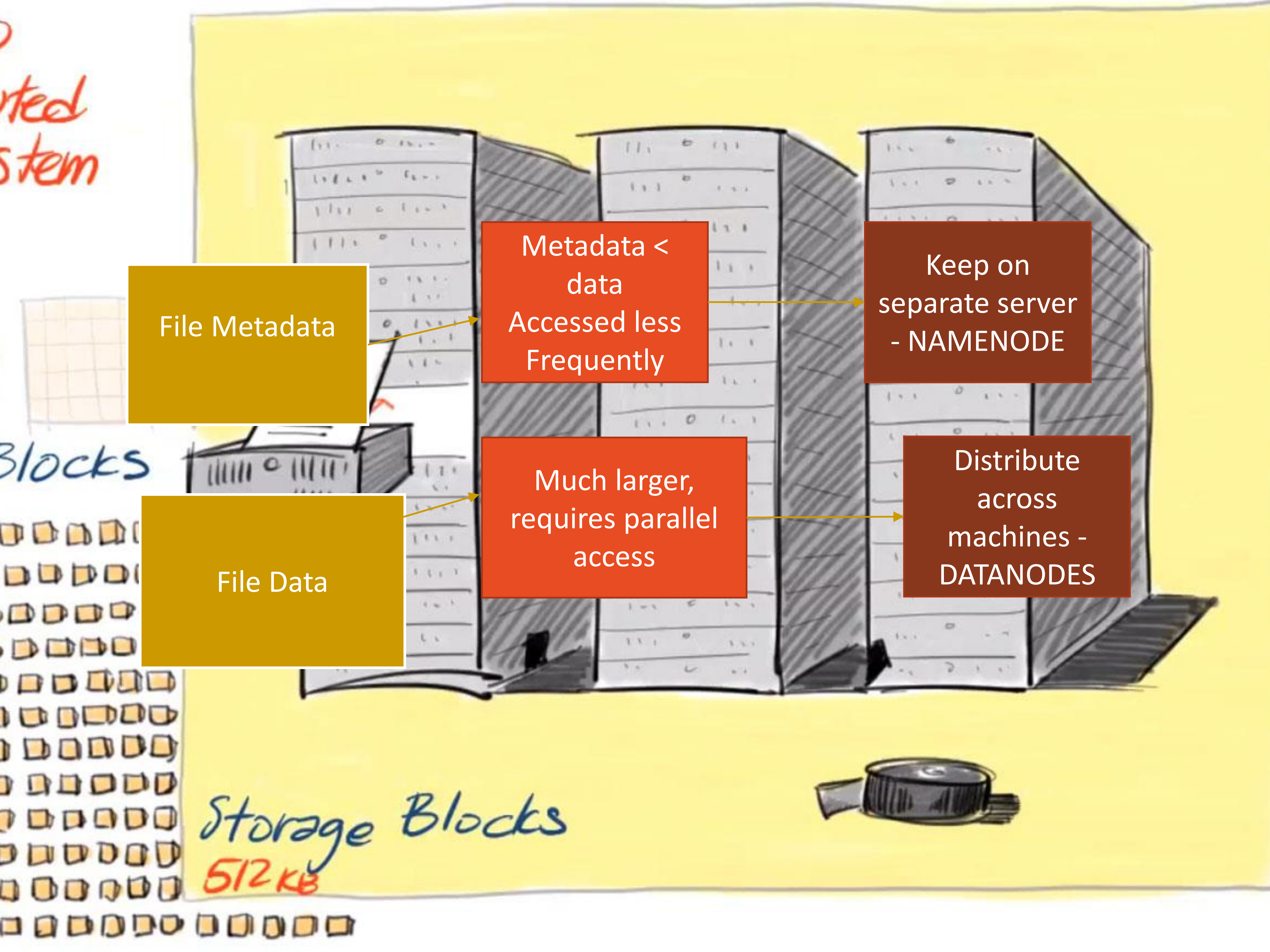


File Metadata

Filename, access control information, size, location of where the file is on disk

File Data

The actual data of the file. Will be larger. More time is spent here.



File Metadata

Metadata < data
Accessed less Frequently

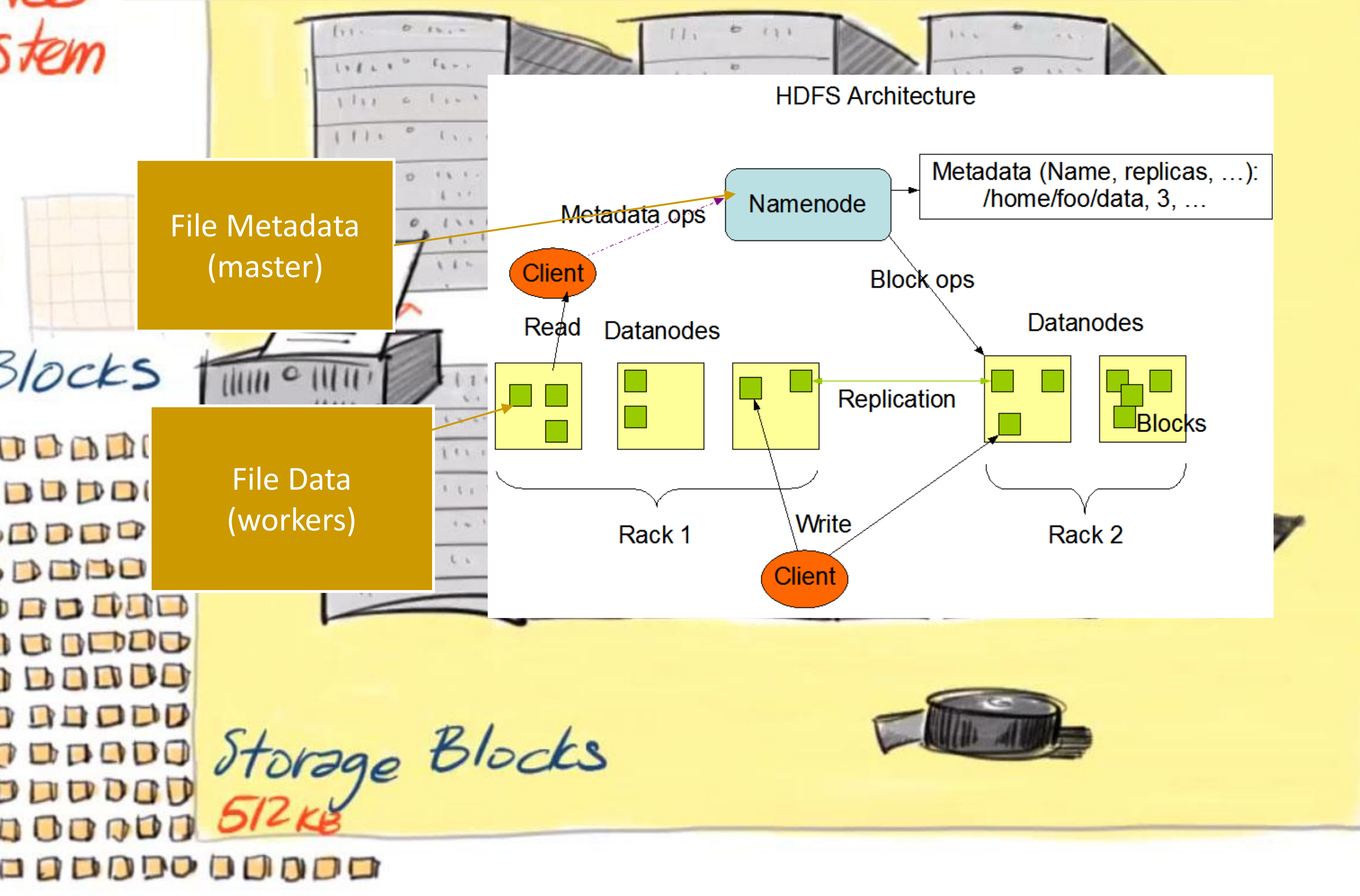
Keep on separate server - NAMENODE

File Data

Much larger, requires parallel access

Distribute across machines - DATANODES

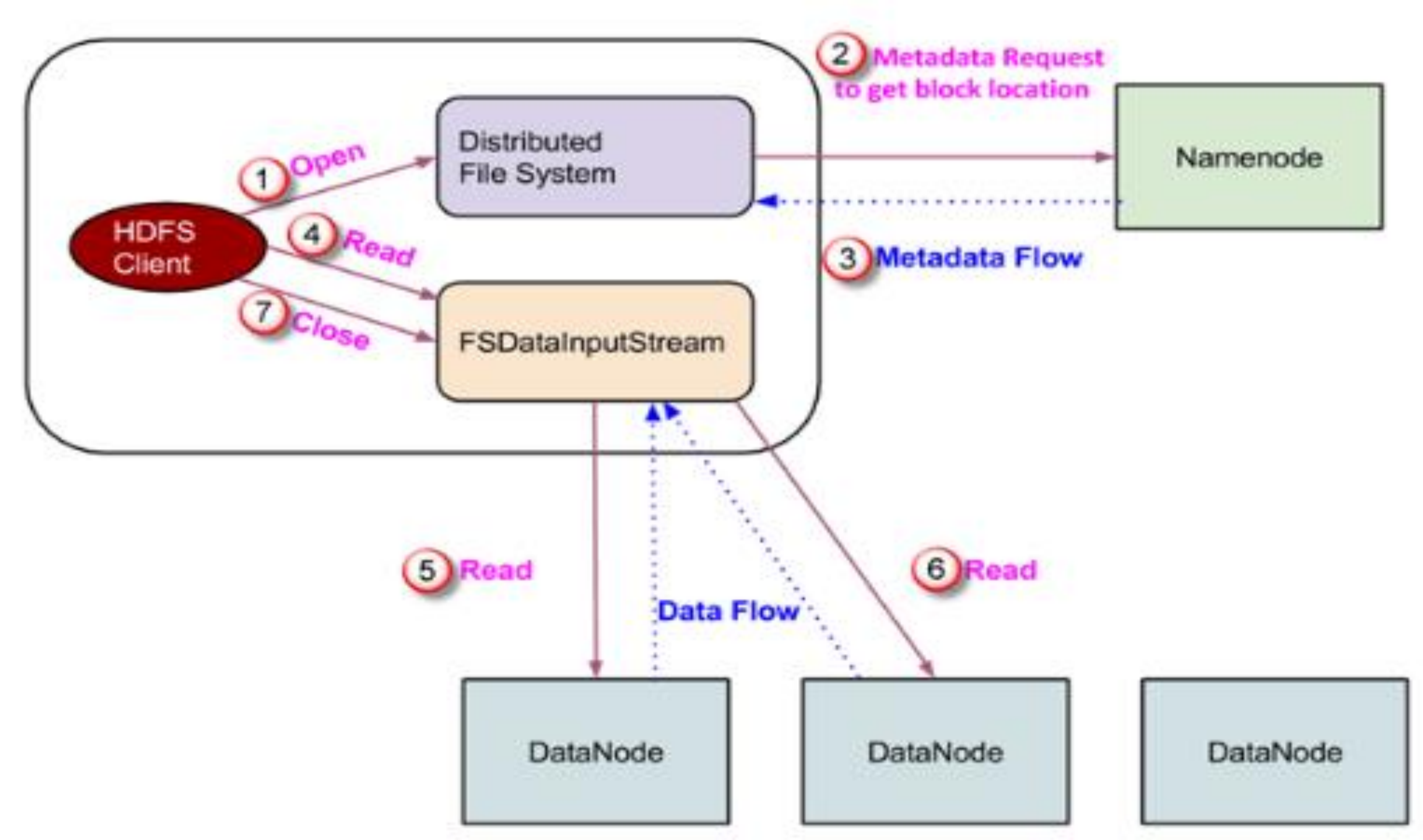
HDFS Architecture



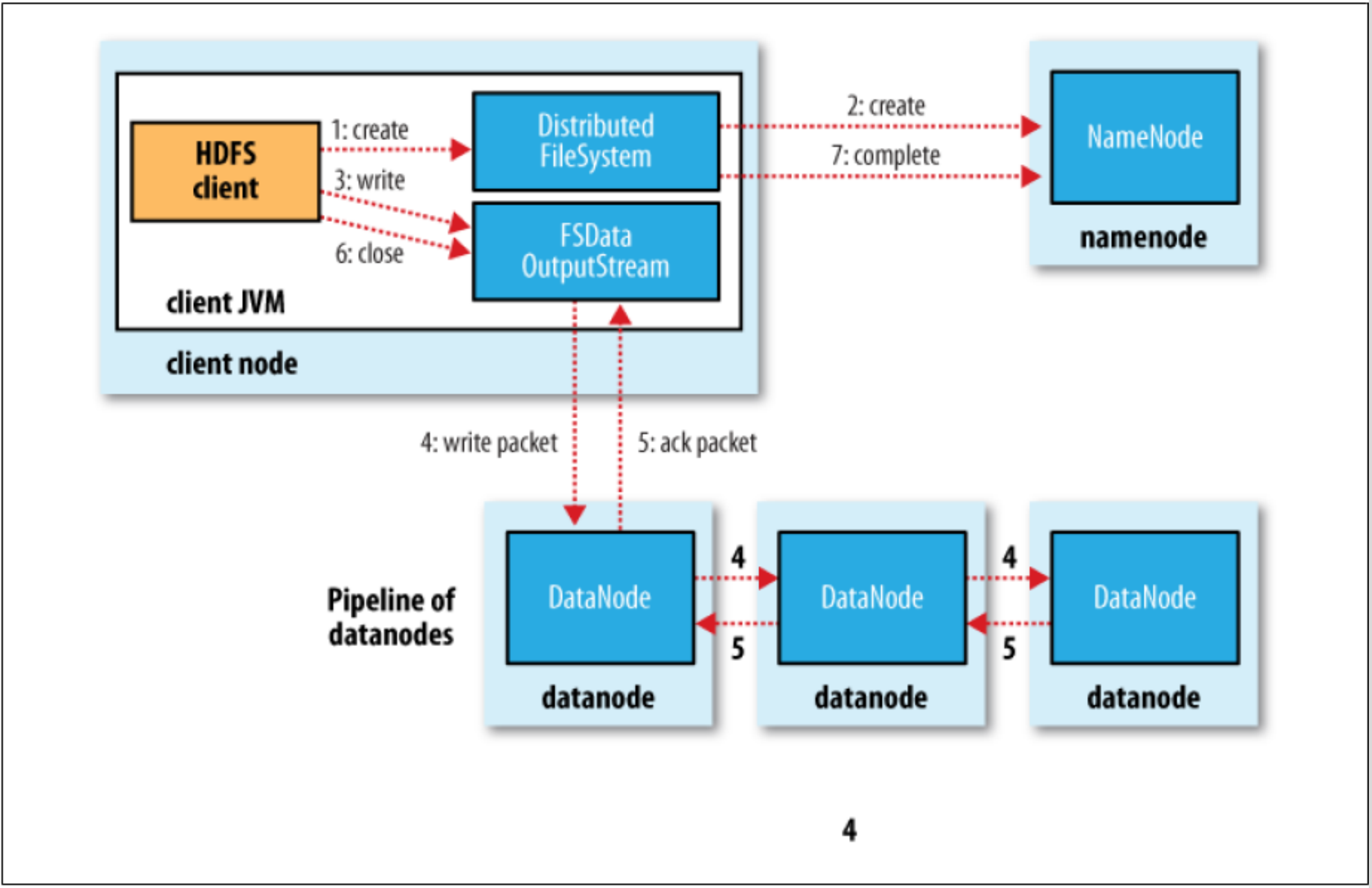
File Metadata (master)

File Data (workers)

Reading a file



Writing a File

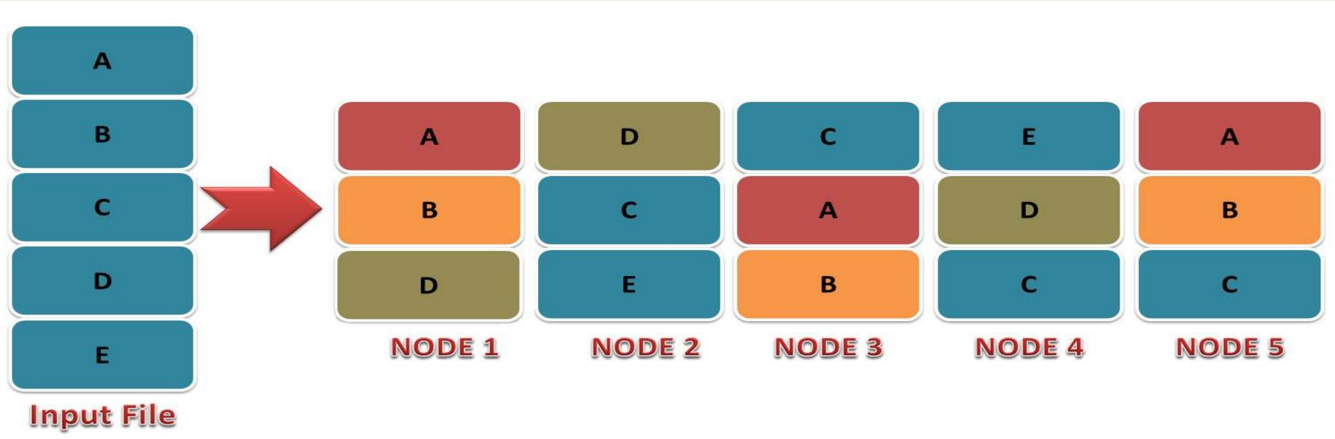


HDFS Fault Tolerance - 1

- Fault tolerance in Hadoop HDFS refers to the working strength of a system in unfavorable conditions and how that system can handle such a situation
- HDFS is highly fault tolerant and handled faults.
    - There are approaches used for handling faults of data nodes or disks holding data blocks.
        - These could be based on replication, to a replication factor till Hadoop 3. So whenever if any machine/disk in the cluster goes down, then data is accessible from other machines/disk in which the same copy of data was created
        - Using Erasure coding in Hadoop 3
    - There are also approaches to handle faults of NameNode and the availability of the metadata pointing to the data blocks

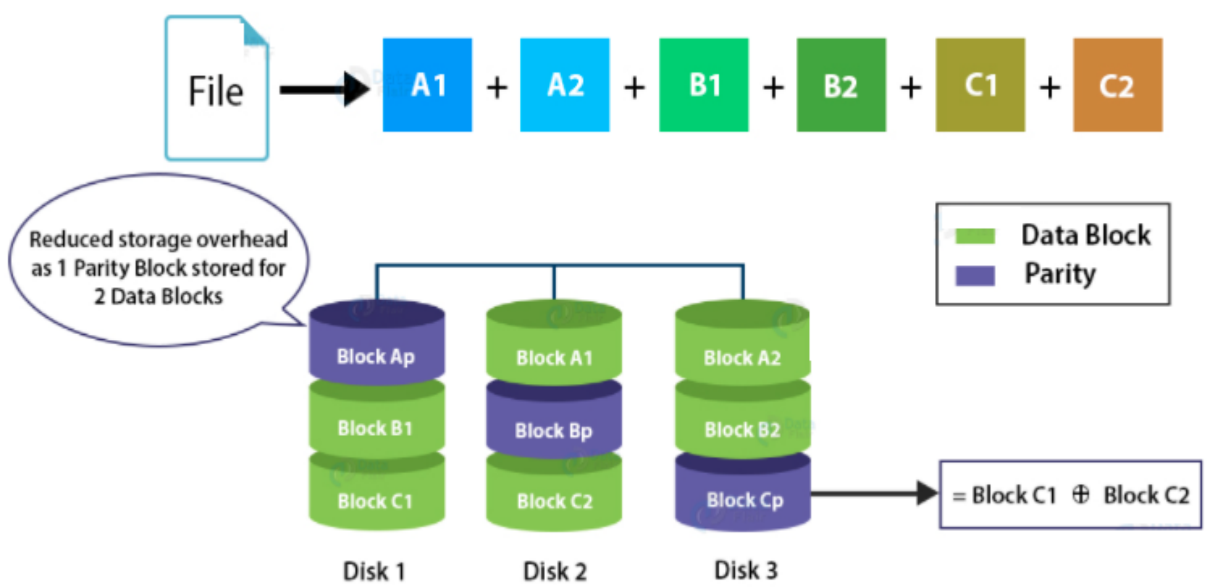
HDFS Fault Tolerance - 2

- Fault tolerance of data using Replicas is achieved by creating a replica of users' data based machines in the HDFS cluster.
- So if any machine in the cluster goes down, then data is accessible from other machines in which the same copy of data was created



HDFS Fault Tolerance - 3

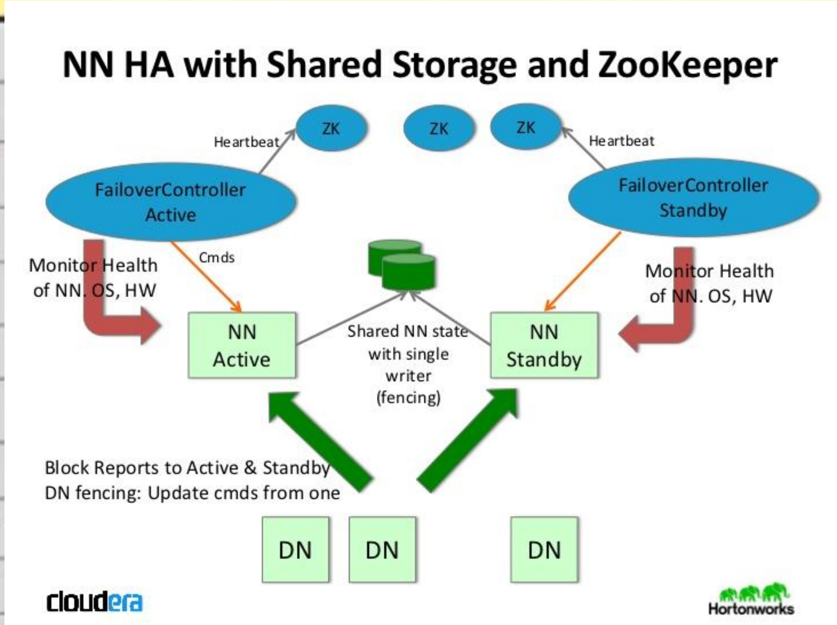
- Fault tolerance of data could also be based on Erasure coding
- Erasure coding works similar to RAID by striping the file into small units of sequential blocks and storing them consecutively on various disks.
- For each strip of the original dataset, a certain number of parity cells are calculated using erasure coding algorithm called encoding and stored. If any of the machines fails, the block can be recovered from the parity cell.
- The error in any striping cell can be recovered from the calculation based on the remaining data and parity cells; the process known as decoding.



- Erasure coding reduces the storage overhead to 50%.

HDFS Failover

- Failover Controller
  - Handles transition from active → standby
  - Failover controllers are pluggable,
  - ZooKeeper to ensure that only one namenode is active.
- Manual Trigger for maintenance
- Ungraceful failure
  - Is it a real failure?
  - Slow network –active treated


NN HA with Shared Storage and ZooKeeper
ZK
ZK
ZK
Heartbeat
Heartbeat
FailoverController Active
FailoverController Standby
Monitor Health of NN. OS, HW
Cmds
NN Active
Shared NN state with single writer (fencing)
NN Standby
Monitor Health of NN. OS, HW
Block Reports to Active & Standby
DN fencing: Update cmds from one
DN
DN
DN
cloudera
Hortonworks

HDFS uses Zookeeper for

- Failure Detection
  - Namenode maintains persistent session with ZK
  - If Machine crashes
    - Session would expire
    - Notify the other NameNode of the crash
- Active Namenode Election
  - On crash, other node takes exclusive lock
  - Indicating that it is the leader

HDFS High Availability (post 2.x)

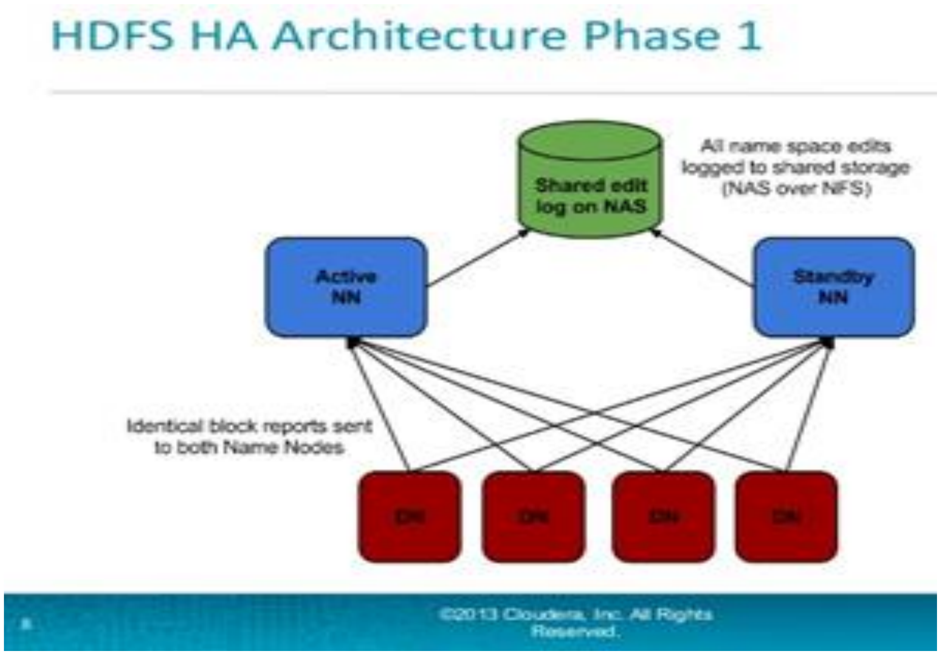
- Make the name node resilient to failure
- Should recover quickly

Configure

- Active –Standby configuration
- Edit logs
  - Store on shared storage
- Block Mappings
  - Stored in memory

On Failure

- Standby takes over using shared edit logs and in-memory block mappings


HDFS HA Architecture Phase 1
Shared edit log on NAS
All name space edits logged to shared storage (NAS over NFS)
Active NN
Standby NN
Identical block reports sent to both Name Nodes
DN
DN
DN
DN

©2013 Cloudera, Inc. All Rights Reserved.

Name node — Edit logs — Shared storage

Datanodes — block reports to both namenodes — Block mappings stored in memory

Clients — Transparency — Provide failover

Where HDFS doesn't s

Low Latency Data Access:

Applications that require low-latency access to data, in the tens of milliseconds range.

The high throughput offered by HDFS may be at the cost of latency.

Lots of Small Files:

namenode holds filesystem metadata in memory,

limit to the number of files in a filesystem is governed by the amount of memory on the namenode.

IAs a rule of thumb, each file, directory, and block takes about 150 bytes.

- Million files → 300MB of memory.
- Billion files → ?

Multiple writers, arbitrary file modifications:

Files in HDFS may be written to by a single writer.

Writes are always made at the end of the file.

No support for multiple writers, or for modifications at arbitrary offsets in the file.

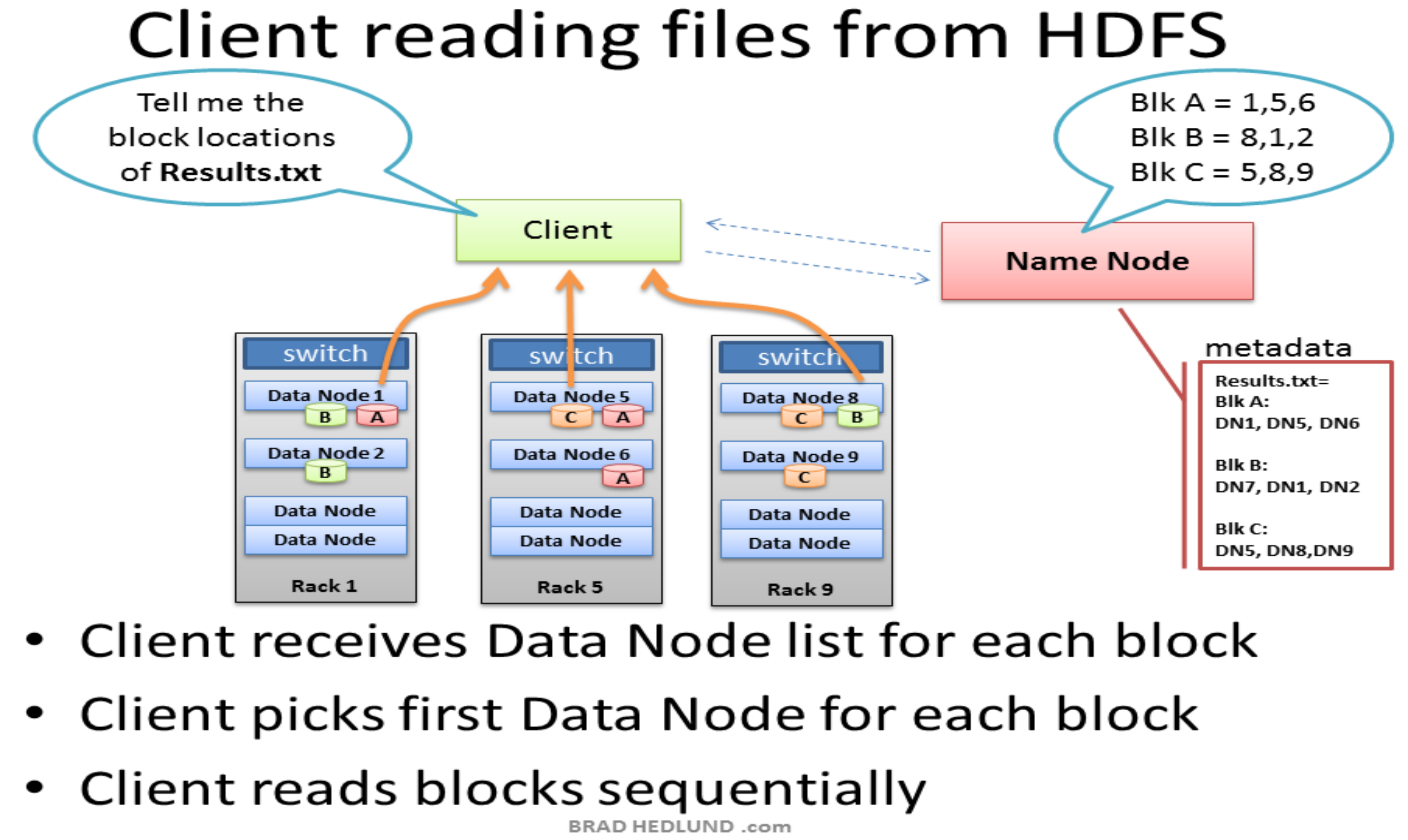
Not Posix compliant

Can't mount a standard FS commands

Need a separate client

Thank you

Reading a file



Client reading files from HDFS

Results.txt=
Blk A:
DN1, DN5, DN6

Blk B:
DN7, DN1, DN2

Blk C:
DN5, DN8,DN9

- Client receives Data Node list for each block
- Client picks first Data Node for each block
- Client reads blocks sequentially

BRAD HEDLUND .com