Parul® University

# Big Data Analytics(303105361)

Prepared By : Himadri Vegad

Parul®
University

DIGITAL LEARNING CONTENT

# CHAPTER-3

# Basics Of Hadoop

# Big Data – The Characteristics

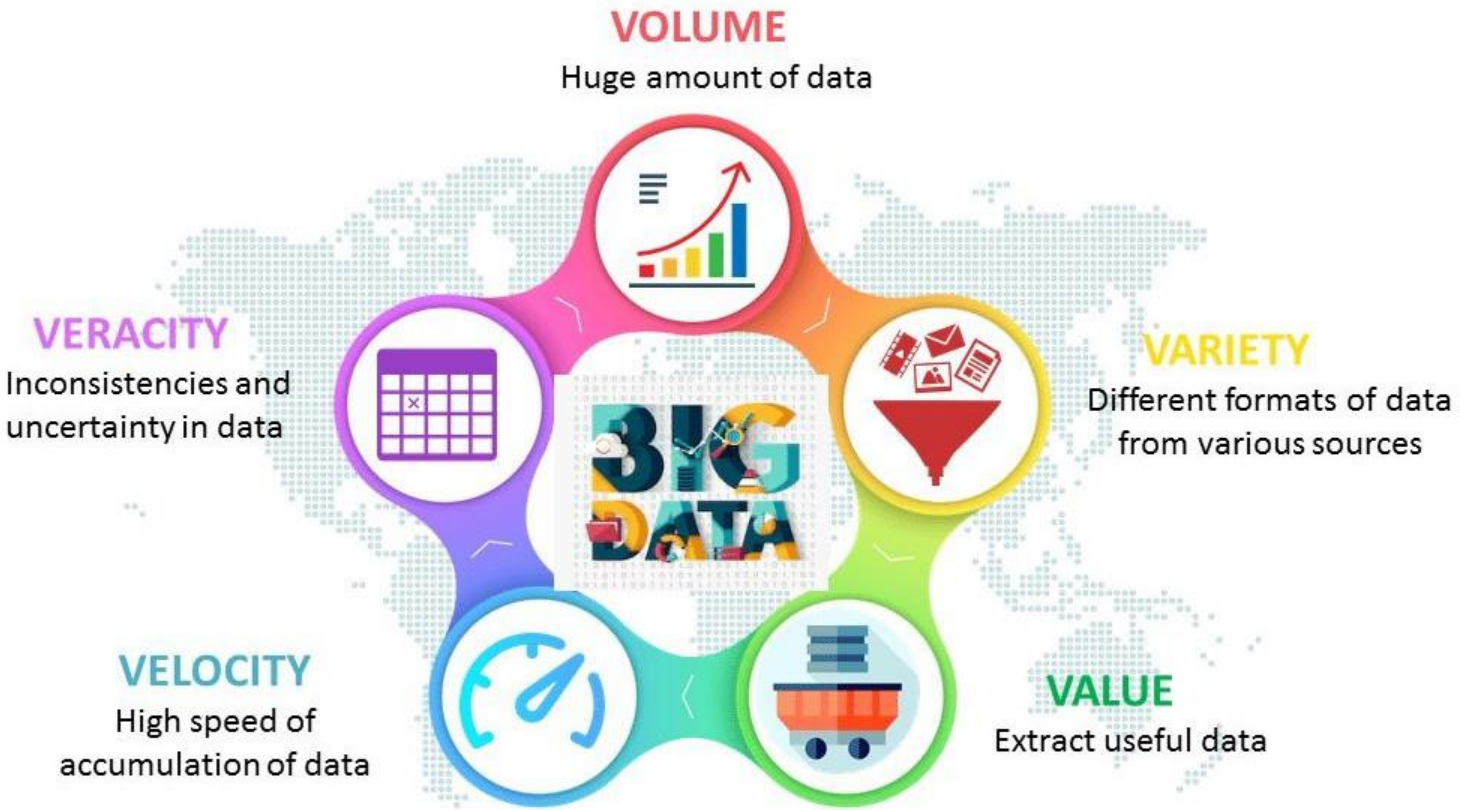



Image source :

# Big Data – The Solution

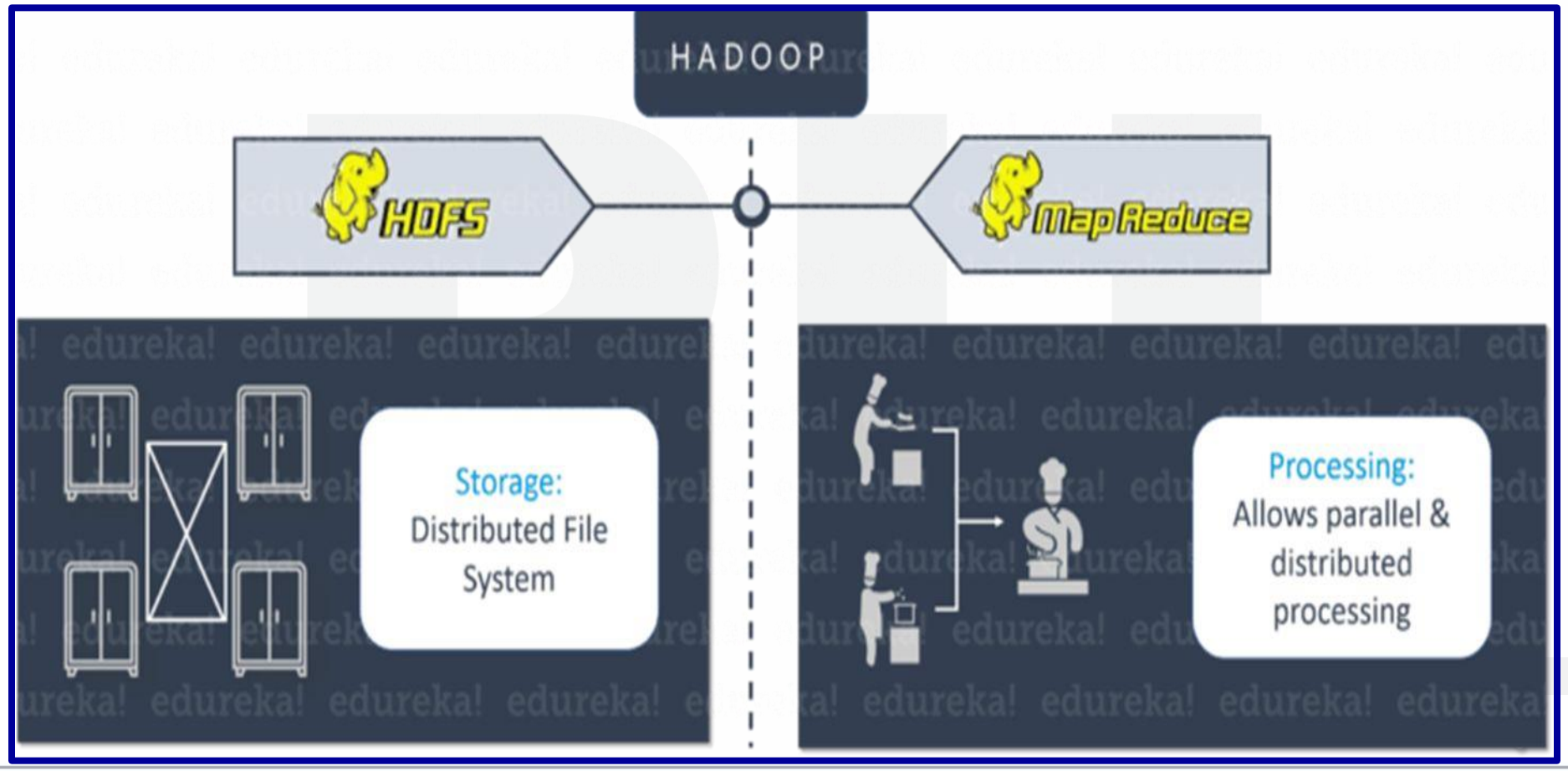
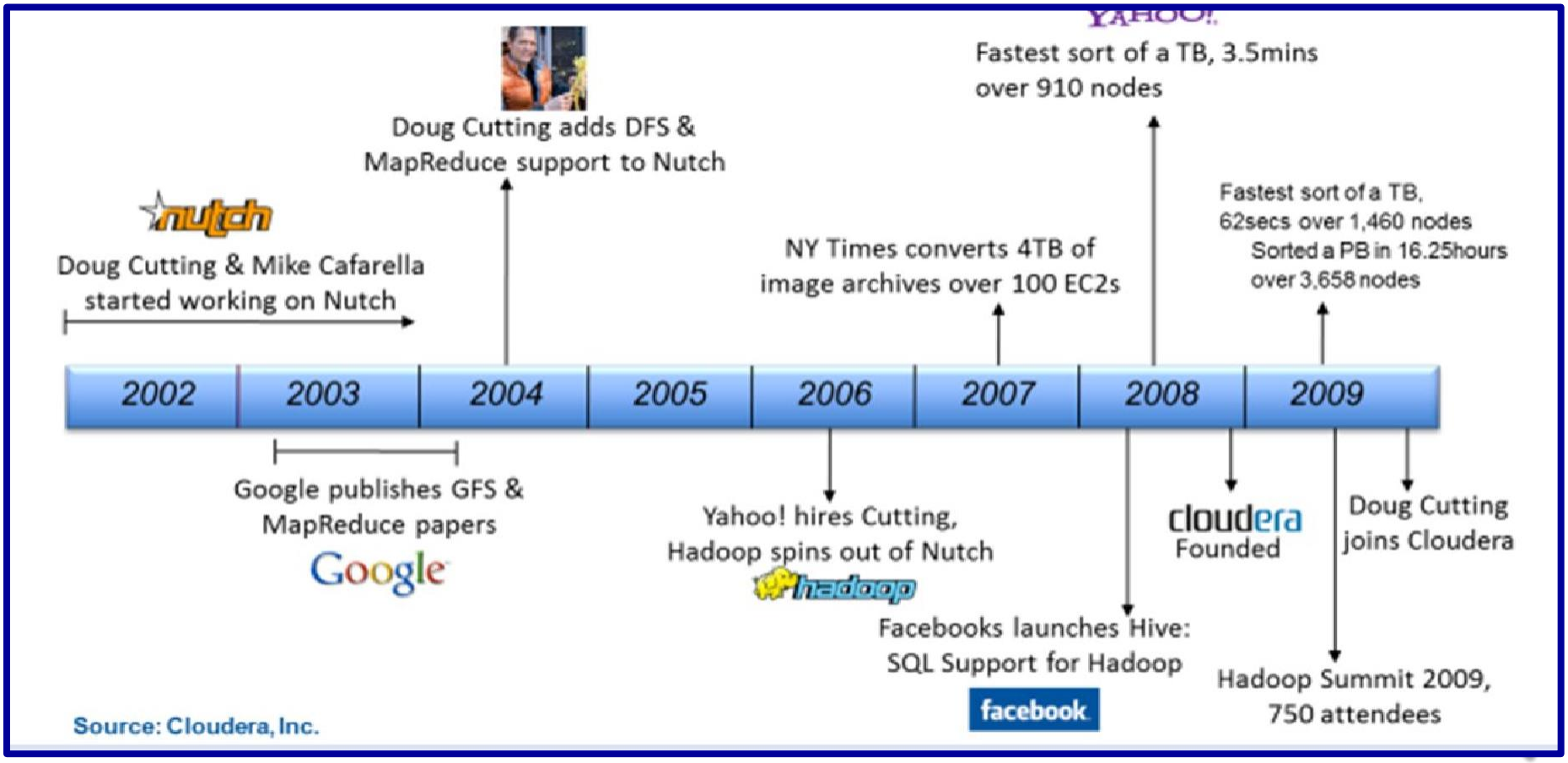



Image source :

# Big Data – The Solution


Image source:

# Hadoop Creation History




# What is Apache Hadoop?

- Open-source software framework designed for storage and processing of large-scale data on clusters of commodity hardware
- Created by Doug Cutting and Mike Carafella in 2005.
- Cutting named the program after his son's toy elephant.


# What is Apache Hadoop?

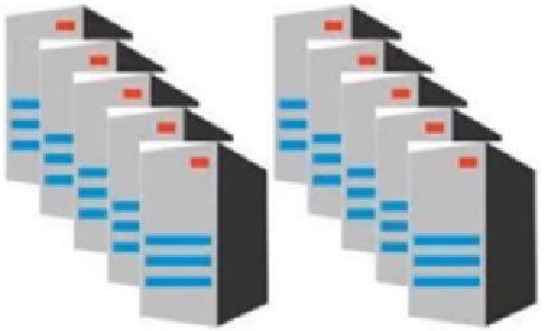
- Apache Hadoop is a framework that allows for the **distributed processing** of **large datasets** across **clusters** of **commodity computers** using a simple **programming model.**

# Distributed File System

## Read 1 TB Data

**1 Machine**

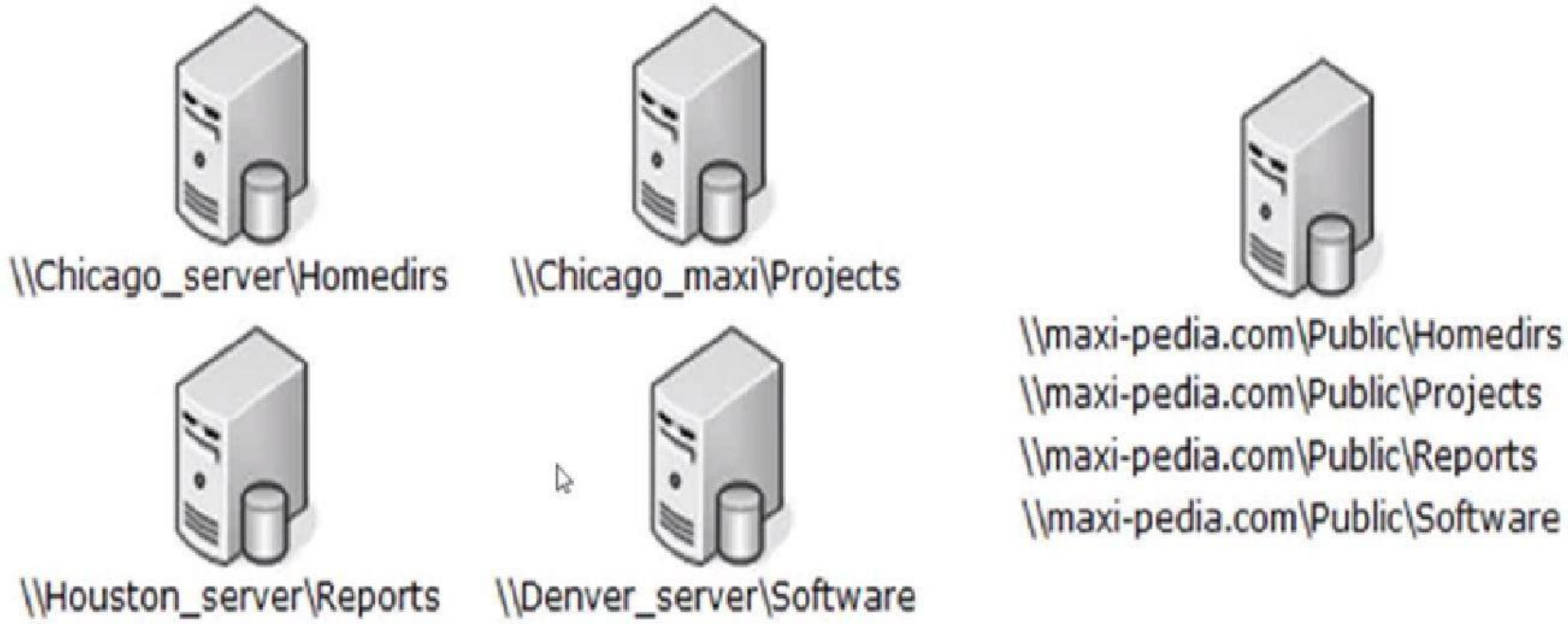
- 4 I/O Channels
- Each Channel – 100 MB/s



**10 Machines**

- 4 I/O Channels
- Each Channel – 100 MB/s

# Distributed File System




# Distributed File System

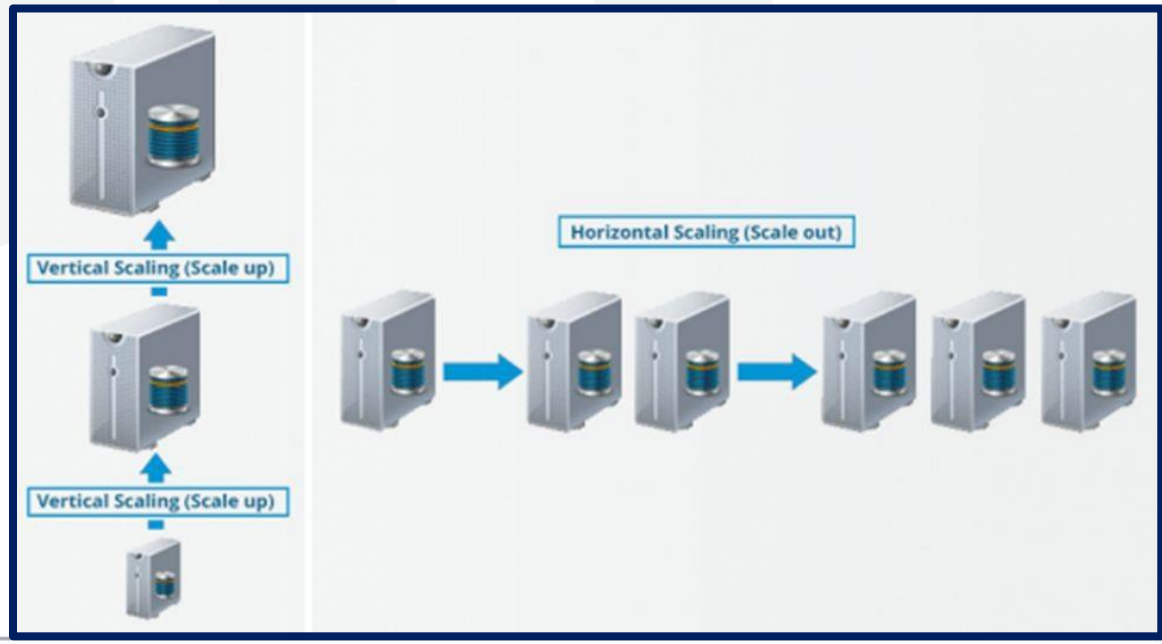
- A Distributed File System ( DFS ) is simply a classical model of a file system distributed across multiple machines. The purpose is to promote sharing of dispersed files.
- This is an area of active research interest today.
- **Clients should view a DFS the same way they would a centralized FS; the distribution is hidden at a lower level.**
- A DFS provides high throughput data access and fault tolerance

# Biggest Indian Use case of Hadoop

- Processing of every enrollment requires matching ten fingerprints, both irises, and demographics with every existing record in the database.
- With the Aadhaar database at 600 million, processing 1 million enrollments every day roughly translates to about 600 trillion matches every day.
- Do you know how many years do one trillion seconds make? **More than 31,000 years.**
- **Started using Hadoop**


# Scalability

- Scale up – Vertical growth
- Scale out - Horizontal Growth
  - Increase/decrease as per requirements
  - Fault tolerance




# Vertical Scaling

- **limit to which you can increase your hardware** capacity.

- In vertical scaling, you **stop your machine first**. Then you increase the RAM or CPU to make it a more robust hardware stack. After you have increased your hardware capacity, you **restart the machine**. **This down time when you are stopping your system becomes a challenge.**


# Horizontal Scaling

- **Add more nodes to existing cluster** instead of increasing the hardware capacity of individual machines
- **add more machines on the go** i.e. Without stopping the system. **NO downtime**.


# Main Characteristics of Hadoop

- High scalability and availability
- Use commodity (cheap!) hardware with little redundancy
- Fault-tolerance
- Move computation rather than data

# RDBMS V/S HADOOP

- **Supported Data Format / Schema**
  - **Hadoop : s**tructured, semi-structured and unstructured data.
  - **RDBMS** : only work on structured data

**SQL is based on the Entity-Relationship model** of its RDBMS, hence cannot work on unstructured data.

On the other hand**, Hadoop does not depend on any consistent relationship** and supports all data formats like XML, Text, and JSON, etc. So Hadoop can efficiently deal with big data.

# RDBMS V/S HADOOP

- **Volume of Data**
  - **RDBMS :** lower volume of data such **as few GB's** if RDBMS fulfills the requirement it is the best
  - **Hadoop :** Hadoop framework's processing power comes into realization when the file sizes are very **large**(Terabytes and Petabytes) and **streaming reads** and **processing** is the demand of the situation


# Is Hadoop faster than SQL ?

- Hadoop **doesn't support OLTP** (Real-time Data processing).
- Hadoop **supports large-scale Batch Processing** (OLAP) mainly used in data mining techniques.
- Using OLAP, **you can execute very complex queries** along with aggregations. Data processing time in Hadoop varies based on the volume of data and **sometimes can take several hours.**

- On the other hand, **RDBMS supports OLTP** (Real-time data processing) **which does not support Batch Processing.** Due to highly normalized data, SQL performs fast data processing.

- Hence, Is Hadoop faster than SQL? Probably the answer is '**"NO"**

# RDBMS V/S HADOOP

- **Working property**
  - **RDBMS : ACID (Consistency over Availability)**
  - **Hadoop : BASE (Availability over Consistency)**

In the case of Hadoop nothing like ACID is existent. But if we want to talk in the context of Distributed Databases there is a BASE property (Basically Available, Soft State, Eventually Consistent).

# BASE Properties

- **Basically Available:** The system is guaranteed to be available for querying by all users. (No isolation here.)
- ***Soft State:*** **The values stored in the system may change because of the eventual consistency model, as described in the next bullet.**
- **Eventually Consistent:** As data is added to the system, **the system's state is gradually replicated** across all nodes. For example, in Hadoop, when a file is written to the HDFS, the replicas of the data blocks are created in different data nodes after the original data blocks have been written. **For the short period before the blocks are replicated, the state of the file system isn't consistent.**

# RDBMS V/S HADOOP

- **Data Storing Technique**
  - **RDBMS :** data is stored in tables containing **relational structure characterized by defined rows and columns**
  - data is stored in **interrelated tables**
  - **Hadoop :** a basic data can begin in any format and shape.
  - In the long run, **it changes into a key-value pair**. Because once the data enters into Hadoop, it is replicated across multiple nodes in the Hadoop Distributed File System (HDFS).

# RDBMS V/S HADOOP

- **Performance – Throughput**

(It is the total volume of output data processed in a particular period)

  - SQL database fails to achieve a higher throughput as compared to the Hadoop Framework.

- **Performance – Latency**

  - **RDBMS:** RDBMS can give a **very quick response when the data size is ideal for its processing power**
  - **Hadoop:** Hadoop is **efficient for batch processing** of data. The results are only available after a large amount of data has been processed. Therefore, **Hadoop is not the ideal platform to use when immediate results are expected**.

# RDBMS V/S HADOOP

- **Scalability**
- **RDBMS:** With RDBMS you can add more hardware like memory, CPU in the cluster to scale up the machine. It is known as **vertical scalability or scaling.**
- Hadoop: you can add more machines in the existing cluster. This is known as **horizontal scalability or scaling out**. Moreover, in Hadoop, there is no single point of failure. Hence, it is fault tolerant.
- Hadoop vs SQL database – of course, **Hadoop is more scalable.**

Parul® University

DIGITAL LEARNING CONTENT

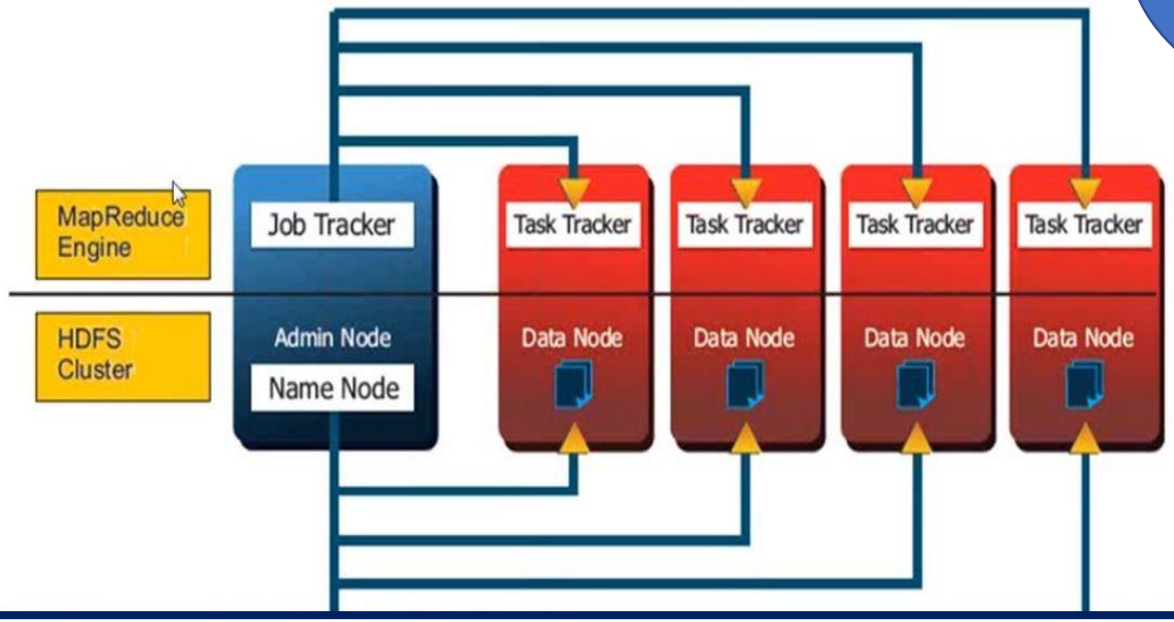
# RDBMS V/S HADOOP

| Feature | RDBMS | Hadoop |
|---|---|---|
| Data Variety | Mainly for Structured data. | Used for Structured, Semi-Structured and Unstructured data |
| Data Storage | Average size data (GBS) | Use for large data set (Tbs and Pbs) |
| Querying | SQL Language | HQL (Hive Query Language) |
| Schema | Required on write (static schema) | Required on read (dynamic schema) |
| Speed | Reads are fast | Both reads and writes are fast |
| Cost | License | Free |
| Use Case | OLTP (Online transaction processing) | Analytics (Audio, video, logs etc), Data Discovery |
| Data Objects | Works on Relational Tables | Works on Key/Value Pair |
| Throughput | Low | High |
| Scalability | Vertical | Horizontal |
| Hardware Profile | High-End Servers | Commodity/Utility Hardware |

# HADOOP Main Components (Hadoop 1.x)

- ✓ HDFS – Hadoop Distributed File System (storage)
- ✓ MapReduce (processing)

Minimal Configuration




Parul® University

DIGITAL LEARNING CONTENT

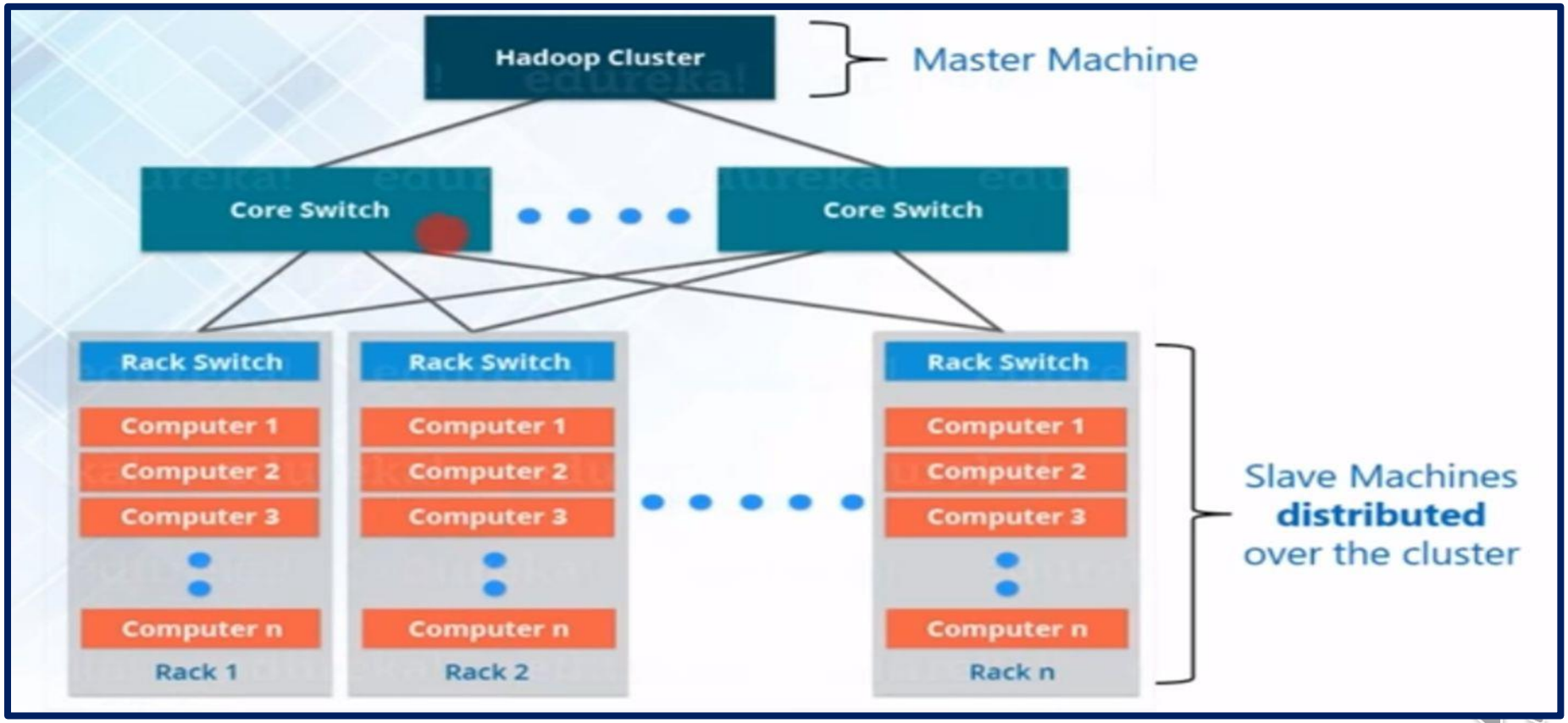
# Hadoop Components (Hadoop 2.x)

| | |
|---|---|
| Hadoop Common | • Contains Libraries and other modules |
| HDFS | • Hadoop Distributed File System |
| Hadoop YARN | • Yet Another Resource Negotiator |
| Hadoop MapReduce | • A programming model for large scale data processing |

# HDFS

- Hadoop Distributed File Sysytem
  - Highly fault tolerant
  - High throughput
  - Suitable for applications with large data sets
  - Streaming access to file system
  - Can be built out of commodity hardware
- HDFS uses a master/slave architecture in which one device (master) termed as NameNode controls one or more other devices (slaves) termed as DataNode

# Hadoop Cluster Architecture




# How data is stored in HDFS ?

- Data gets stored as blocks in HDFS.
- Default size : 128 MB in Apache Hadoop 2.x

  64MB in Apache Hadoop 1.x
- Is 128 MB is fixed ?
  - No.
  - We can configure the block size as per our requirement by changing the dfs.block.size property in hdfs-site.xml

# How data is stored in HDFS ?

Why block size is made fix as 128 MB?

What if larger than 128 MB ?

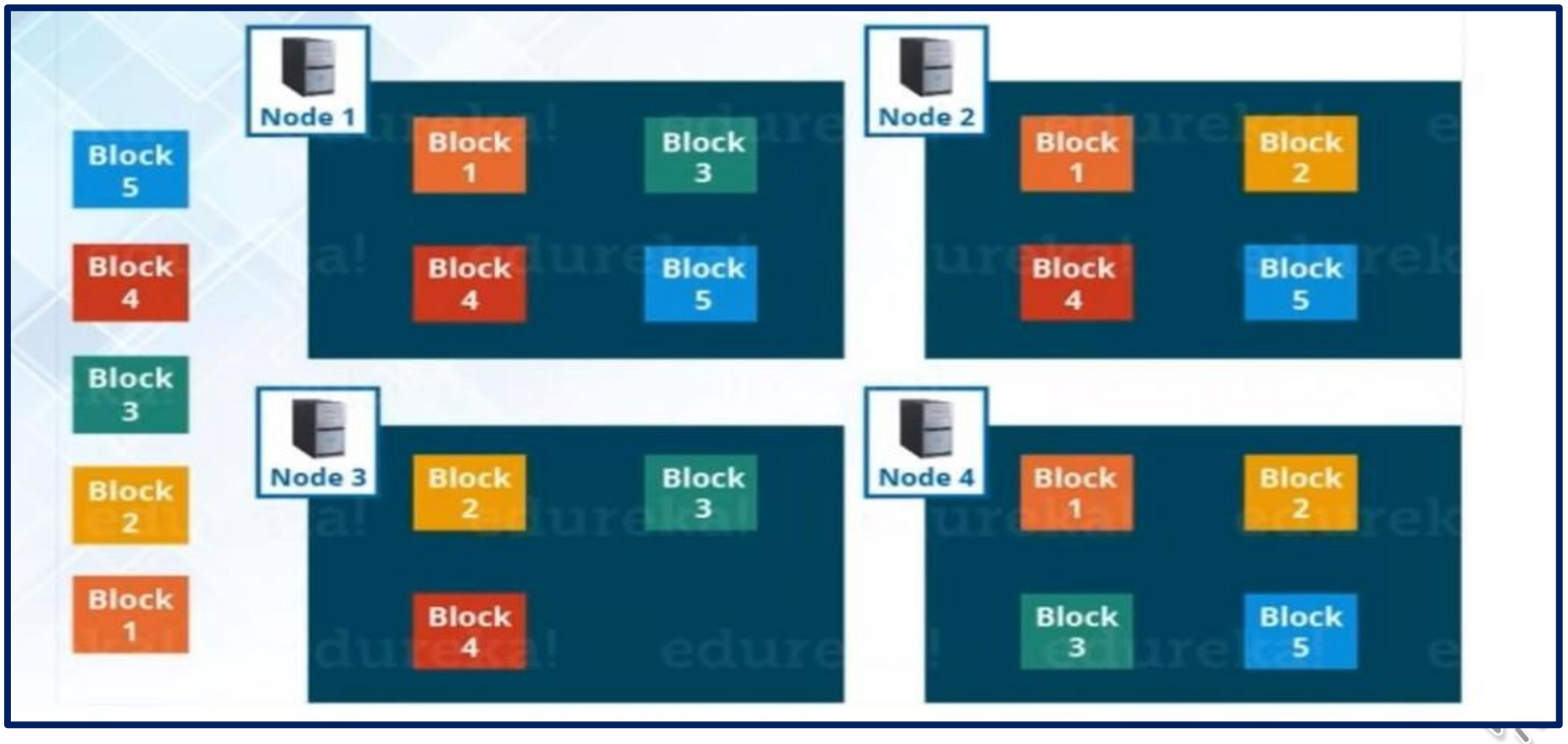
What if smaller than 128 MB?


# Replication of blocks in Hadoop

- If there's single copy of block stored on commodity hardware, in failure of machine, will lead to lose of the block stored on that machine as well.

PU

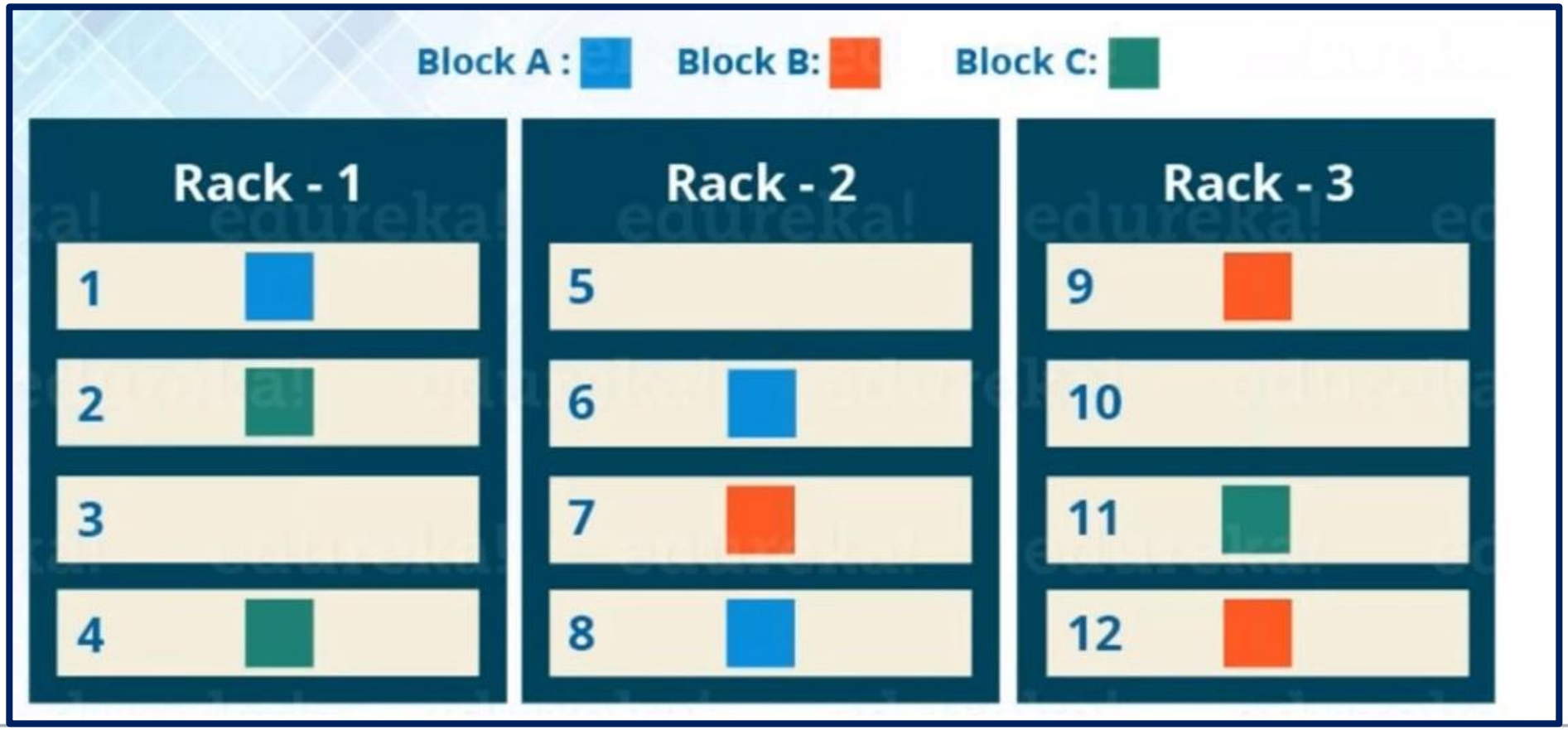
# Replication of blocks in Hadoop




# How the blocks are replicated ?

- Rack awareness policies:
  - Not more than one replica be placed on one node.
  - Not more than two replicas are placed on the same rack.
  - Also, the number of racks used for block replication should always be smaller than the number of replicas.

# How the blocks are replicated ?




# Example

- How many blocks will be created for Hadoop 2.x for the file size of 514 MB ? Specify the block size of each block.

# Example – Solution

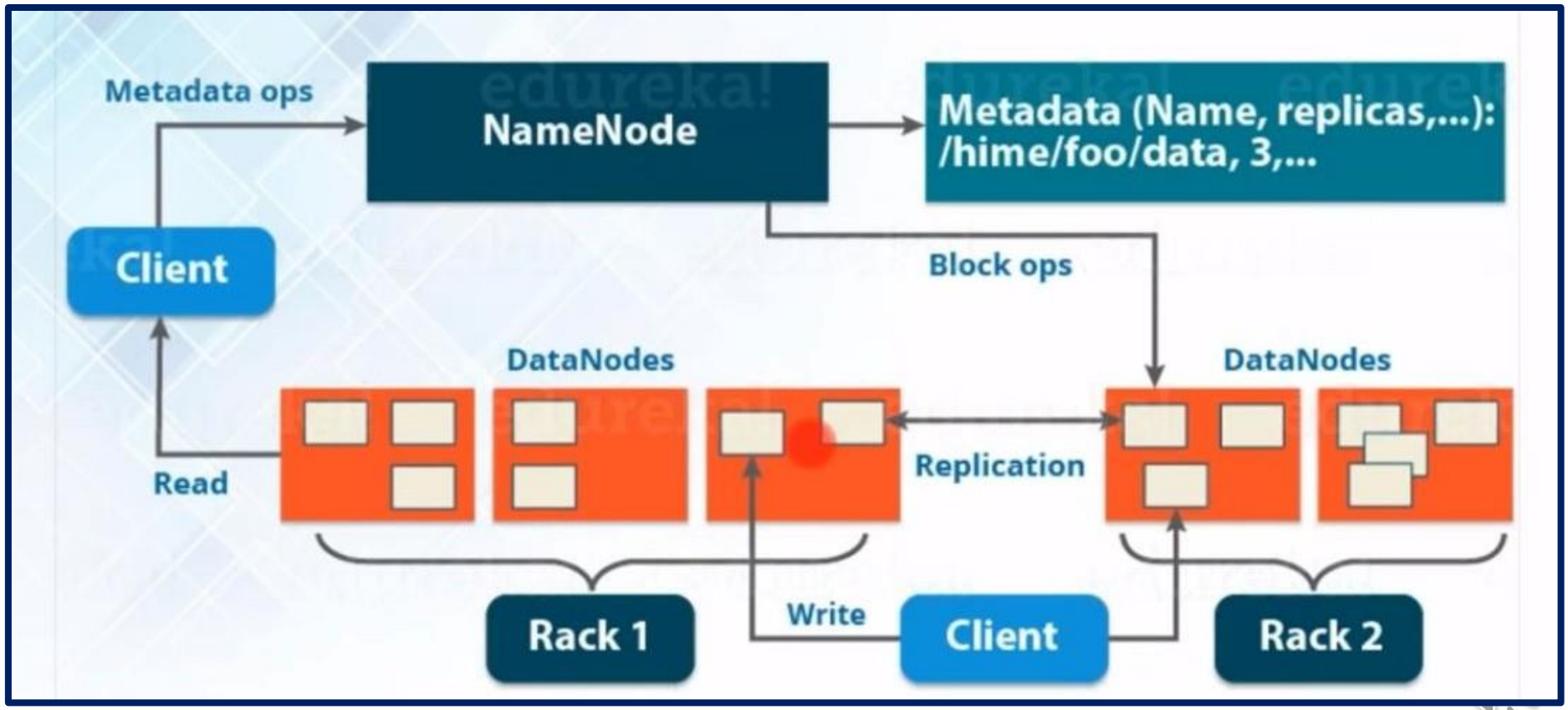
- Default block size in Hadoop 2.x is 128 MB.
- So, a file of size 514 MB will be divided into 5 blocks ( 514 MB/128 MB) where the first four blocks will be of 128 MB and the last block will be of 2 MB only.
- Since, we are using the default replication factor i.e. 3, each block will be replicated thrice.
- Therefore, **we will have 15 blocks in total** where **12 blocks will be of size 128 MB each and 3 blocks of size 2 MB each.**


# Example

- How many blocks will be created for Hadoop 2.x for the file size of 1030 MB ? Specify the block size of each block.

PU

# HDFS Architecture




# HDFS Components

- The main components of **HDFS** are: **NameNode** and **DataNode**.

# NameNode

- It is the **master daemon that maintains and manages** the DataNodes (slave nodes)
- It **records the metadata** of all the blocks stored in the cluster, e.g. location of blocks stored, size of the files, etc.
- It **records each and every change** that takes place to the file system metadata
- If a file is deleted in HDFS, the NameNode will immediately record this in the EditLog
- It **regularly receives a Heartbeat and a block report from all the DataNodes** in the cluster to ensure that the DataNodes are live
- It keeps a record of all the blocks in the HDFS and DataNode in which they are stored

# NameNode

- Metadata in Namenode
  - Stored in local system as two files

    - **the filesystem image (FsImage)**

    (It contains the complete state of the file system namespace since the start of the NameNode.)

    - **Edit log**

    (It contains all the recent modifications made to the file system with respect to the most recent FsImage.)

- Without the NameNode, the filesystem can't be used.
- **So, NameNode is the single point of failure in HDFS.**


# NameNode

- If the machine running the NameNode crashes, all files on the filesystem would be lost as there would be no way of knowing how to reconstruct the files from the blocks on DataNodes.

- To cope up with failures, make the NameNode robust.

# Making NameNode Robust

- Two ways

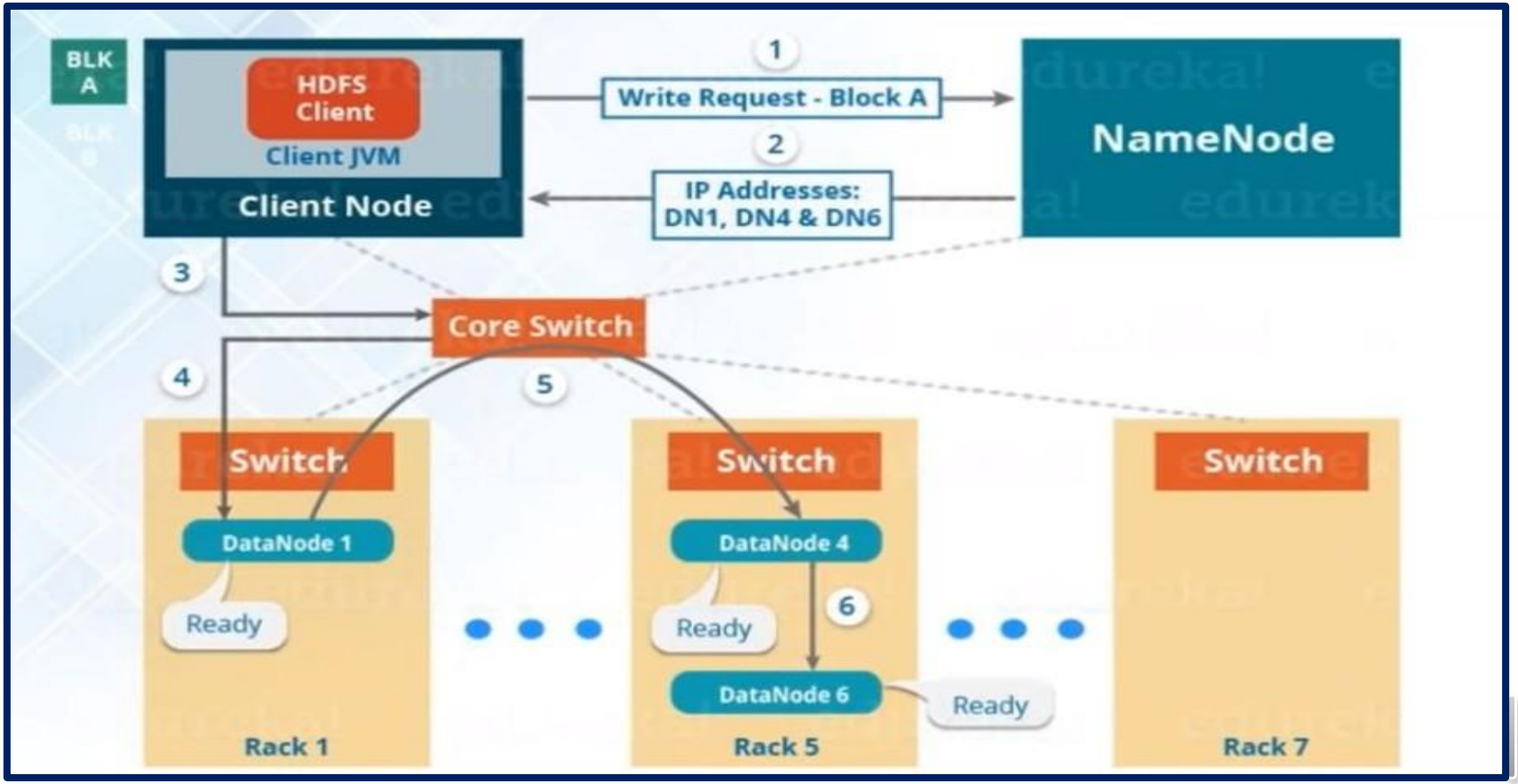
- Hadoop can be set in a way that NameNode creates its state for various file systems. The normal setup choice is to **write to the local disk and to a remote NFS mount**.

- Another way is to run a **Secondary NameNode**
  - It periodically reads the filesystem, changes the log and apply them into the fsimage file.
  - When the NameNode is down, **secondary NameNode will be online but this node will only have read permissions** to fsimage and editlog file.


# DataNode

- It is the slave daemon which **run on each slave machine**
- The **actual data is stored** on DataNodes
- It is **responsible for serving read and write** requests from the clients
- It is also responsible for **creating blocks, deleting blocks and replicating the same** based on the decisions taken by the NameNode
- It **sends heartbeats to the NameNode periodically** to report the overall health of HDFS, by default, this frequency is set to 3 seconds

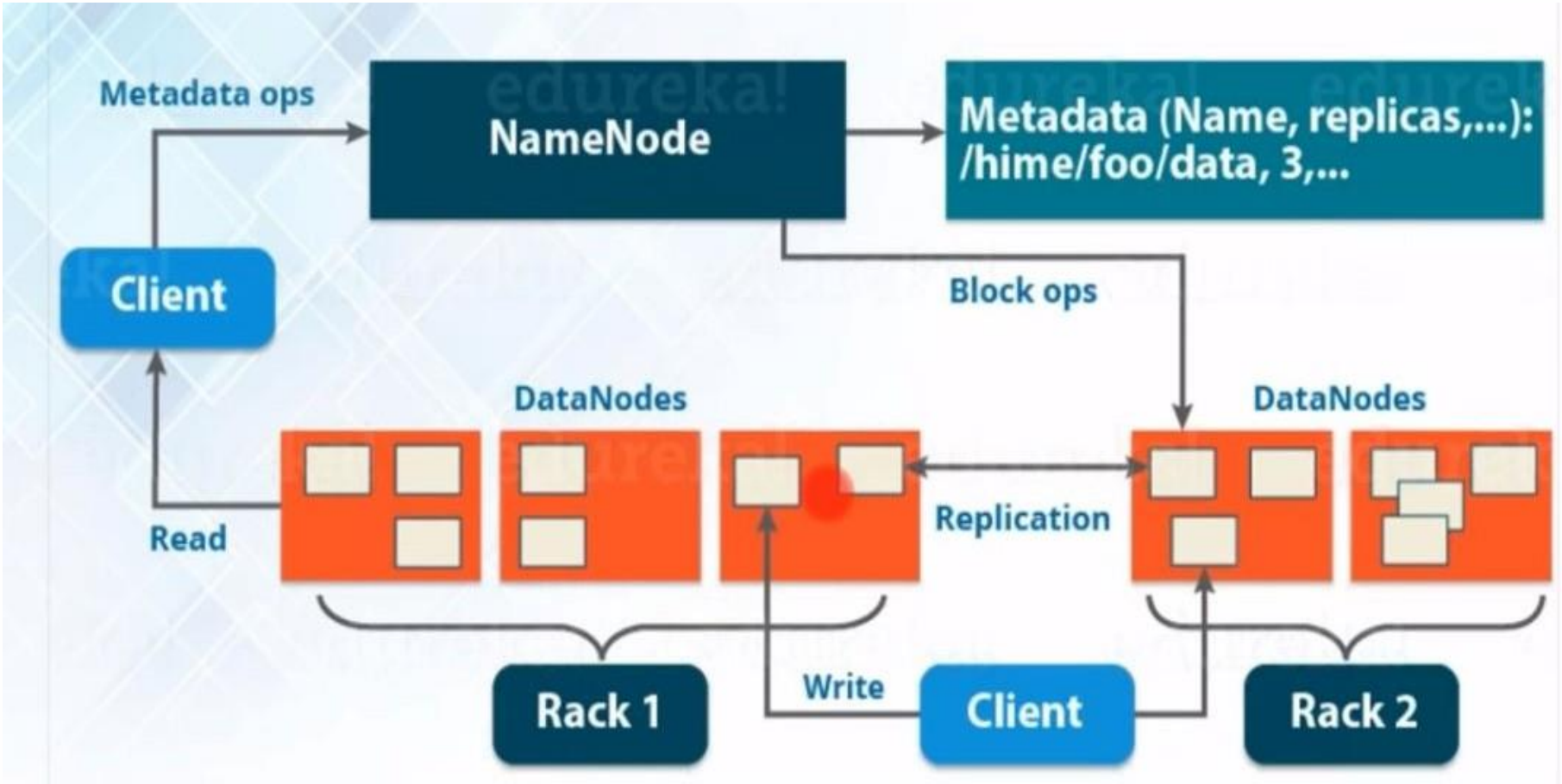
# WRITE Mechanism




# Point to Consider

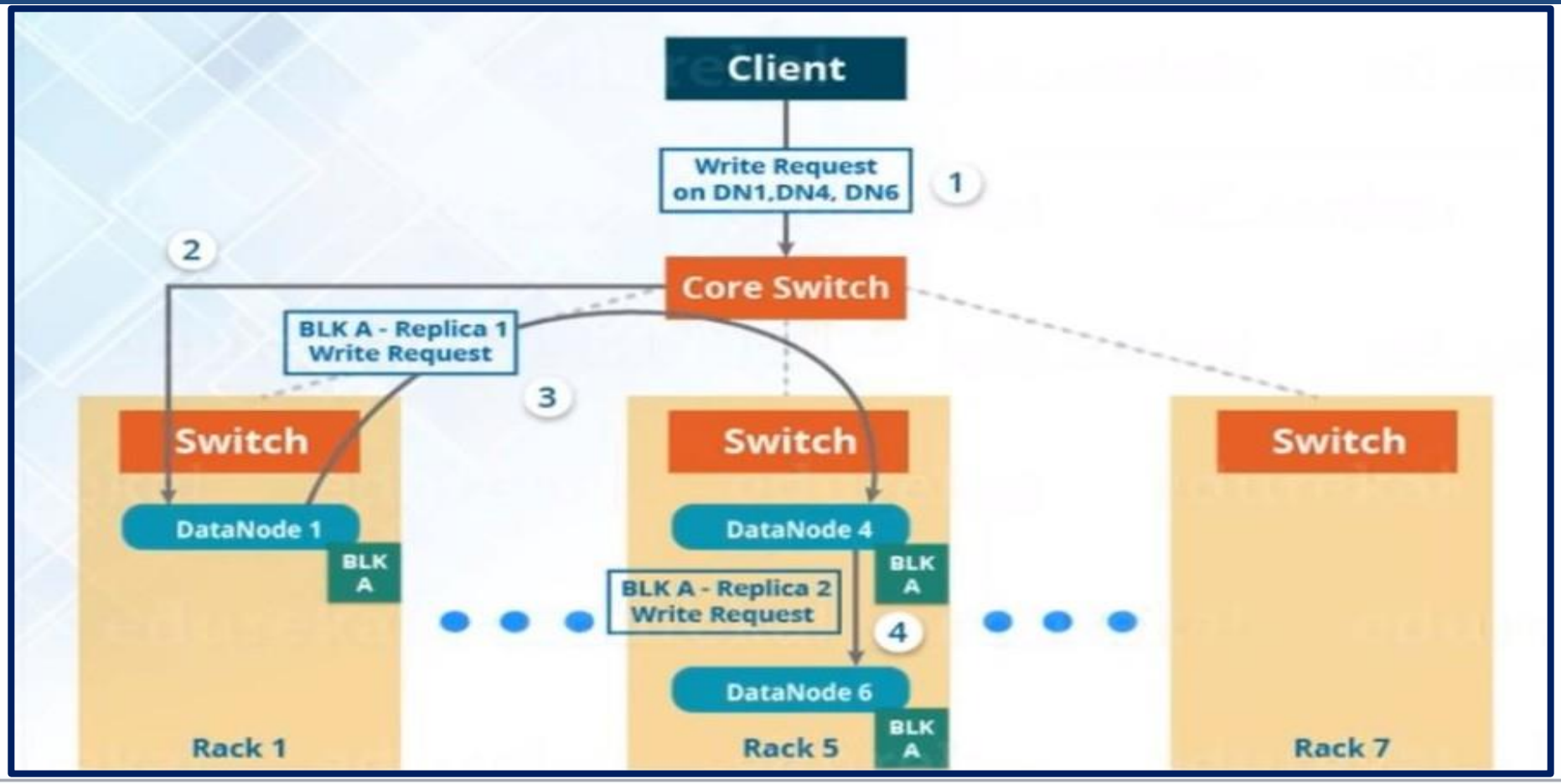
Why the replicas are created in the same rack only ?

# Replication

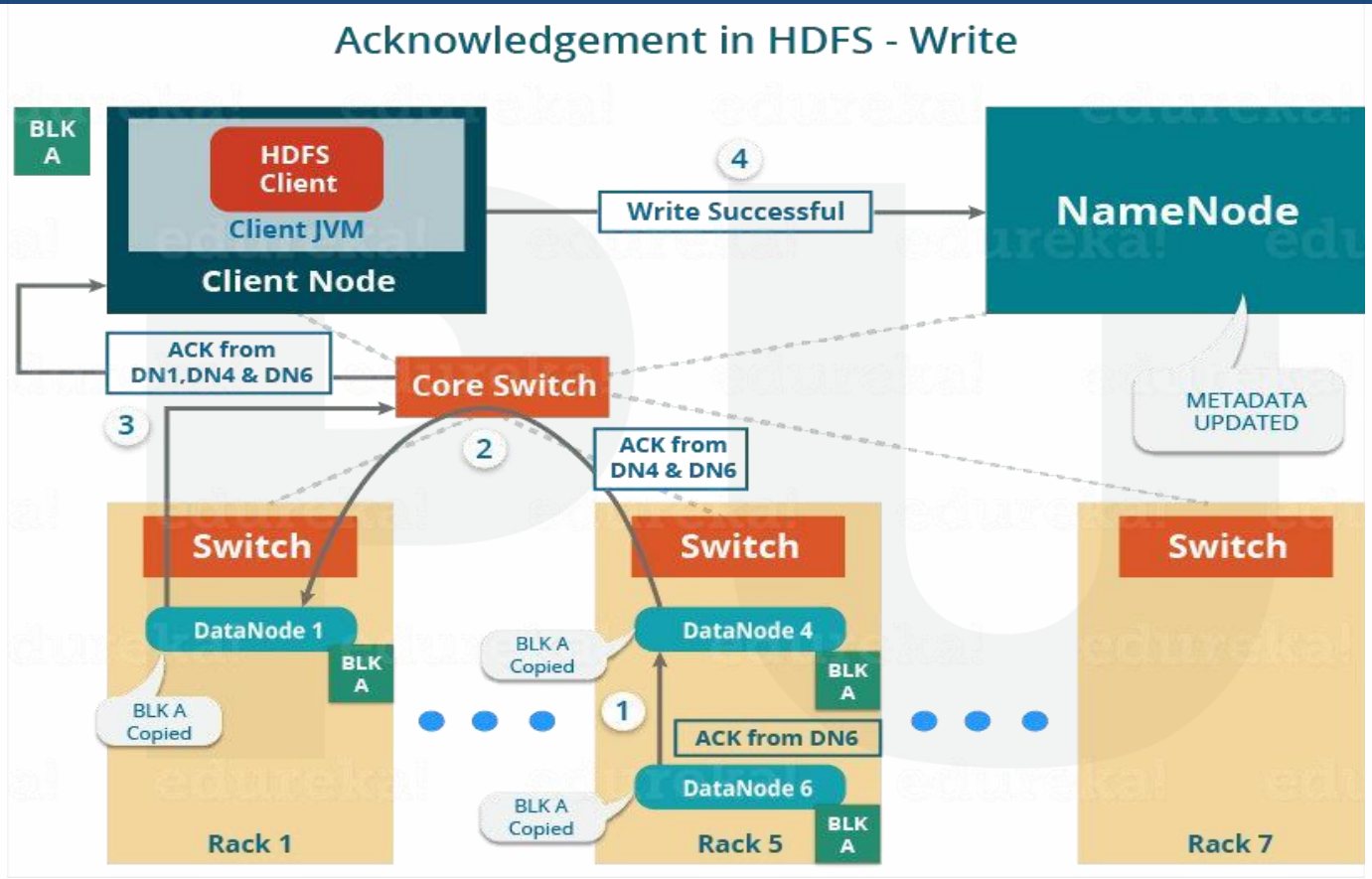


# Write Mechanism – Setting up HDFS-write pipeline

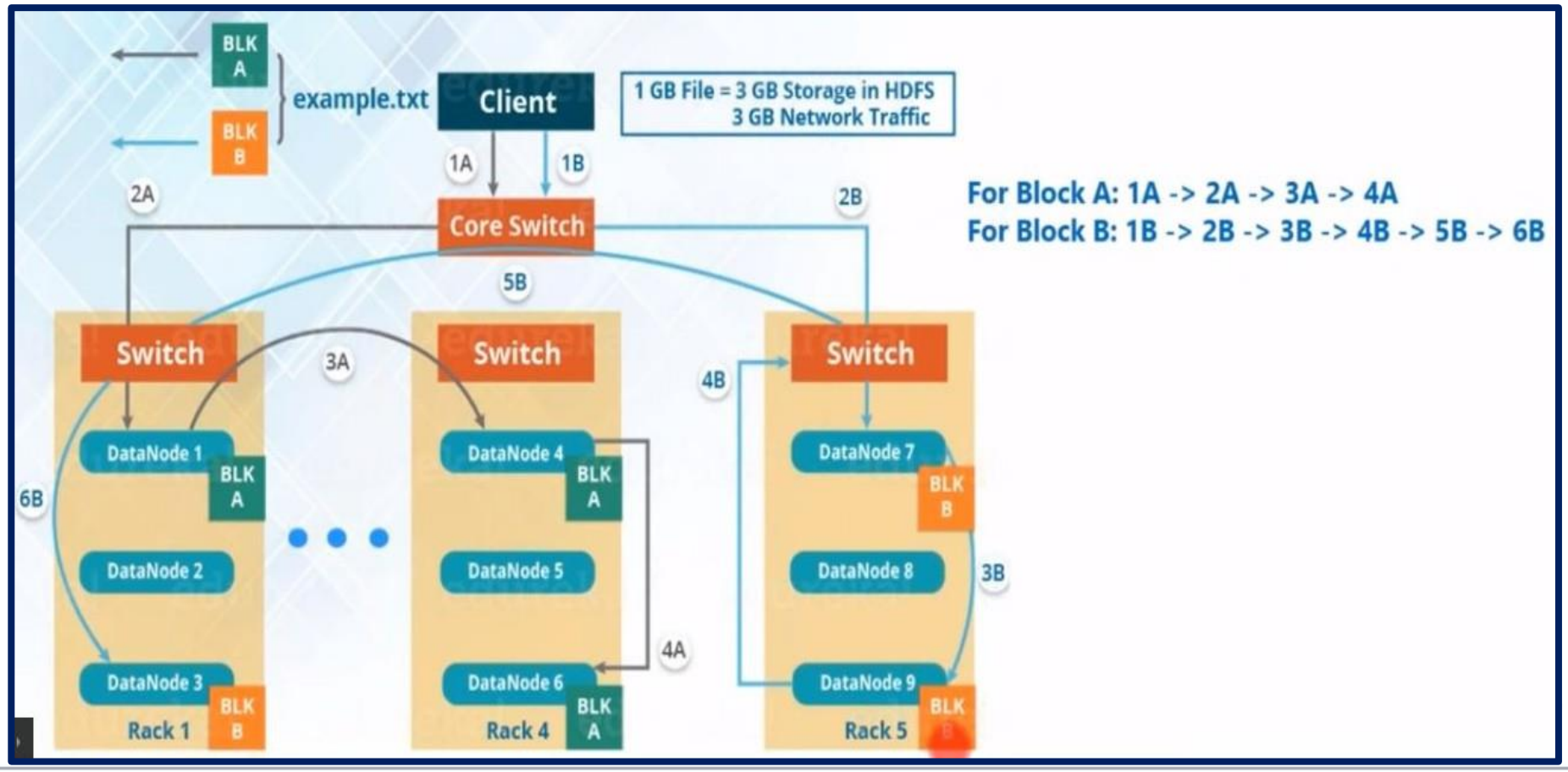
# Write Pipeline - Write operation (Sequential)



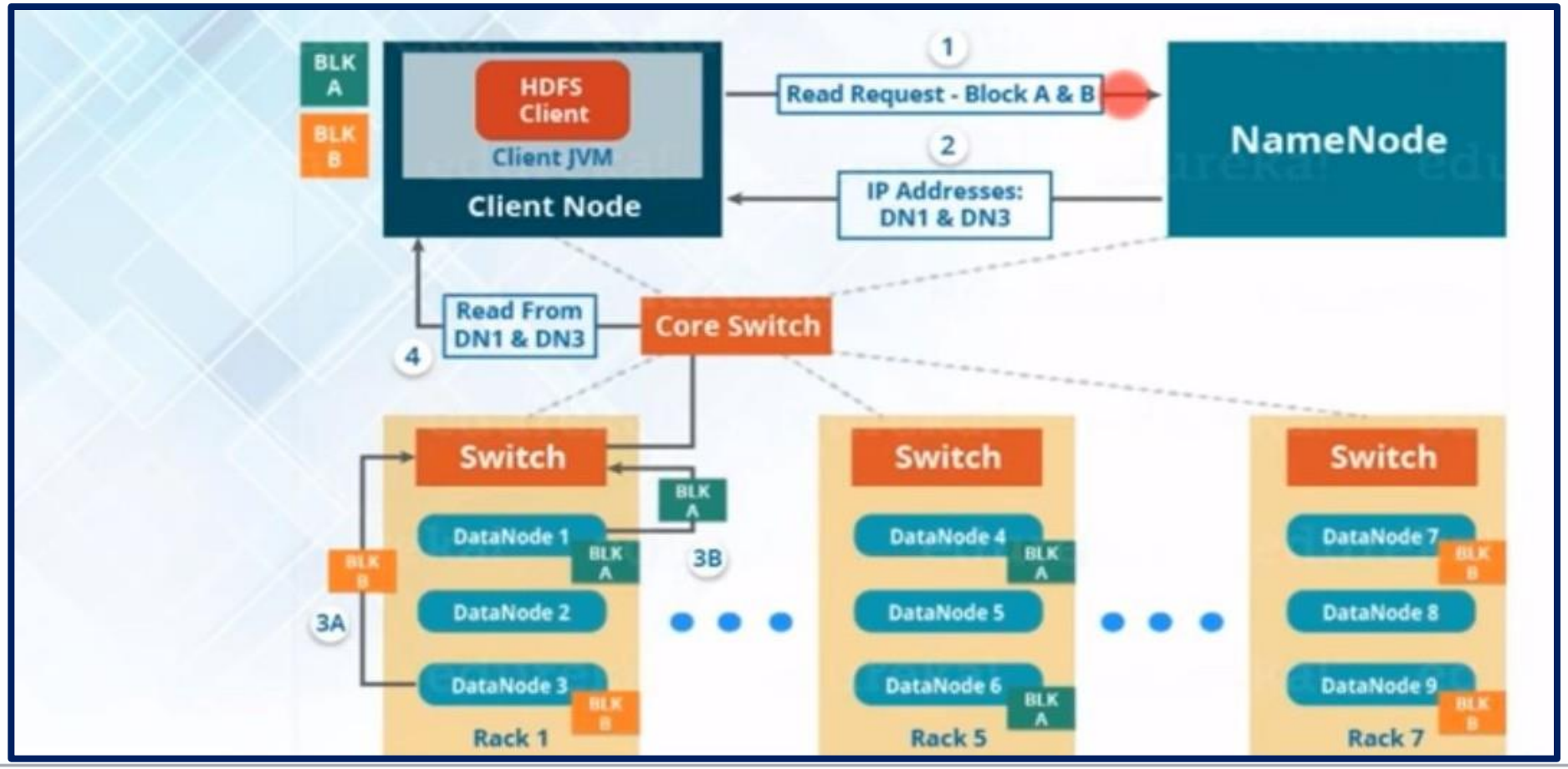
# Shut Down Write Pipeline with Write operation Acknowledgement



# Multiple Block Write Mechanism



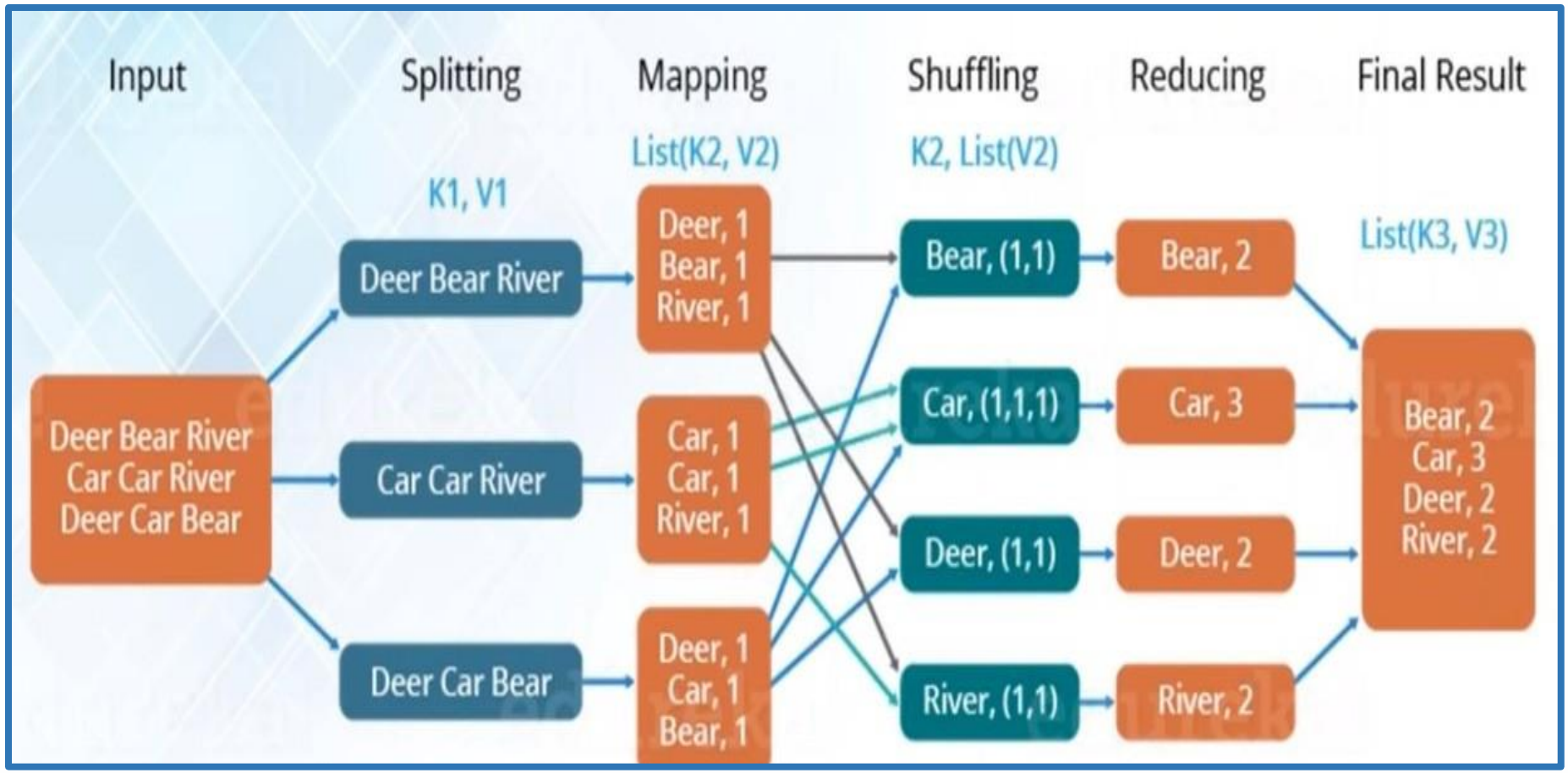
# Read Mechanism



# MapReduce Overview

- MapReduce is based on parallel programming framework to process large amount of data dispersed across different systems.
- We can process the Big data stored on HDFS using Map-Reduce program in Hadoop architecture.
- The process is initiated when a user request is received to execute the MapReduce program and terminated once the results are written back to the HDFS.
- Two main operations : Map and Reduce

# Overall MapReduce Process




# Drawbacks of MapReduce Approach

- Since MapReduce is suitable only for batch processing jobs, implementing interactive jobs and models becomes impossible.
- Problems that cannot be trivially partitionable or recombinable becomes a candid limitation of MapReduce problem solving. For instance, Travelling Salesman problem.
- Tasks that has a dependency on each other cannot be parallelized, which is not possible through MapReduce.

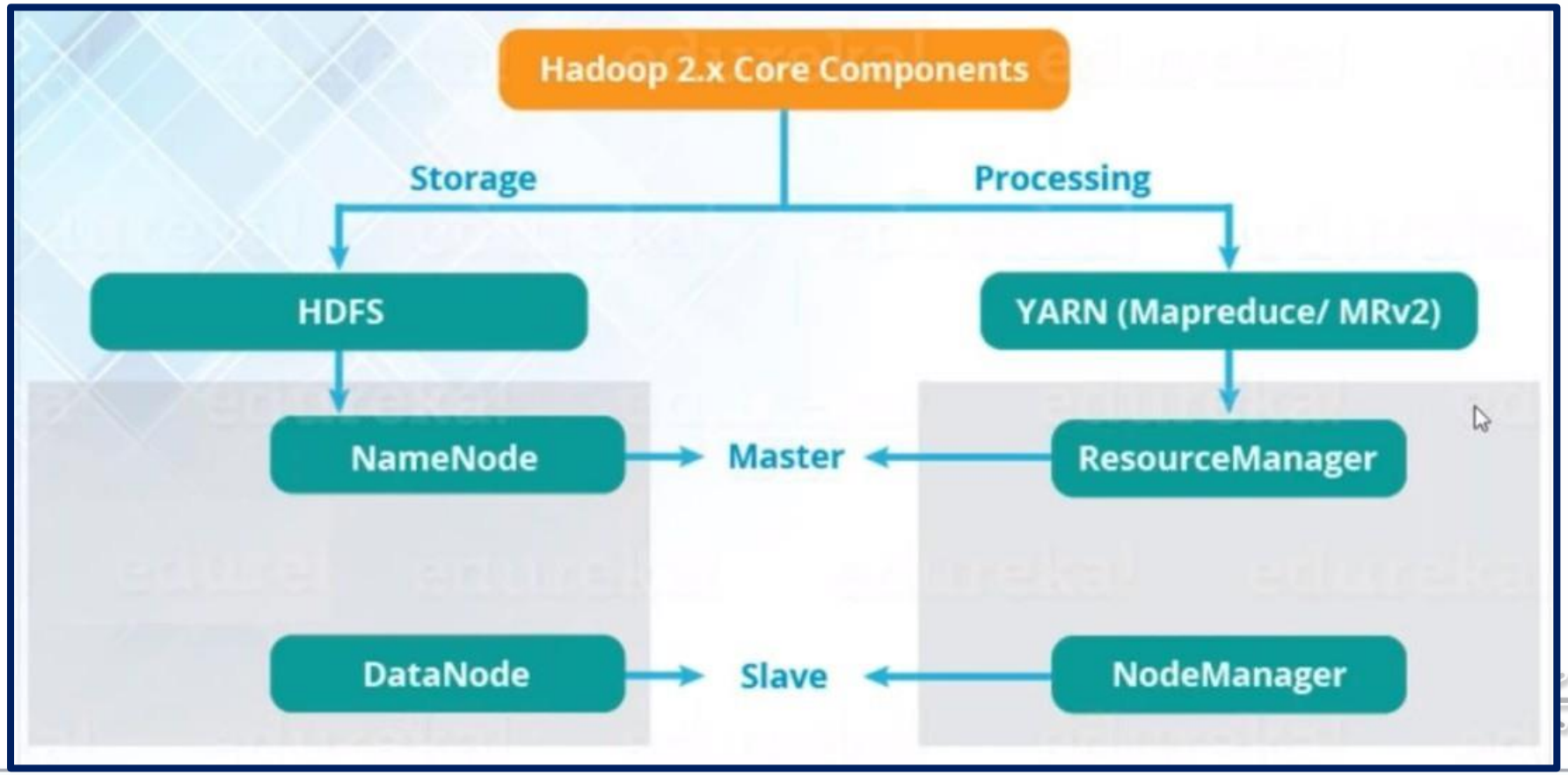
Parul® University

DIGITAL LEARNING CONTENT

# Hadoop Components (Hadoop 2.x)

| | |
|---|---|
| Hadoop Common | • Contains Libraries and other modules |
| HDFS | • Hadoop Distributed File System |
| Hadoop YARN | • Yet Another Resource Negotiator |
| Hadoop MapReduce | • A programming model for large scale data processing |

# Hadoop 2.x Components



# Hadoop YARN Architecture

- Apache YARN framework contains
  - a Resource Manager (master daemon),
  - Node Manager (slave daemon), and
  - an Application Master.

# Hadoop YARN Architecture – Resource Manager

- Resource Manager is the master daemon of YARN
- It is basically used for job scheduling

- Resource Manager has two components:
  - **Scheduler:** Scheduler's task is to distribute resources to the running applications
  - **Application Manager:** Application Manager manages applications running in the cluster. It performs tracking and monitoring of applications

# Hadoop YARN Architecture – Node Manager

- Node Manager is the slave daemon of YARN. It has the following responsibilities:
  - Node Manager has to monitor the container's resource usage, along with reporting it to the Resource Manager.
  - The health of the node on which YARN is running is tracked by the Node Manager.
  - It keeps the data in the Resource Manager updated
  - Node Manager can also destroy or kill the container if it gets an order from the Resource Manager to do so.

# Hadoop YARN Architecture – Application Manager

- Every job submitted to the framework is an application, and every application has a specific Application Master associated with it.
- Application Master performs the following tasks:
  - It coordinates the execution of the application in the cluster, along with managing the faults.
  - It negotiates resources from the Resource Manager.
  - At regular intervals, heartbeats are sent to the Resource Manager for checking its health.

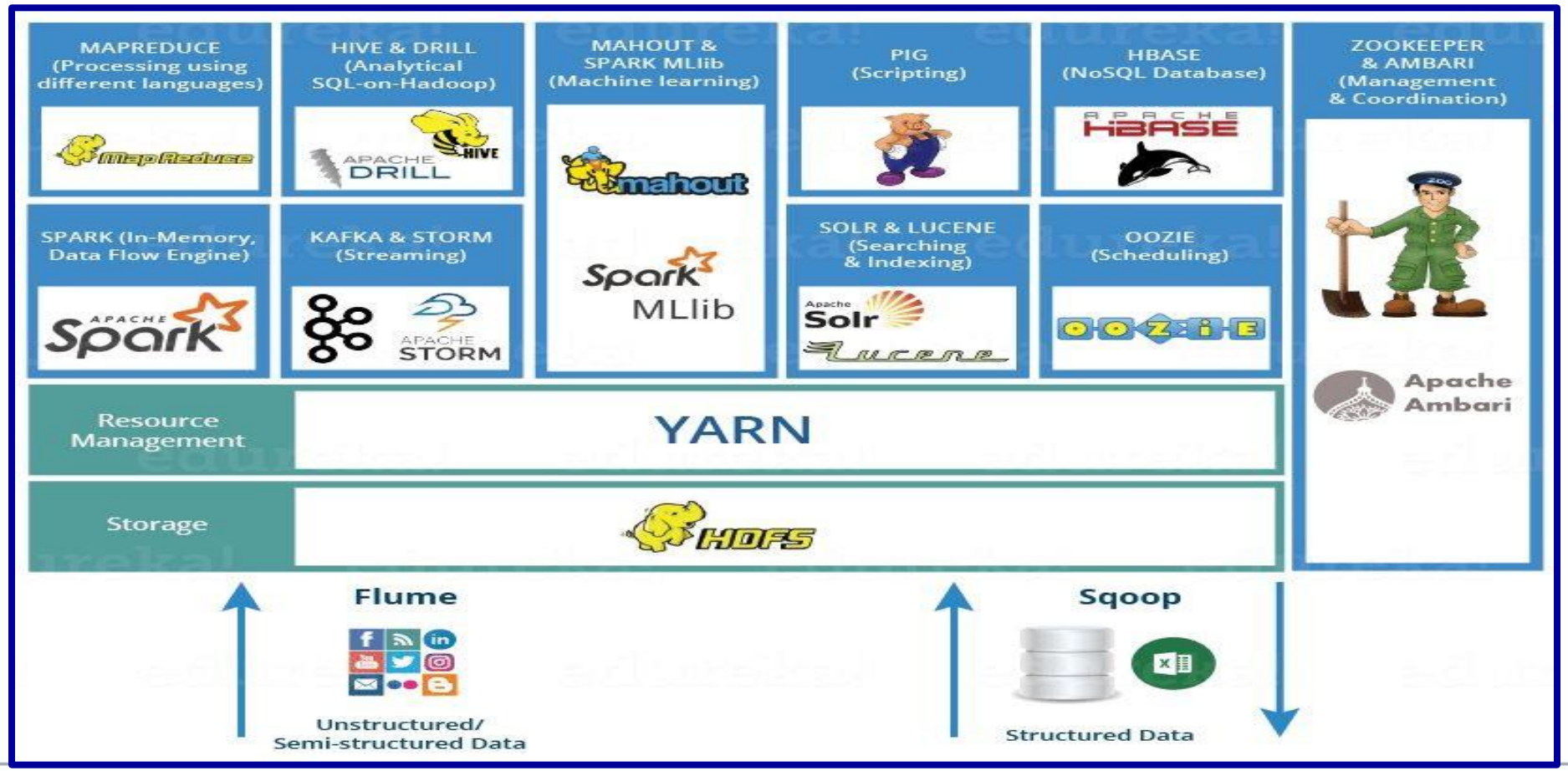
# Hadoop YARN vs Hadoop MapReduce

| Criteria | MapReduce | YARN |
|---|---|---|
| Type of processing | Batch processing with a single engine | Real-time, batch, and interactive processing with multiple engines |
| Cluster resource optimization | Average due to fixed Map and Reduce slots | Excellent due to central resource management |
| Suitable for | Only MapReduce applications | MapReduce and non-MapReduce applications |


# Hadoop Ecosystem

- A Hadoop ecosystem can be defined as a comprehensive **collection of tools and technologies** that can be effectively implemented and deployed to provide Big Data solutions in a **cost-effective manner**.

# Hadoop Ecosystem



# Hadoop 2.x vs Hadoop 3.x

- Fault Tolerance
  - **Hadoop 2.x:** Replication technique provides for fault tolerance in Hadoop 2.x. In the event of loss of any file block, Hadoop recovers it from the existing replicated blocks.
  - **Hadoop 3.x:** This version of Hadoop provides the technique of Erasure Coding for Fault Tolerance. Under erasure coding the blocks are not replicated in fact HDFS calculates the parity blocks for all file blocks. Now whenever the file blocks get corrupted, the Hadoop framework recreates using the remaining blocks along with the parity blocks.

# Hadoop 2.x vs Hadoop 3.x

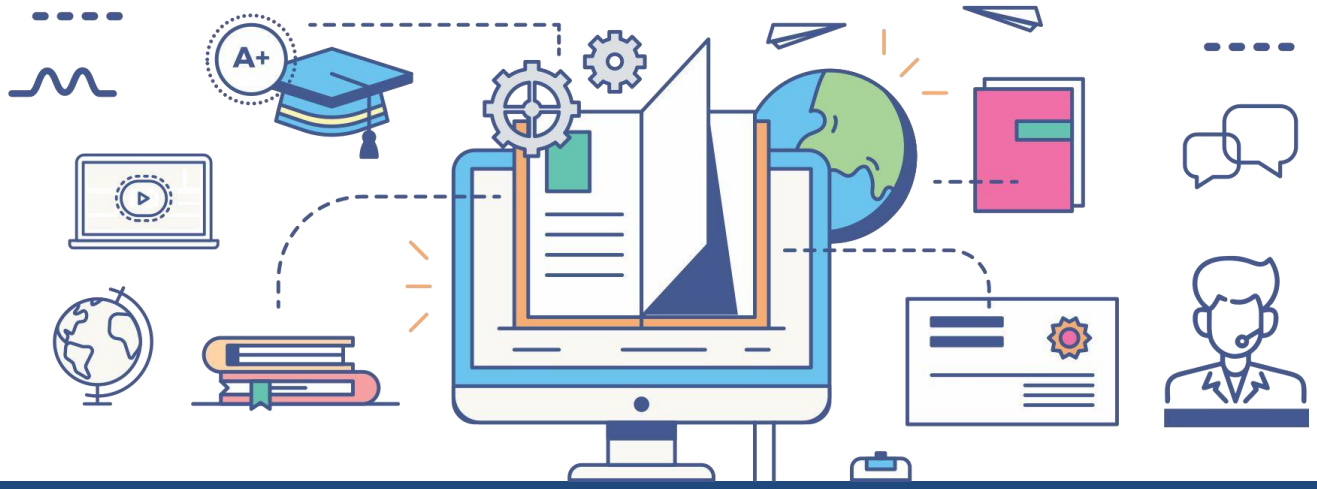
- Storage Overhead
  - **Hadoop 2.x:** Suppose a file “A” divides into 6 blocks in HDFS. With a replication factor of 3, we would be having 18 blocks for the file “A” stored in the system
  - **Hadoop 3.x:** As Hadoop 3.x adopts Erasure Coding for fault tolerance, it minimizes the storage overhead of the data. Again take the example of a file with 6 blocks. Erasure Coding creates 3 more parity blocks.

# Hadoop 2.x vs Hadoop 3.x

- Default Port for Multiple services
  - **Hadoop 2.x:** Other services in Linux use the default ports (that are default ports for multiple Hadoop services) as well, hence, they conflict with Hadoop services. Therefore, Hadoop services would fail to bind at startup.
  - **Hadoop 3.x:** To mitigate the above drawback this new version moves the default port out of Linux ephemeral port range. .


# Who Uses HADOOP?

# DIGITAL LEARNING CONTENT




Parul® University

www.paruluniversity.ac.in