Unit 4: HDFS

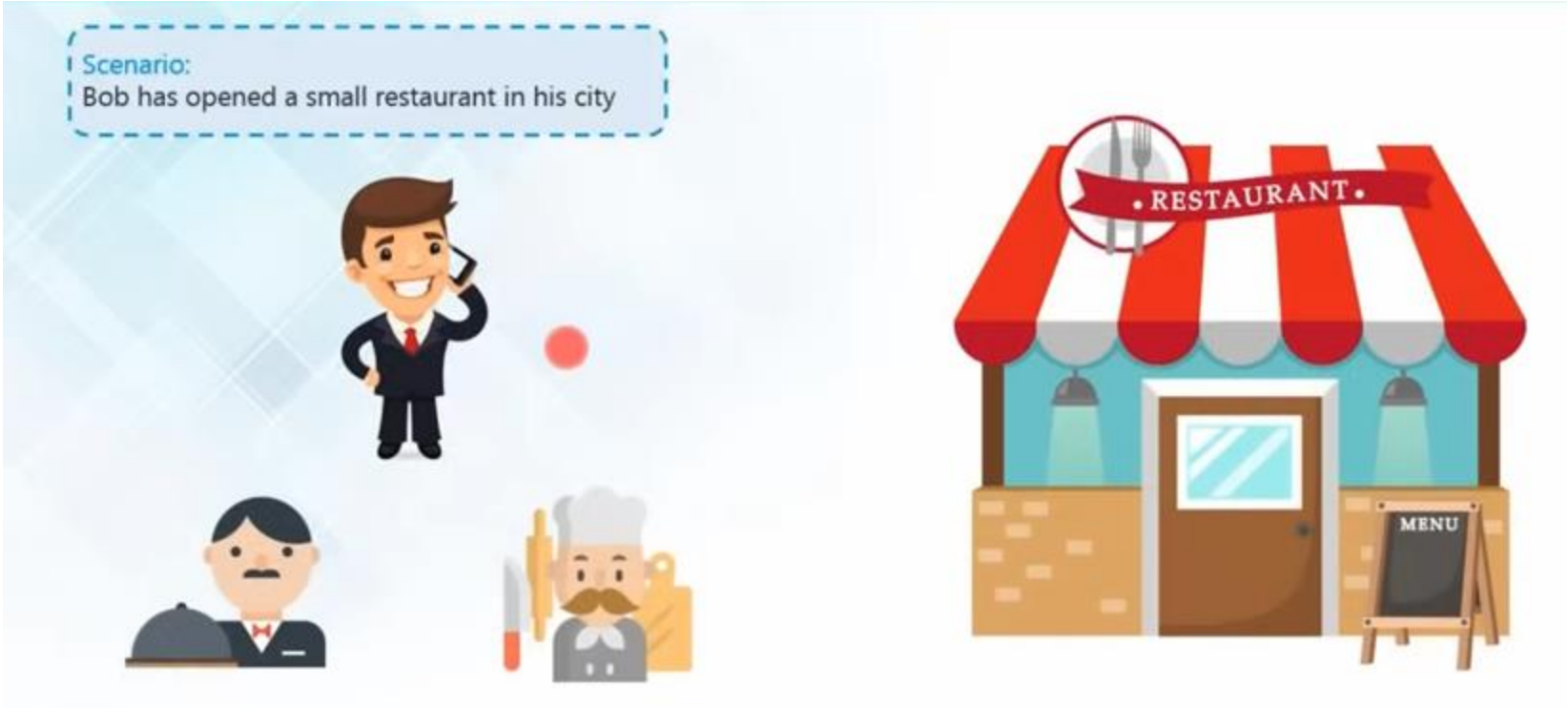
Problems with Traditional Approach

Storing huge and exponentially growing datasets

Processing data having complex structure (structured, un-structured, semi-structured)

Bringing huge amount of data to computation unit becomes a bottleneck

Analogy of Hadoop



Analogy of Hadoop

Traditional Scenario:

2 orders per hour

Single Cook

Food Shelf

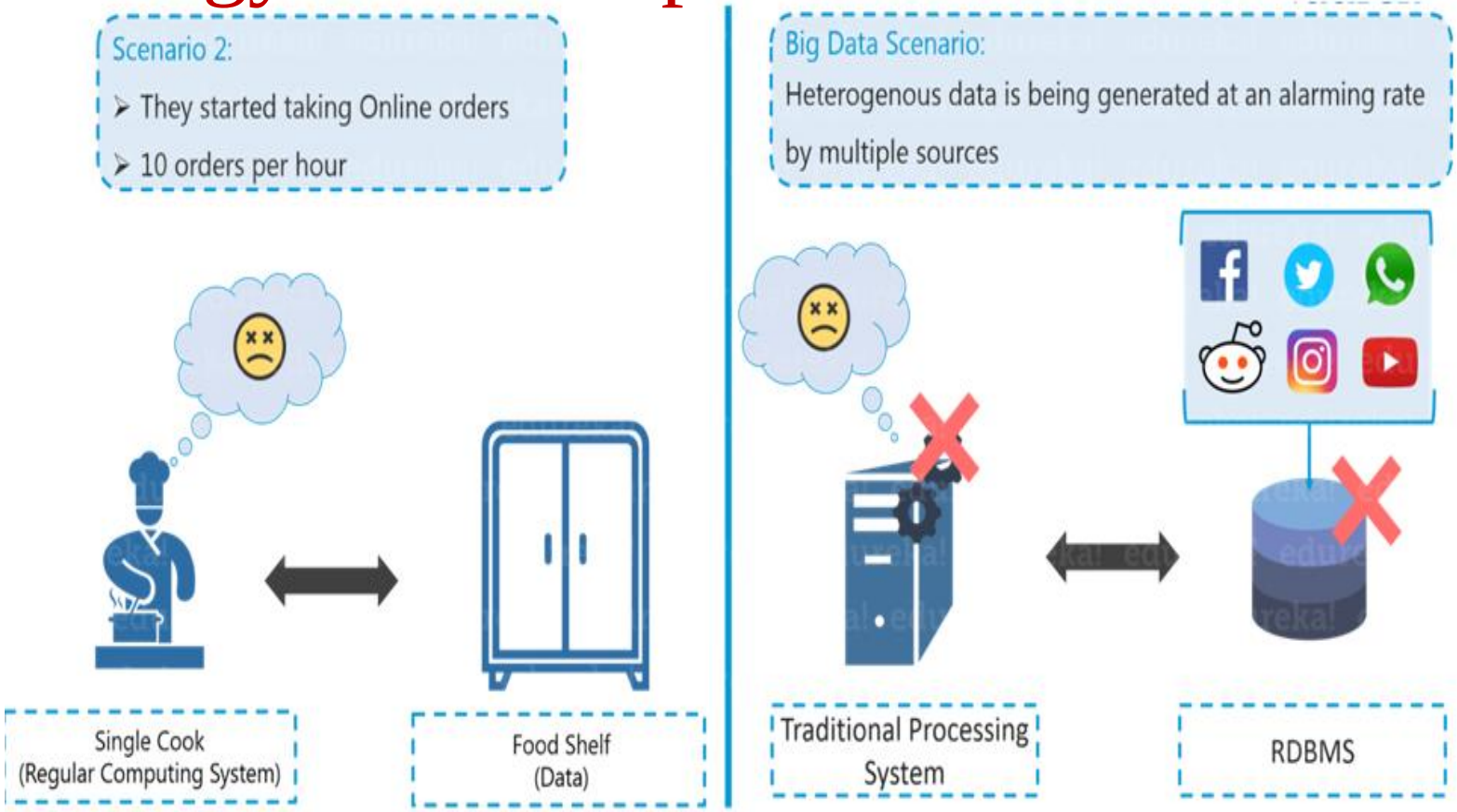
Traditional Scenario:

Data is generated at a steady rate and is structured in nature

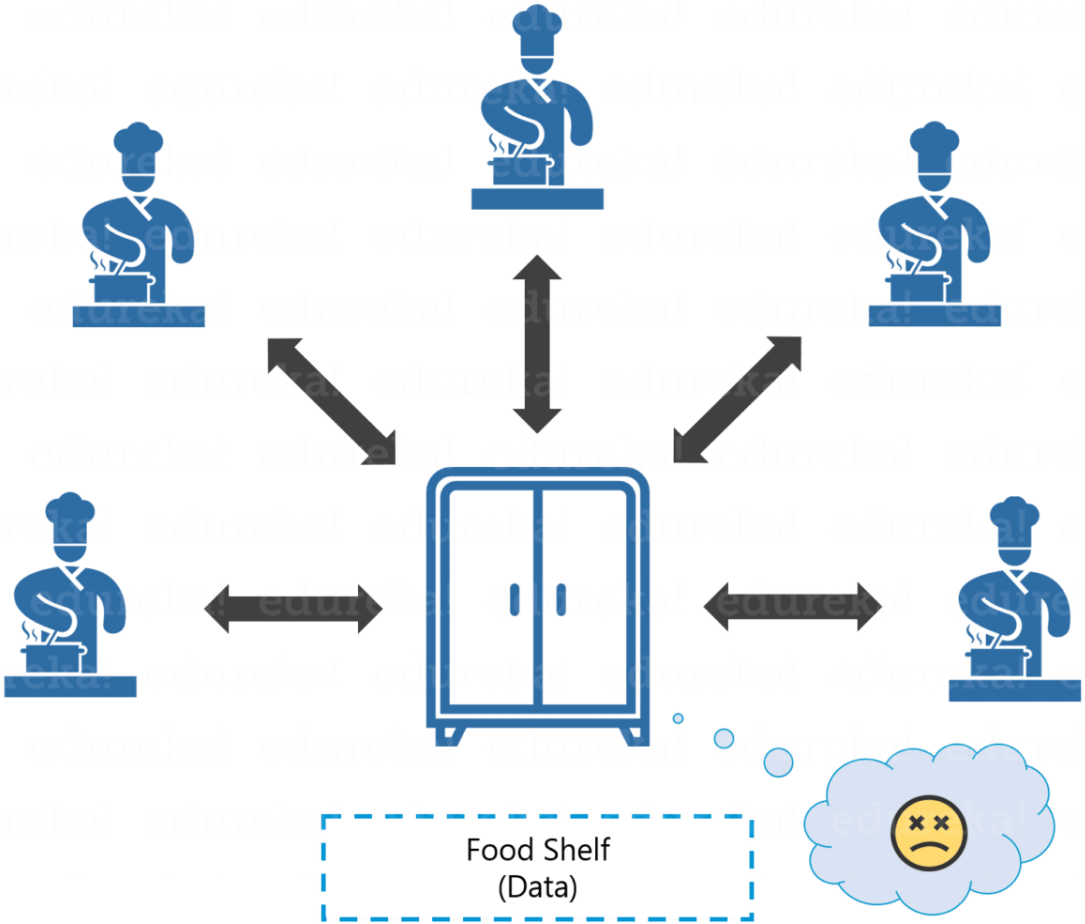
Traditional Processing System

RDBMS

Analogy of Hadoop



Analogy of Hadoop



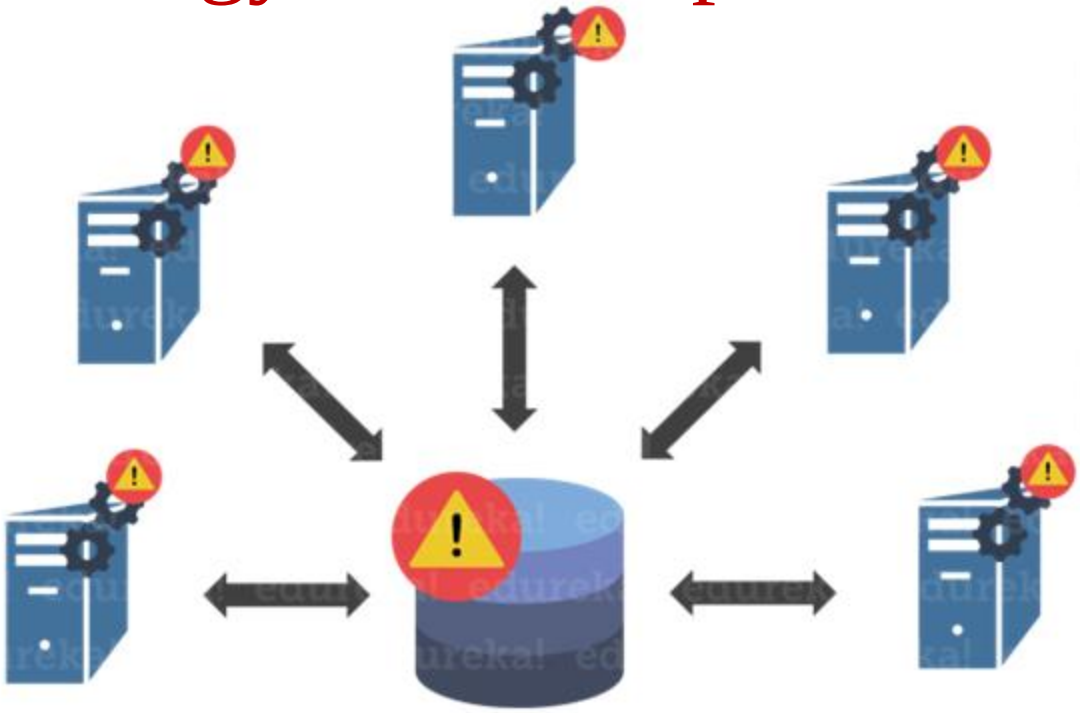
Scenario:

Multiple Cook cooking food

Issue:

Food Shelf becomes the BOTTLENECK

Analogy of Hadoop



Data Warehouse

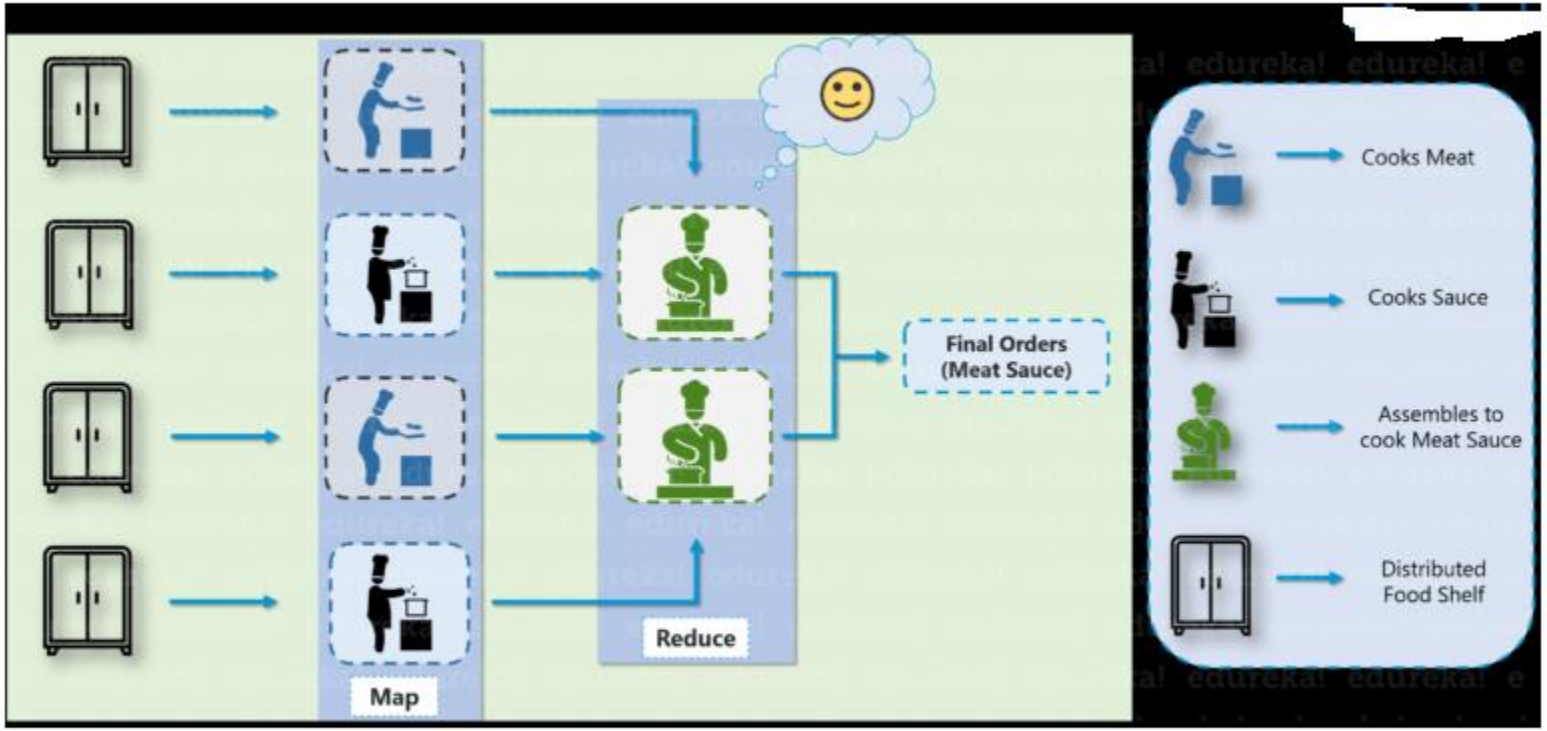
Scenario:

Multiple Processing Unit for data processing

Issue:

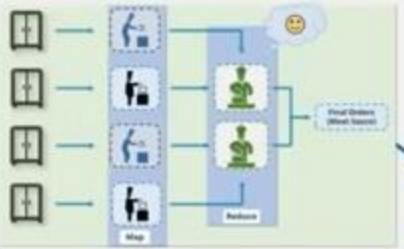
Bringing data to processing generated lot of Network overhead

Analogy of Hadoop



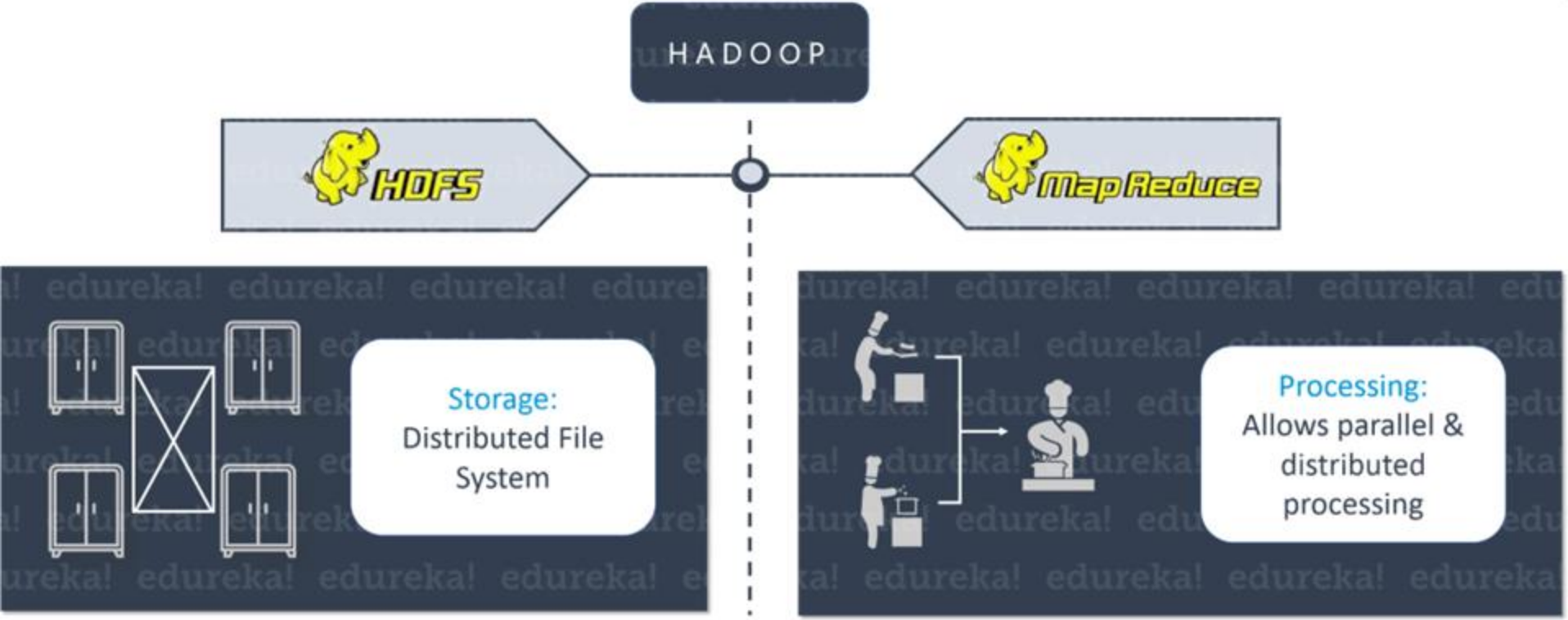
Data locality

Need of a Framework



Do we have a framework that works like that ?

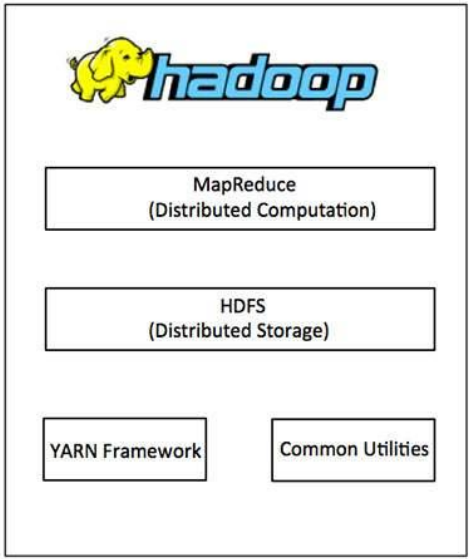
Hadoop



Hadoop cluster

Hadoop

- Hadoop is an open source framework from Apache and is used to store ,process and analyze data which are very huge in volume.
- Hadoop has two major layers namely –
  - Processing/Computation layer (MapReduce)
  - Storage layer (Hadoop Distributed File System)



- MapReduce: MapReduce is a **parallel programming model** for writing distributed applications devised at Google for efficient processing of large amounts of data (multi-terabyte data-sets), on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. The MapReduce program runs on Hadoop which is an Apache open-source framework.
- Hadoop Distributed File System: The **HDFS** is based on the **GFS** and provides a distributed file system that is designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. It is highly fault-tolerant and is designed to be deployed on low-cost hardware. It provides high throughput access to application data and is suitable for applications having large datasets.
- Apart from the above-mentioned two core components, Hadoop framework also includes the following two modules −
- **Hadoop Common** − These are Java libraries and utilities required by other Hadoop modules.
- **Hadoop YARN** − This is a framework for job scheduling and cluster resource management.

What is HDFS

- Hadoop comes with a distributed file system called HDFS.
- In HDFS data is distributed over several machines and replicated to ensure their durability to failure and high availability to parallel application.
- It is **cost effective** as it uses **commodity hardware**. It involves the concept of blocks, data nodes and node name.

Where to use HDFS

- **Very Large Files:** Files should be of hundreds of megabytes, gigabytes or more.

- **Streaming Data Access:** The time to read whole data set is more important than latency in reading the first. HDFS is built on write-once and read-many-times pattern.

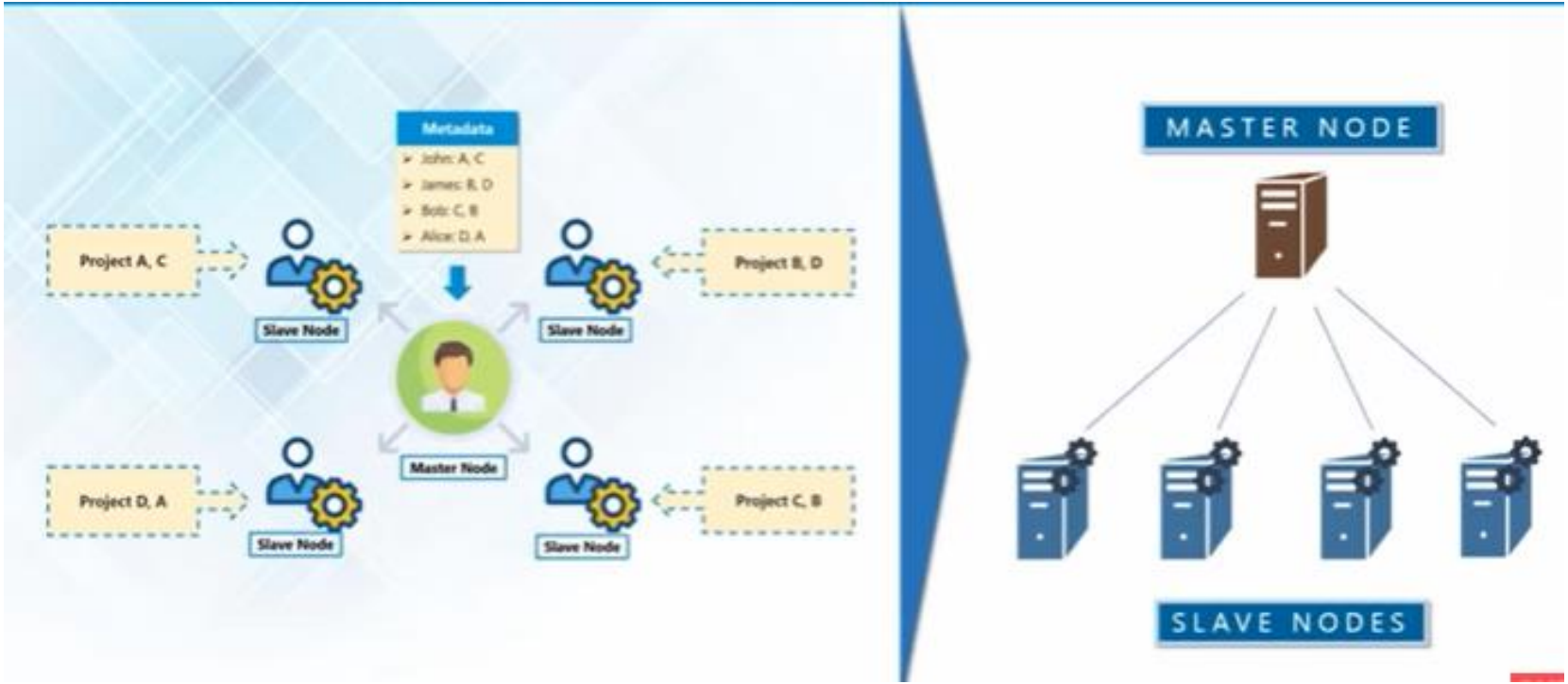
- **Commodity Hardware:** It works on low cost hardware.

Where to not use HDFS

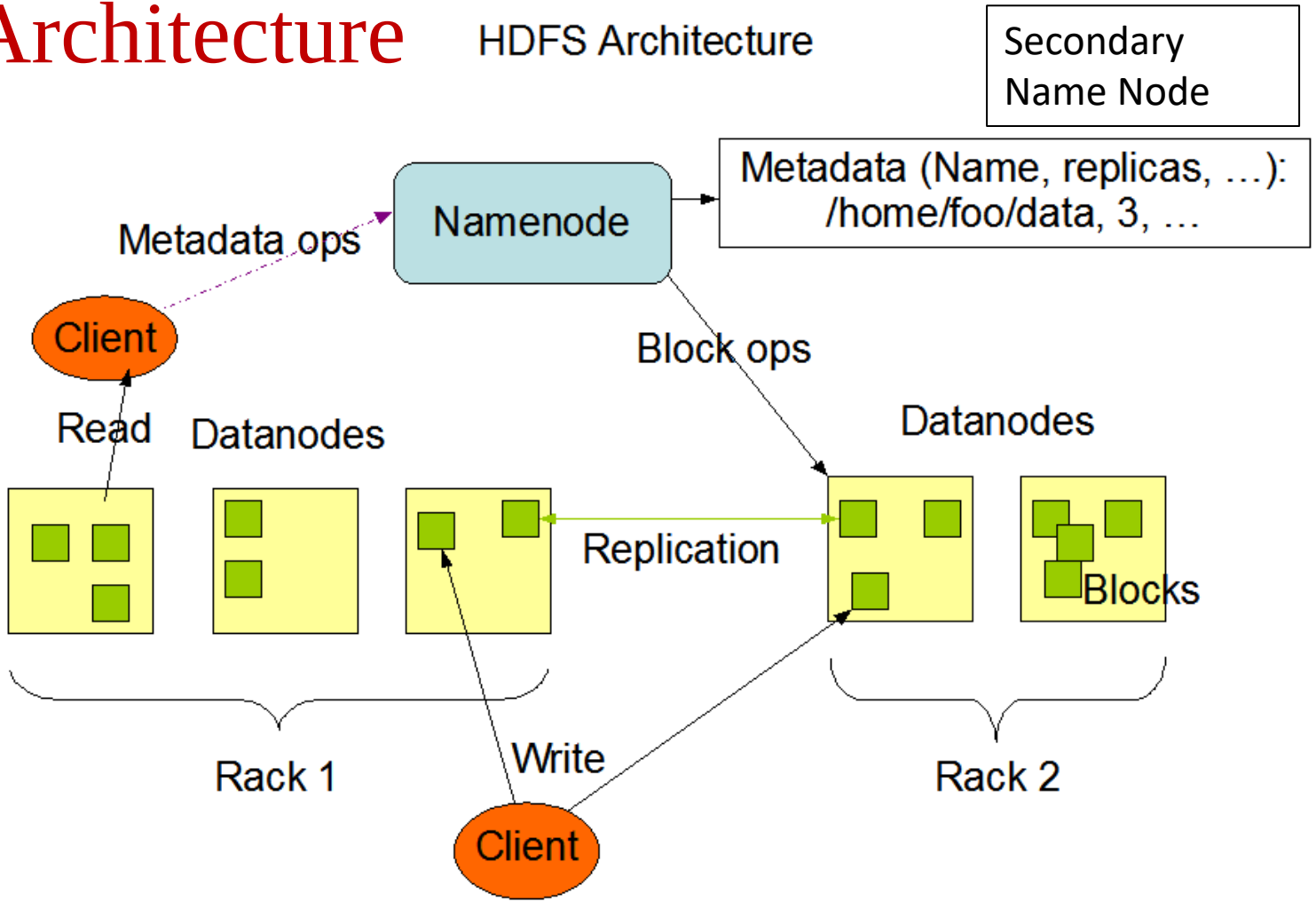
- **Low Latency data access:** Applications that require very less time to access the first data should not use HDFS as it is giving importance to whole data rather than time to fetch the first record.
- **Lots Of Small Files:** The name node contains the metadata of files in memory and if the files are small in size it takes a lot of memory for name node's memory which is not feasible.
- **Multiple Writes:** It should not be used when we have to write multiple times.

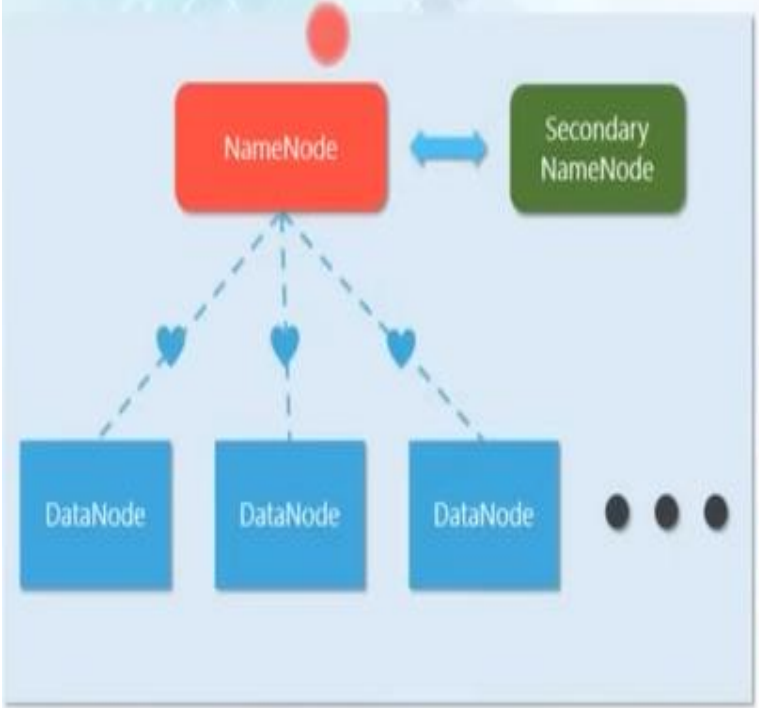
HDFS

- HDFS is the file system which is used in Hadoop based distributed file system. The Hadoop is an open-source distributed computing framework and provided by Apache. Many network stations use it to create systems such as Amazon, facebook.
- The Hadoop cores are Mapreduce and HDFS.
- The HDFS is a master and slaver framework. It consists of **single name node as the master node** and **a number of data nodes** as the slave nodes.
- **The name node manages the file system namespace** and regulates access to files by clients.
- The data nodes are distributed, one data node per machine in the cluster, which manage data blocks attached to the machines where they run.


Metadata
John: A, C
James: B, D
Bob: C, B
Alice: D, A
Project A, C
Project B, D
Project D, A
Project C, B
Slave Node
Slave Node
Slave Node
Slave Node
Master Node
MASTER NODE
SLAVE NODES

HDFS Architecture





NameNode:

- Maintains and Manages DataNodes
- Records metadata i.e. information about data blocks e.g. location of blocks stored, the size of the files, permissions, hierarchy, etc.
- Receives heartbeat and block report from all the DataNodes

DataNode:

- Slave daemons
- Stores actual data
- Serves read and write requests from the clients

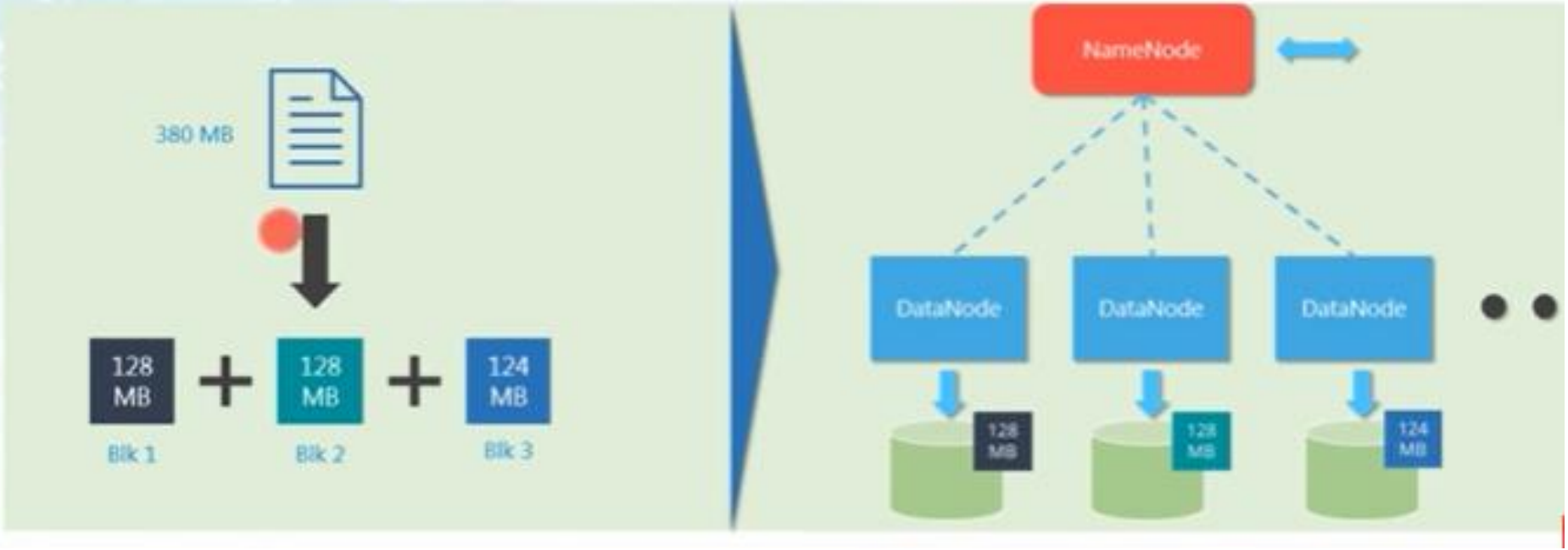
- The namenode executes the operations on file system namespace and maps data blocks to data nodes.
- The data nodes are responsible for **serving read and write** requests from clients and perform **block operations** upon instructions from name node
- HDFS **distributes data chunks** and replicas across the server for higher performance, load-balancing and resiliency.
- With data distributed across all servers, any server may be participating in the reading, writing, or computation of a data-block at any time. HDFS replicates file blocks for fault tolerance.
- The name node makes all decisions concerning block replication.

HDFS Concepts

1.Blocks: A Block is the minimum amount of data that it can read or write.

- HDFS blocks are 128 MB by default and this is configurable.
- Files in HDFS are broken into block-sized chunks, which are stored as independent units.
- Unlike a file system, if the file is in HDFS is smaller than block size, then it does not occupy full block size, i.e. 5 MB of file stored in HDFS of block size 128 MB takes 5MB of space only.
- The HDFS block size is large just to minimize the cost of seek.

- Each file is stored on HDFS as blocks
- The default size of each block is 128 MB in Apache Hadoop 2.x (64 MB in Apache Hadoop 1.x)



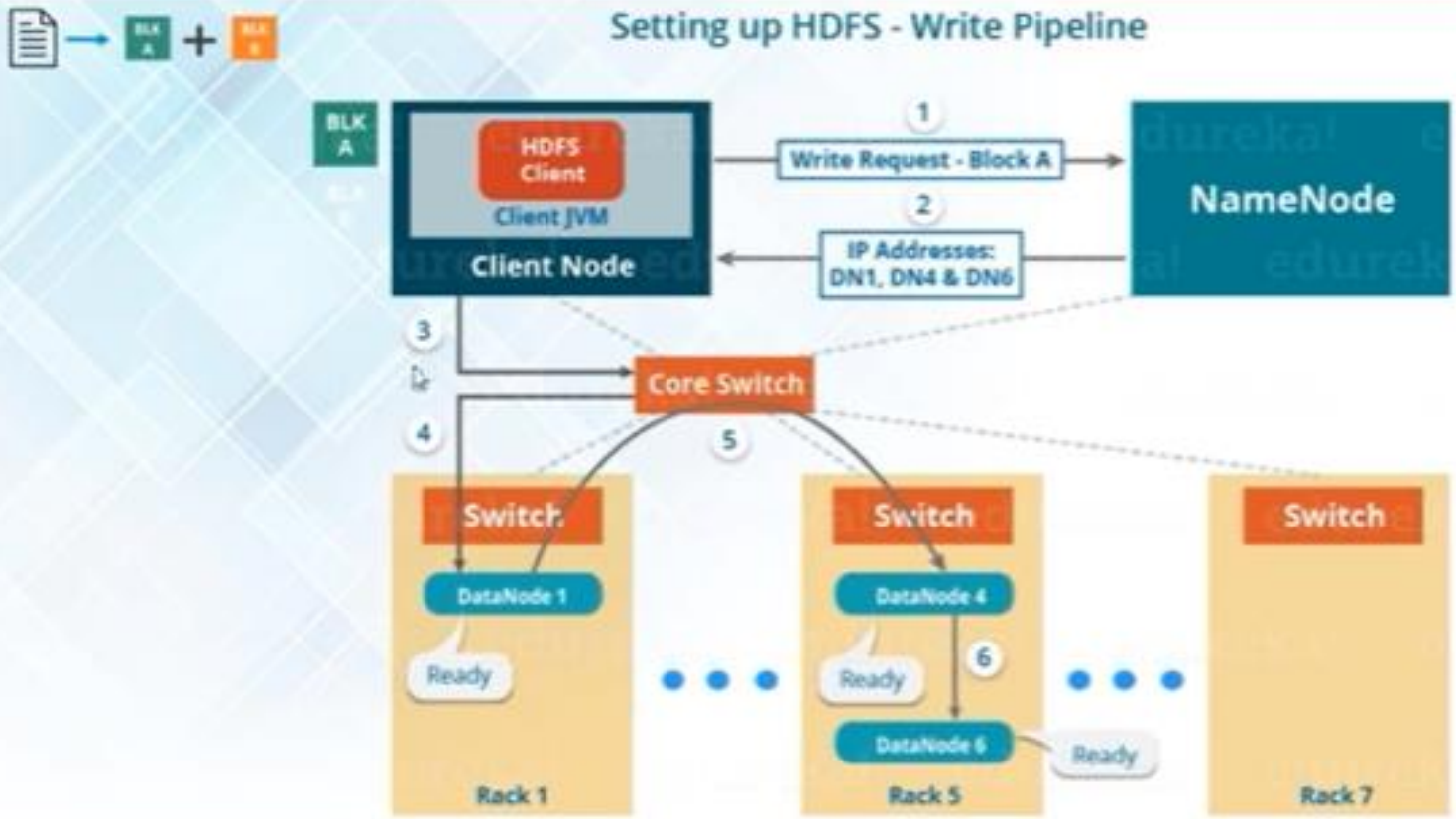
Replication factor

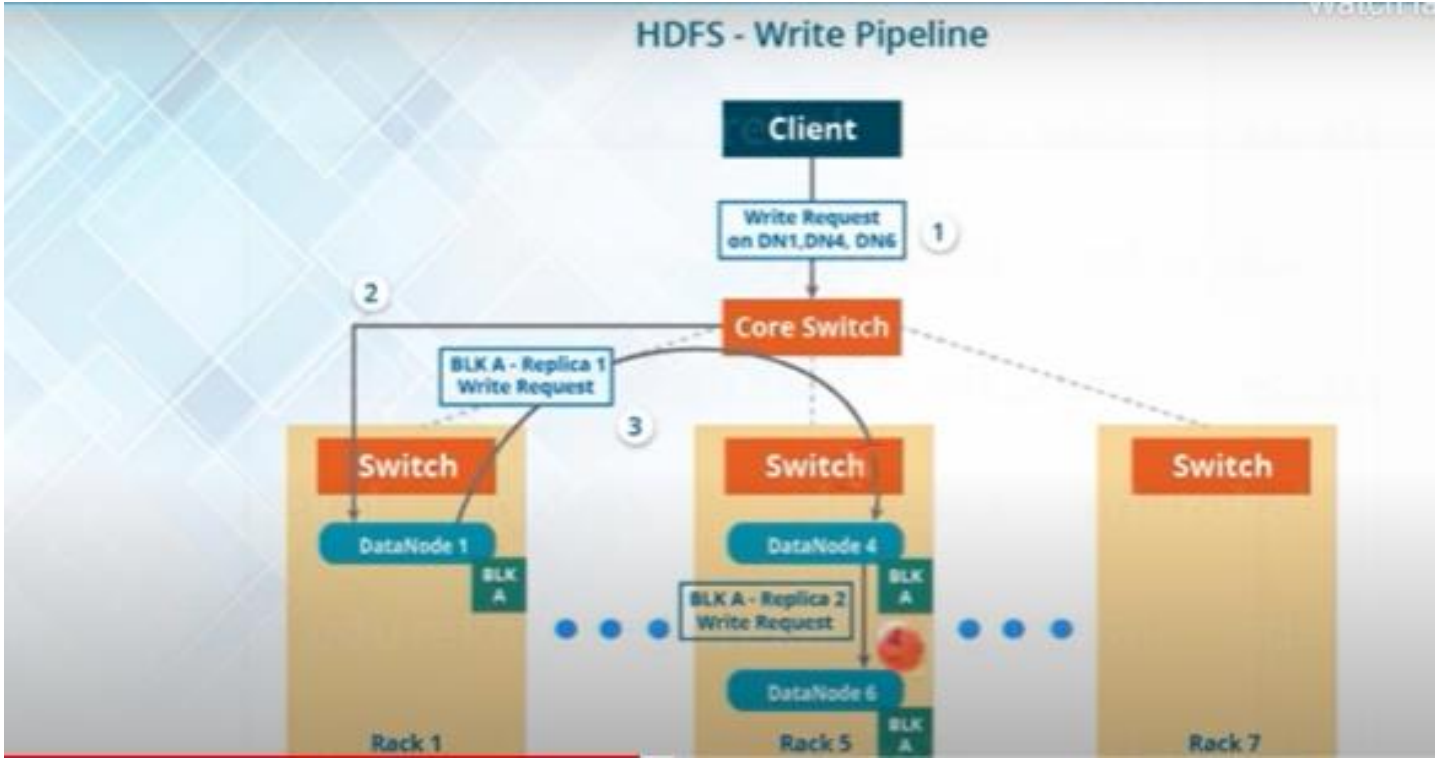
- **Name Node:** HDFS works in master-worker pattern where the name node acts as master.
- Name Node is controller and manager of HDFS as it knows the status and the metadata of all the files in HDFS;
- the metadata information being file permission, names and location of each block.
- The metadata are small, so it is stored in the memory of name node,allowing faster access to data.
- Moreover the HDFS cluster is accessed by multiple clients concurrently, so all this information is handled by a single machine. The file system operations like opening, closing, renaming etc. are executed by it.

3. Data Node: They store and retrieve blocks when they are told by client or name node.

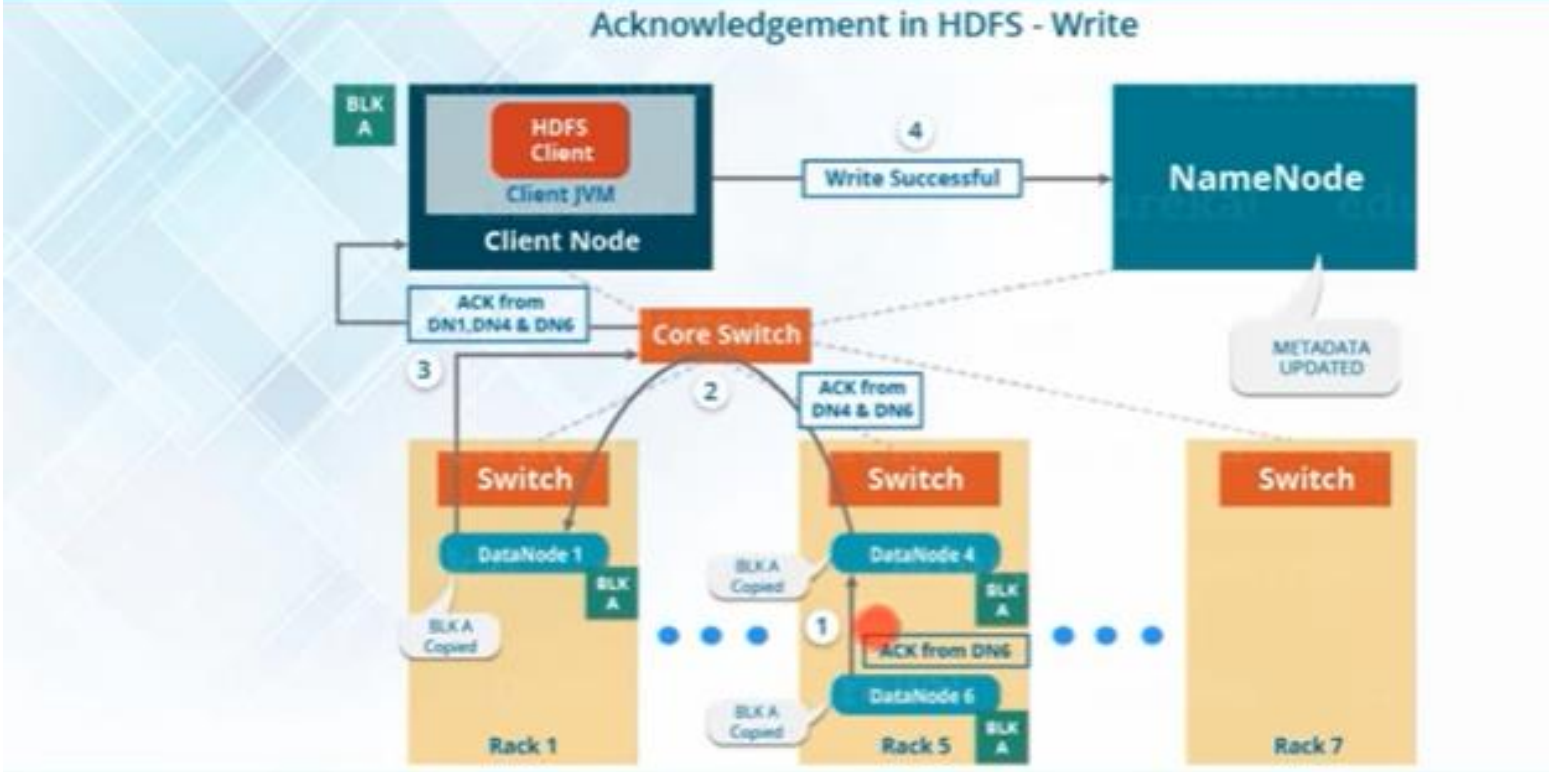
- They report back to name node periodically, with list of blocks that they are storing.
- The data node being a commodity hardware also does the work of block creation, deletion and replication as stated by the name node.
- Data nodes perform read-write operations on the file systems, as per client request.

HDFS Write Mechanism – Pipeline Setup

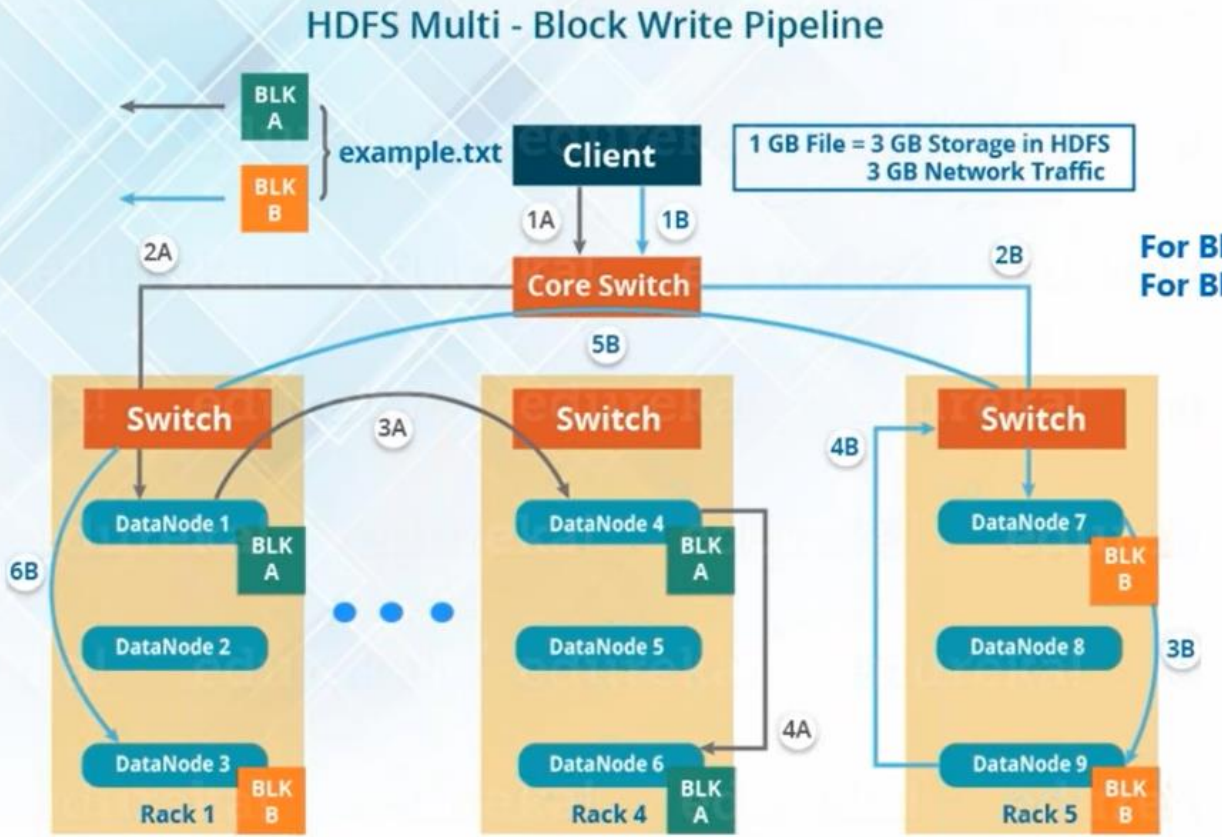



HDFS - Write Pipeline
Client
Write Request on DN1,DN4, DN6
1
2
Core Switch
BLK A - Replica 1 Write Request
3
Switch
Switch
Switch
DataNode 1
BLK A
DataNode 4
BLK A
BLK A - Replica 2 Write Request
DataNode 6
BLK A
Rack 1
Rack 5
Rack 7

HDFS Write Mechanism - Acknowledgement



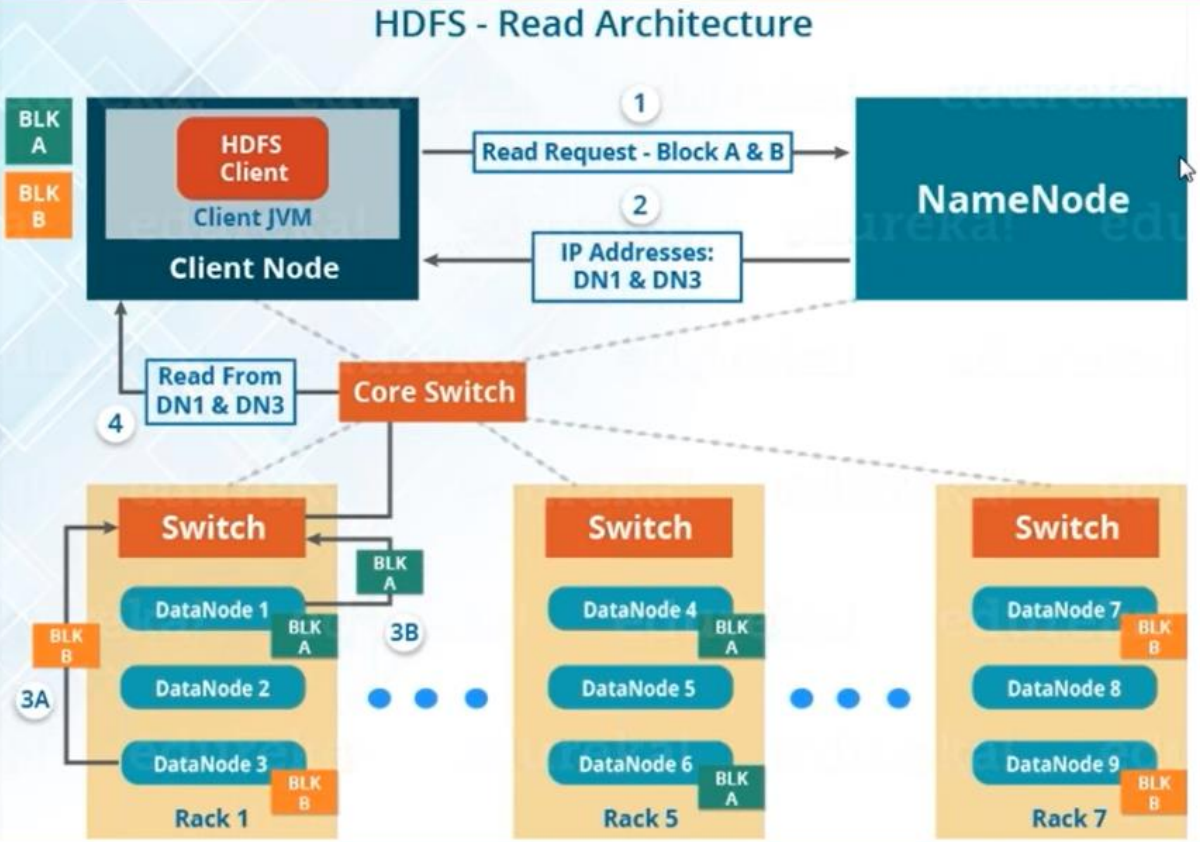
HDFS Multi-Block Write Mechanism



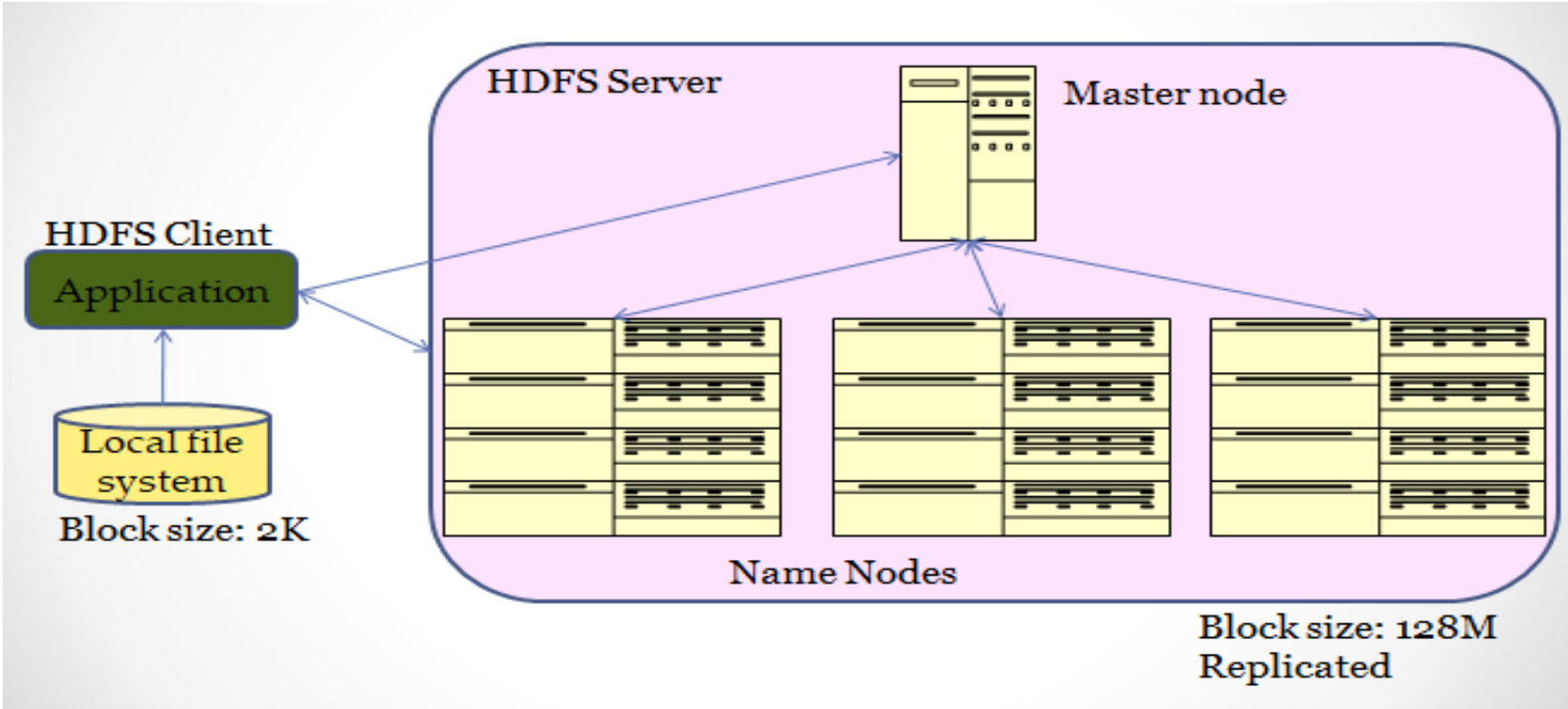
For Block A: 1A -> 2A -> 3A -> 4A
For Block B: 1B -> 2B -> 3B -> 4B -> 5B -> 6B

1st copy of block is created parallel and 2nd,3rd copy sequentially by datanodes

HDFS Read Mechanism



Hadoop Distributed File System



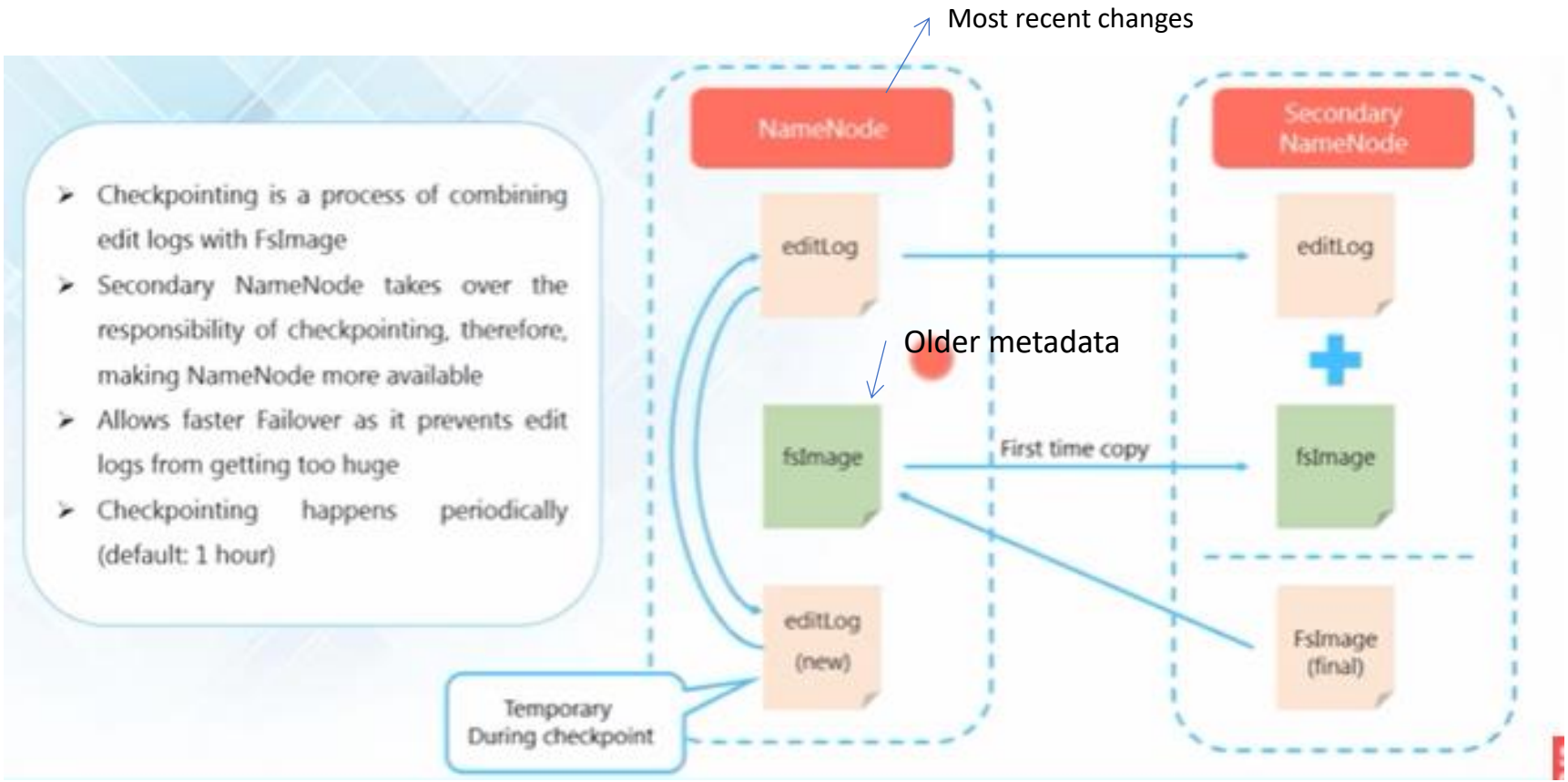
HDFS

- Where not to use HDFS
  - **Low Latency data access:** Applications that require very less time to access the first data should not use HDFS as it is giving importance to **whole data rather than time** to fetch the first record.
  - **Lots Of Small Files:** The name node contains the metadata of files in memory and if the files are small in size it takes a lot of memory for name node's memory which is not feasible.
  - **Multiple Writes:**It should not be used when we have to write multiple times.

HDFS

- Since all the metadata is stored in name node, it is very important.
- If it fails the file system can not be used as there would be no way of knowing how to reconstruct the files from blocks present in data node.
- To overcome this, the concept of secondary name node arises.
- **Secondary Name Node:** It is a separate physical machine which acts as a helper of name node. It performs periodic check points. It communicates with the name node and take snapshot of meta data which helps minimize downtime and loss of data.

Secondary name node



Features of HDFS

•**Highly Scalable -** HDFS is highly scalable as it can scale hundreds of nodes in a single cluster.
•**Replication -** Due to some unfavorable conditions, the node containing the data may be loss. So, to overcome such problems, HDFS always maintains the copy of data on a different machine.
•**Fault tolerance -** In HDFS, the fault tolerance signifies the robustness of the system in the event of failure. The HDFS is highly fault-tolerant that if any machine fails, the other machine containing the copy of that data automatically become active.
•**Distributed data storage -** This is one of the most important features of HDFS that makes Hadoop very powerful. Here, data is divided into multiple blocks and stored into nodes.
•**Portable -** HDFS is designed in such a way that it can easily portable from platform to another.

•Goals :

Handling the hardware failure - The HDFS contains multiple server machines. Anyhow, if any machine fails, the HDFS goal is to recover it quickly.
Streaming data access - The HDFS applications usually run on the general-purpose file system. This application requires streaming access to their data sets.
Coherence Model - The application that runs on HDFS require to follow the write-once-ready-many approach. So, a file once created need not to be changed. However, it can be appended and truncate.

GFS Limitation	HDFS Solution	Impact
Single Point of Failure (Master Node Dependency) □	HDFS introduced Secondary NameNode & High Availability (HA) mode.	□ Reduces risk of complete failure if NameNode crashes.
Inefficient for Small Files □	HDFS Federation & Hadoop Archive (HAR) optimize small file storage.	□ Reduces metadata overhead & improves storage efficiency.
High Metadata Memory Usage □	HDFS supports metadata persistence & edit log checkpoints.	□ Scales better than GFS without excessive RAM use.
Fixed 64 MB Chunk Size	HDFS uses larger (128 MB+) blocks, configurable by users.	□ Reduces metadata load & improves performance.
High Storage Overhead (Full Replication) □	HDFS supports Erasure Coding (EC) to reduce storage replication overhead.	□ Uses less storage while maintaining fault tolerance.
Limited Security Features □	HDFS supports Kerberos authentication, Access Control Lists (ACLs), and encryption.	□ More secure for enterprise use (e.g., financial data).
Not Open Source (Google-Internal Use Only)	HDFS is open-source under Apache.	□ Anyone can use, modify, and integrate HDFS.
No Windows Support □	HDFS can run on both Linux and Windows environments.	□ Broader platform compatibility for enterprises.
Slow Recovery from Failures ⚠	HDFS re-replicates blocks faster using Heartbeats & Block Reports.	□ Faster self-healing, reducing downtime.

COMPARATIVE ANALYSIS OF HDFS WITH GFS

Feature	GFS	HDFS	Observation
Main Server	GFS master	NameNode	Master and NameNode store the metadata.
Data Storing Server	GFS chunk server	DataNode	Chunkserver and DataNode are used for accessing data.
Client	GFS client	HDFS client	Client is used for accessing
Technology Support	MapReduce, BigTable, Chubby, Percolator, Pregel, Dremel, Megastore, Spanner and Omega	Hadoop MapReduce, YARN, Hive, HCatalog, HBase, ZooKeeper, Oozie, Pig and Sqoop	GFS from Google and HDFS from Apache carry they own suite of Tools and technologies for storing and analysis .

COMPARATIVE ANALYSIS OF HDFS WITH GFS

Key Point	HDFS Framework	GFS Framework
Objective	Main objective of HDFS to handle the Big-Data	Main objective of HDFS to handle the Big-Data
Language used to Develop	Java Language	C, CPP Language
Implemented by	Open source community, Yahoo, Facebook, IBM	Google
Platform	Work on Cross-platform	Work on Linux
License by	Apache	Proprietary or design by google for its own used.
Files Management	HDFS supports a traditional hierarchical directories data structure [9].	GFS supports a hierarchical directories data structure and access by path names [9].
Types of Nodes used	NameNode and DataNode	Chunk-server and MasterNode

Hardware used	Commodity Hardware or Server	Commodity Hardware or Server
Append Opration	Only supports append operation	supports append operation and we can also append base on offset.
Database Files	Hbase	Bigtable is the database
Delete Opration and Garbage Collection	First, deleted files are renamed and store in particular folder then finally remove using garbage collection method.	GFS has unique garbage collection method in which we cannot reclaiminstantly. It will rename the namespace It will delete after the 3 days during the second scaned.
Default size	HDFS has by default DataNode size 128 MB but it can be change by the user	GFS has by default chunk size 64 MB but it can be change by the user
Snapshots	HDFS allowed upto 65536 snapshots for each directory in HDFS 2.	In GFS Each directories and files can be snapshotted.
Meta-Data	Meta-Data information managed by NameNode.	Meta-Data information managed by MasterNode.
Data Integrity	Data Integrity maintain in between NameNode and DataNode.	Data Integrity maintain in between MasterNode and Chunk-Server.
Replication	There are two time replicas created by default in GFS [10].	There are three time replicas created by default in GFS [10].
Communication	Pipelining is used to data transfer over the TCP protocol.	RPC based protocol used on top of TCP\IP.
Cache management	HDFS provide the distributed cache facility using Mapreduse framework	GFS does not provide the cache facility

Feature/criteria	Google file system	Hadoop distributed file system
Main goal	Support large files	Support large files
	Variety of file formats	Variety of file formats
	Huge data storage size (Peta bytes)	Huge data storage size (Peta bytes)
	Data are kept forever	Data are kept forever
Design goal	Data-intensive computing	Data-intensive computing
	Not for normal end-users	Not for normal end-users
Underlying file systems	Only GFS file systems	Underlying file systems might be ext3, ext4 or xfs
Scalability	Cluster based architecture	Cluster based architecture
	Clusters may contains thousands of nodes	Clusters may contains thousands of nodes
Implementations	The GFS is proprietary file system	The HDFS based on Apache Hadoop open-source project used
	Can't be used by any other company	in Facebook, Twitter, LinkedIn, Adobe, A9.com (Amazon)
File serving	Chunk size is 64 MB	Block size is 128 MB. Can be adjustable
Data flow (I/O)	Master server and chunk server	NameNode and DataNode
Internal connections	The TCP connections	The TCP connections
	For data transfer, pipelining is used over TCP connections	Remote Procedure Call (RPC) is used to conduct external communication between clusters and blocks
Cache management	Cache metadata are saved in client's memory	The HDFS has "distributedcache"
	Chunk servers don't need cache file data	DistributedCache files can be private or public
	Linux system caches frequently accessed data in memory	
Files protection and permission	The GFS splits files up and stores it in multiple pieces on multiple machines	The HDFS implements POSIX-like mode permission for files and directories
Replication Strategy	The GFS has two replicas:	The HDFS has an automatic replication rack based system
	Primary replicas and secondary replicas	By default two copies of each block are stored
File system namespace	Files are organized hierarchically in directories and identified by path names.	The HDFS supports a traditional hierarchical file organization. Users or application can create directories to store files inside.
	The GFS is exclusively for Google only	The HDFS also supports third-party file systems such as CloudStore and Amazon Simple Storage Service (S3)
Database	Bigtable	HBase

- Hadoop Architecture | HDFS Architecture
  https://www.youtube.com/watch?v=m9v9lky3zcE
- MapReduce Explained https://www.youtube.com/watch?v=WRr8lK7JFFI
- https://www.edureka.co/blog/apache-hadoop-hdfs-architecture/
- https://www.edureka.co/blog/hadoop-tutorial/
- https://mytechroad.com/google-file-system-gfs-v-s-hdfs/