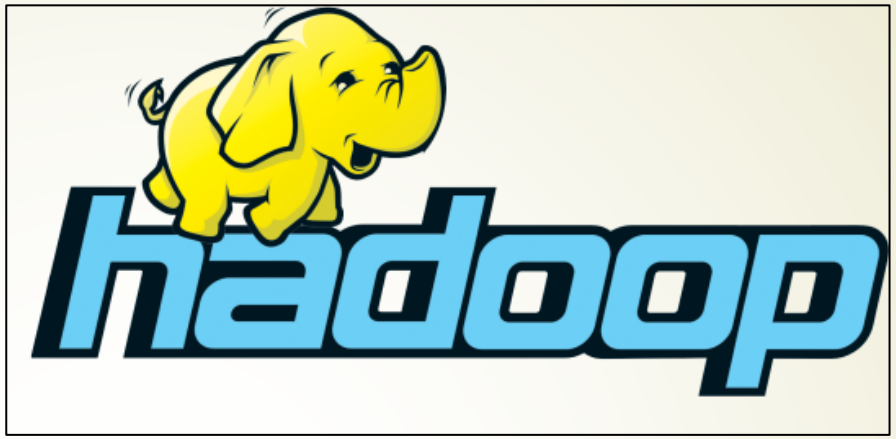
Chapter 6



Bal Krishna Nyaupane
Assistant Professor
Department of Electronics and Computer Engineering
Institute of Engineering, Tribhuvan University
bkn@wrc.edu.np

Terminology

Google calls it:	Hadoop equivalent:
MapReduce	Hadoop
GFS	HDFS
Bigtable	HBase
Chubby	Zookeeper

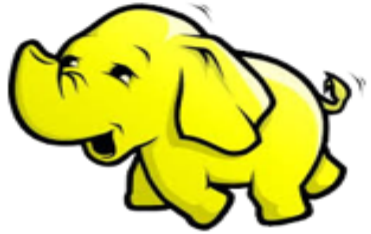
Hadoop's Developers

Doug Cutting

2005: Doug Cutting and Michael J. Cafarella developed Hadoop to support distribution for the Nutch search engine project.

The project was funded by Yahoo.

2006: Yahoo gave the project to Apache Software Foundation.



What is Hadoop?

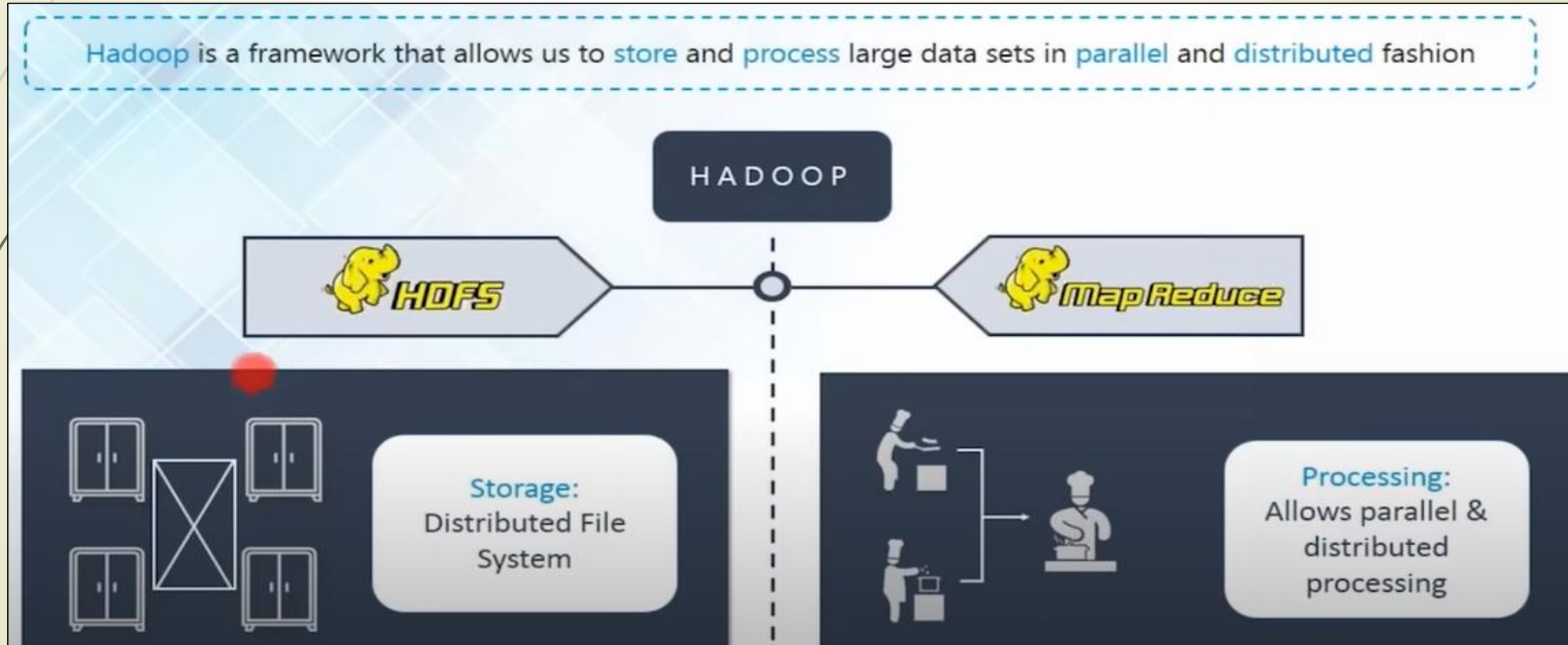
- ***Created by Doug Cutting and Mike Carafella in 2005.***
- ***Cutting named the program after his son's toy elephant.***
- ***Open source software framework*** designed for storage and processing of large scale data on clusters of commodity hardware
- ***Based on work done by Google in the early 2000s***
  - "The Google File System" in 2003
  - "MapReduce: Simplified Data Processing on Large Clusters" in 2004
- ***The core idea was to distribute the data as it is initially stored***
  - Each node can then perform computation on the data it stores without moving the data for the initial processing.

What is Hadoop?

- Apache Hadoop is an ***open source software framework*** that provides ***highly reliable distributed processing*** of ***large data sets*** using simple programming models.
- Hadoop, ***known for its scalability***, is built on clusters of ***commodity computers***, providing a cost-effective solution for storing and ***processing massive amounts*** of structured, semi-structured ***and unstructured data with no format requirements***.
- At ***Google MapReduce operation*** are run on a special file system called Google File System (GFS) ,***not open source***, that is highly optimized for this purpose.
- Doug Cutting and Yahoo! reverse engineered the GFS and called it Hadoop Distributed File System (HDFS).

What is Hadoop?

- The software framework that supports ***HDFS, MapReduce*** and other related entities is called the project Hadoop or simply Hadoop.
- This is distributed by Apache.



Core Hadoop Concept

Based on work done by Google in the early 2000s	• "The Google File System" in 2003 • "MapReduce: Simplified Data Processing on Large Clusters" in 2004
The core idea was to distribute the data as it is initially stored	• Each node can then perform computation on the data it stores without moving the data for the initial processing
Applications are written in a high-level programming language	• No network programming or temporal dependency
Nodes should communicate as little as possible	• A "shared nothing" architecture
Data is spread among the machines in advance	• Perform computation where the data is already stored as often as possible

Hadoop High-Level Overview

When data is loaded onto the system it is divided into blocks

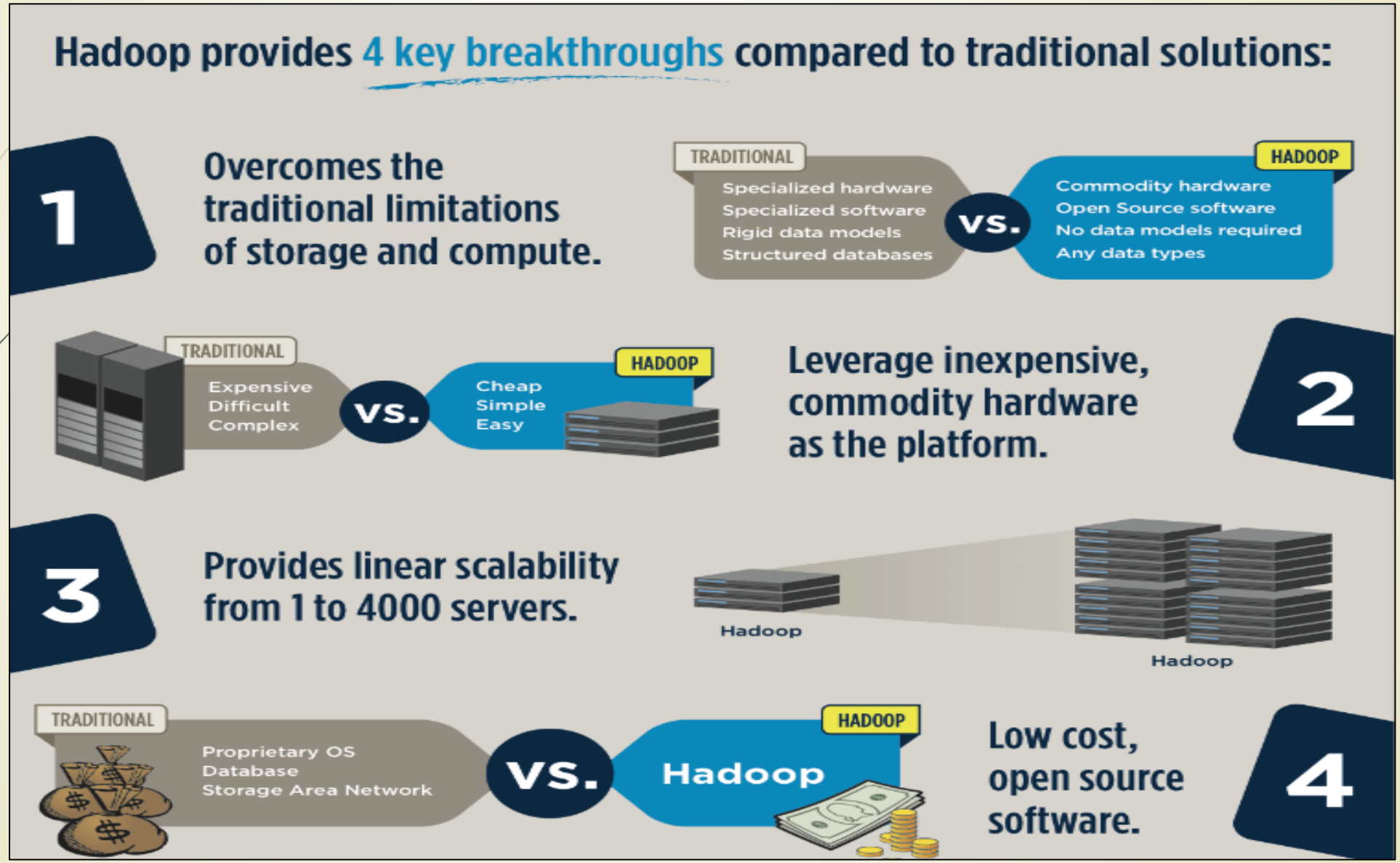
- Typically 64MB or 128MB

Tasks are divided into two phases

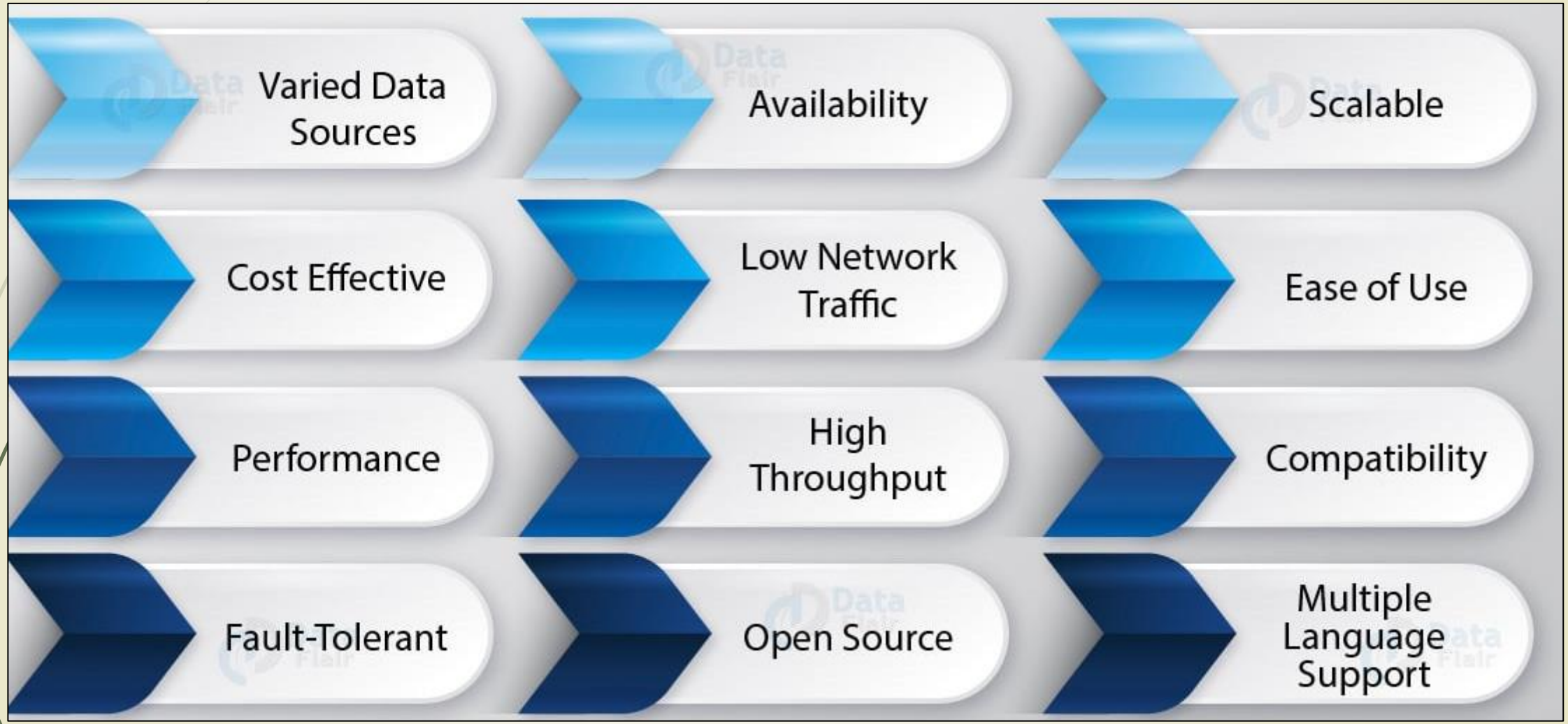
- Map tasks which are done on small portions of data where the data is stored
- Reduce tasks which combine data to produce the final output

A master program allocates work to individual nodes

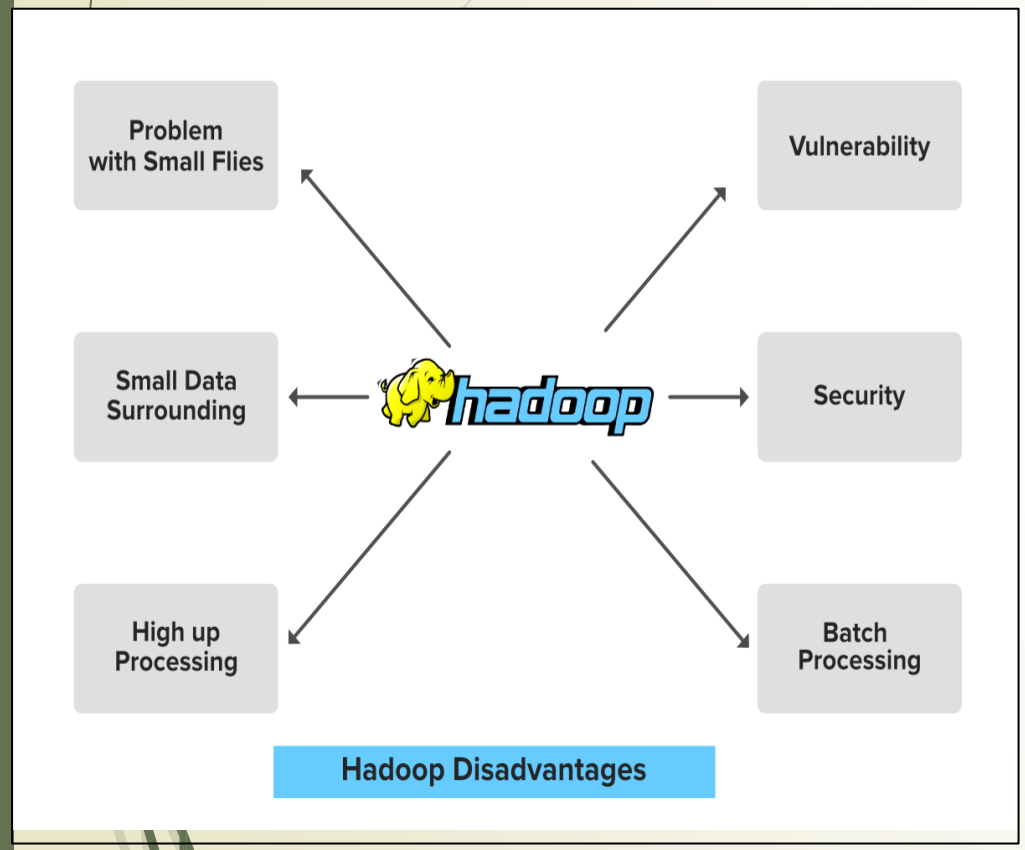
Why Hadoop?


Hadoop provides 4 key breakthroughs compared to traditional solutions:
1
Overcomes the traditional limitations of storage and compute.
TRADITIONAL
Specialized hardware
Specialized software
Rigid data models
Structured databases
VS.
HADOOP
Commodity hardware
Open Source software
No data models required
Any data types
TRADITIONAL
Expensive
Difficult
Complex
VS.
HADOOP
Cheap
Simple
Easy
Leverage inexpensive, commodity hardware as the platform.
2
3
Provides linear scalability from 1 to 4000 servers.
Hadoop
Hadoop
TRADITIONAL
Proprietary OS
Database
Storage Area Network
VS.
HADOOP
Hadoop
Low cost, open source software.
4

Why Hadoop?


Varied Data Sources
Availability
Scalable
Cost Effective
Low Network Traffic
Ease of Use
Performance
High Throughput
Compatibility
Fault-Tolerant
Open Source
Multiple Language Support

Disadvantages of Hadoop



- Hadoop is written in ***Java which is a widely used programming*** language hence it is ***easily exploited by cyber criminals*** which makes ***Hadoop vulnerable to security breaches***.
- At the core, Hadoop ***has a batch processing engine*** which is not efficient in stream processing. It ***cannot produce output in real-time with low latency***. It only works on data which we collect and store in a file in advance before processing.
- ***High Up Processing:*** In Hadoop, the ***data is read from the disk and written to the*** disk which makes read/write operations ***very expensive when we are dealing with tera and petabytes*** of data. Hadoop ***cannot do in-memory calculations*** hence it ***incurs processing overhead.***
- Hadoop can efficiently perform ***over a small number of files of large size.*** Hadoop stores the file in the form of file blocks which are from ***128MB in size(by default) to 256MB.***

Who Uses Hadoop?

Motivations for Hadoop

- *What were the limitations of earlier large-scale computing?*
- *What requirements should an alternative approach have?*
- *How does Hadoop address those requirements?*

Motivations for Hadoop

- ***Early Large Scale Computing***
  - Historically computation ***was processor-bound.***
  - Advances in computer technology has historically centered around improving the power of a single machine.
- ***Advances in CPUs***
  - ***Moore's Law:*** The number of transistors on a dense integrated circuit doubles every two years.
  - Single-core computing ***can't scale with current computing needs***
  - ***Single-Core Limitation:*** Power consumption limits the speed increase we get from transistor density.

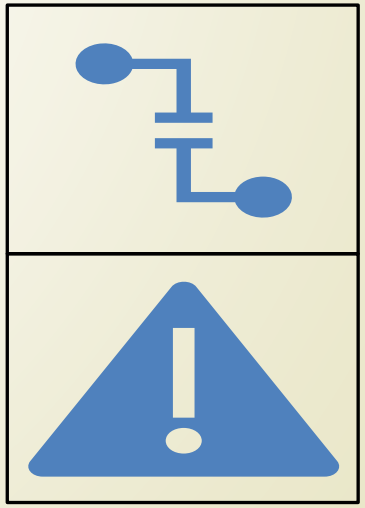
Motivations for Hadoop

- ***Distributed Systems:*** Allows developers to use multiple machines for a single task.
- ***Distributed System Problems:*** Programming on a distributed system is much more complex
  - Synchronizing data exchanges
  - Managing a finite bandwidth
  - Controlling computation timing is complicated

- ***Distributed systems must be designed with the expectation of failure***
- ***Distributed System Data Storage***
  - Typically divided into Data Nodes and Compute Nodes
  - At compute time, data is copied to the Compute Nodes
  - ***Fine for relatively small amounts of data***
  - ***Modern systems deal with far more data than was gathering in the past***

Requirements for Hadoop

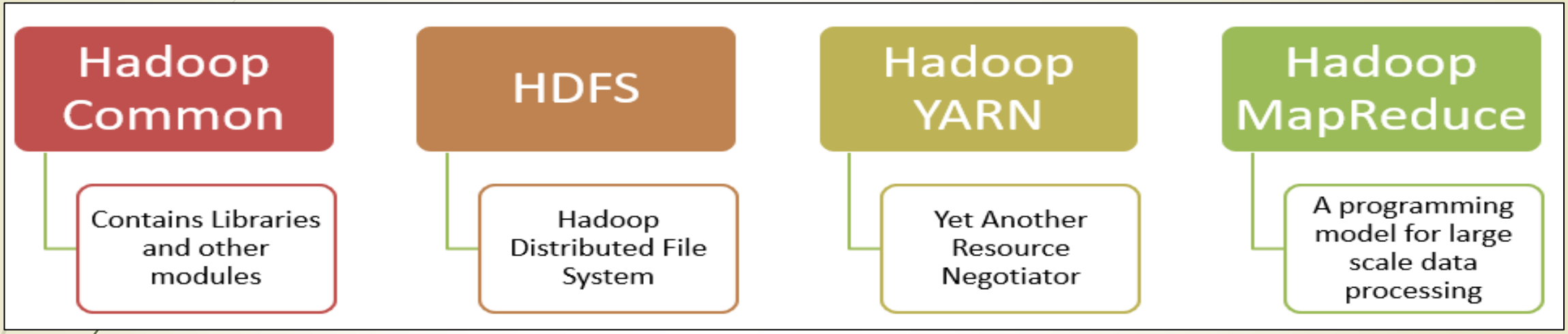
- ***Most support Partial Failures.***
  - Failure of a single component must not cause the failure of the entire system only a degradation of the application performance
  - Failure should not result in the loss of any data
- ***Most support Component Recovery.***
  - If a component fails, it ***should be able to recover without restarting*** the entire system
  - Component ***failure or recovery during a job must not affect*** the final output.



Requirements for Hadoop

- ***Must be scalable.***
  - ***Increasing resources*** should ***increase load capacity.***
  - Increasing the load on the system ***should result in a graceful decline in performance*** for all jobs
    - ***Not system failure***

The Hadoop Ecosystem



Hadoop Common

- The common utilities that support the other Hadoop modules.
- It is an essential part or module of the Apache Hadoop Framework, along with the Hadoop Distributed File System (HDFS), Hadoop YARN and Hadoop MapReduce.
- Hadoop Common is also known as Hadoop Core.

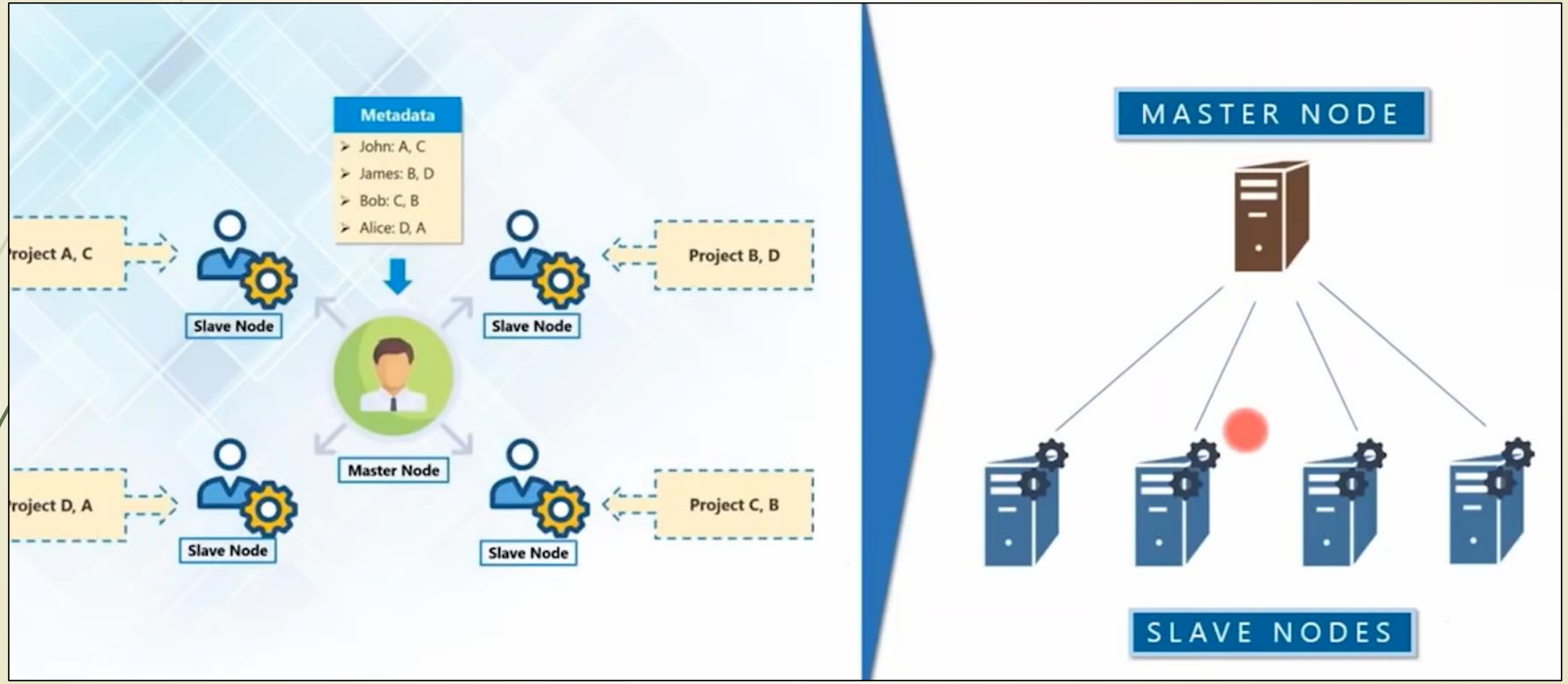
The Hadoop Ecosystem

- Hadoop framework **includes following four modules:**
  - ***Hadoop Common:*** These are **Java libraries and utilities** required by **other Hadoop modules**. These libraries **provides file system and OS level abstractions** and **contains the necessary Java files** and scripts required to start Hadoop.
  - ***Hadoop Distributed File System (HDFS™):*** A distributed file system that provides high-throughput access to application data.
  - ***Hadoop YARN:*** This is a **framework for job scheduling and cluster resource management.**
  - ***Hadoop MapReduce:*** This is YARN-based system **for parallel processing of large data sets.**

HDFS

- HDFS is a file system written in ***Java based on the Google's GFS***
- Responsible for ***storing data on the cluster***
- Provides ***redundant storage for massive amounts of data***
- Data ***files are split into blocks*** and ***distributed across the nodes*** in the cluster
- Each ***block is replicated*** multiple times
- HDFS works best with ***a smaller number of large files***
- Millions as opposed to billions of files
- Typically ***100MB or more per file***

HDFS: Master/Slave Architecture



HDFS: Master/Slave Architecture

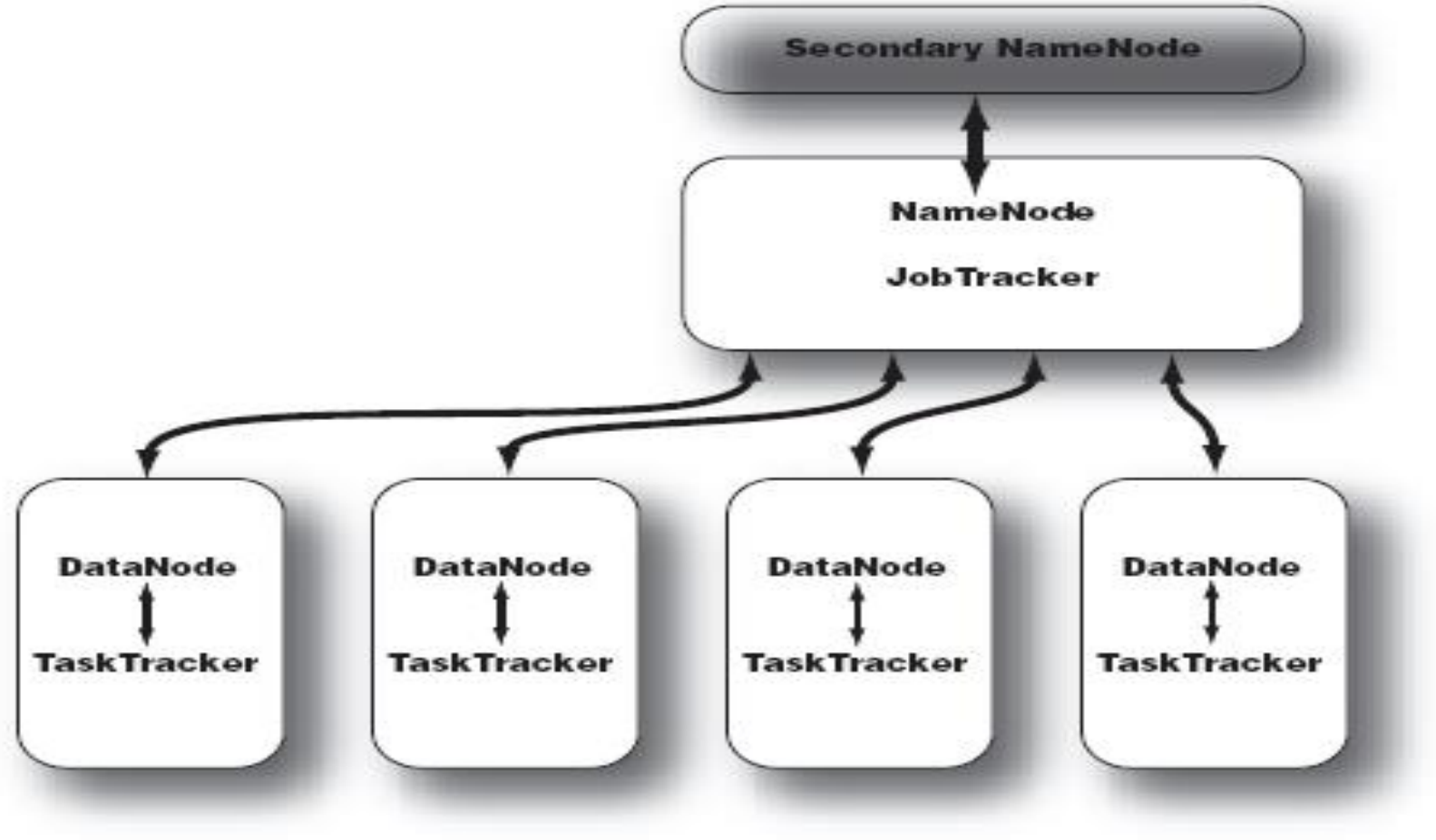


Figure 2.3 Topology of a typical Hadoop cluster. It's a master/slave architecture in which the NameNode and JobTracker are masters and the DataNodes and TaskTrackers are slaves.

HDFS: Master/Slave Architecture

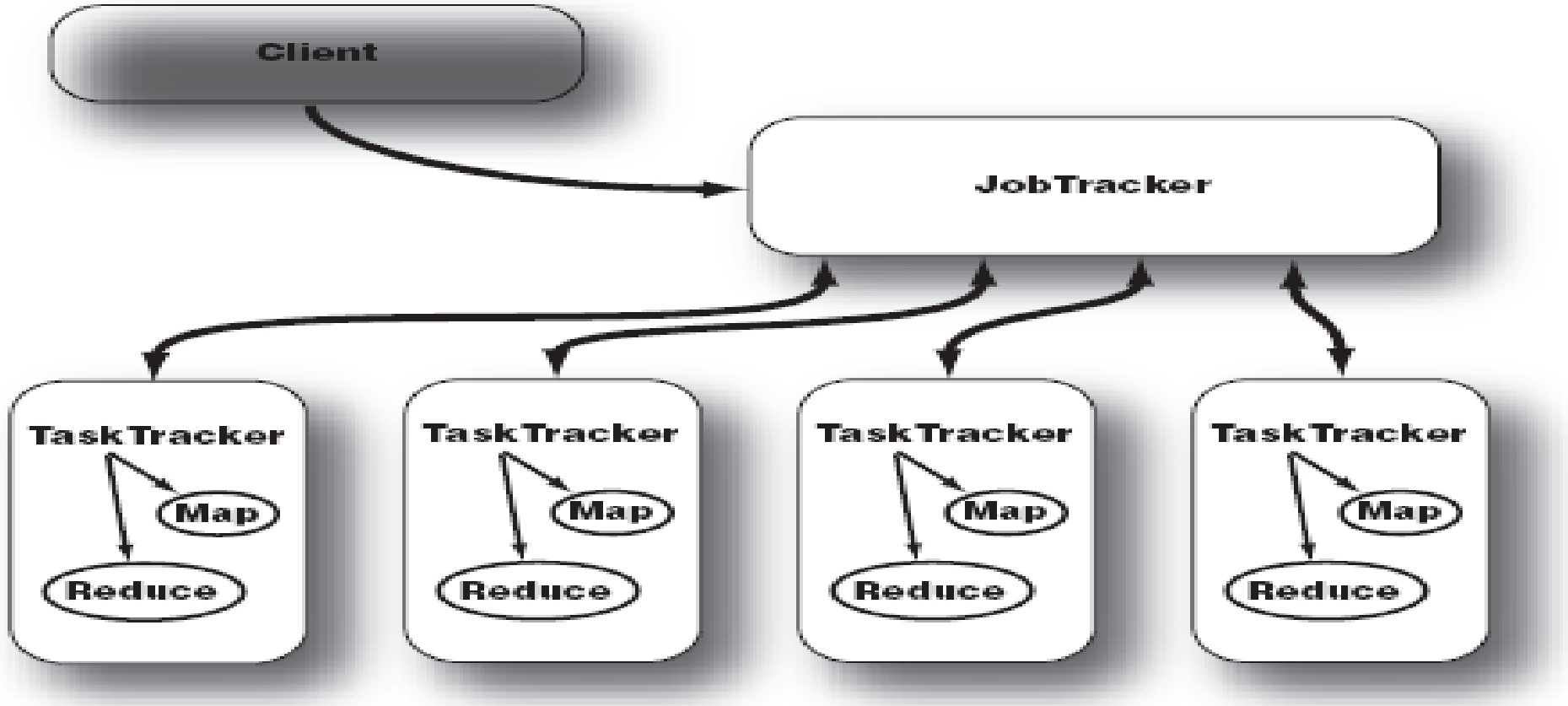
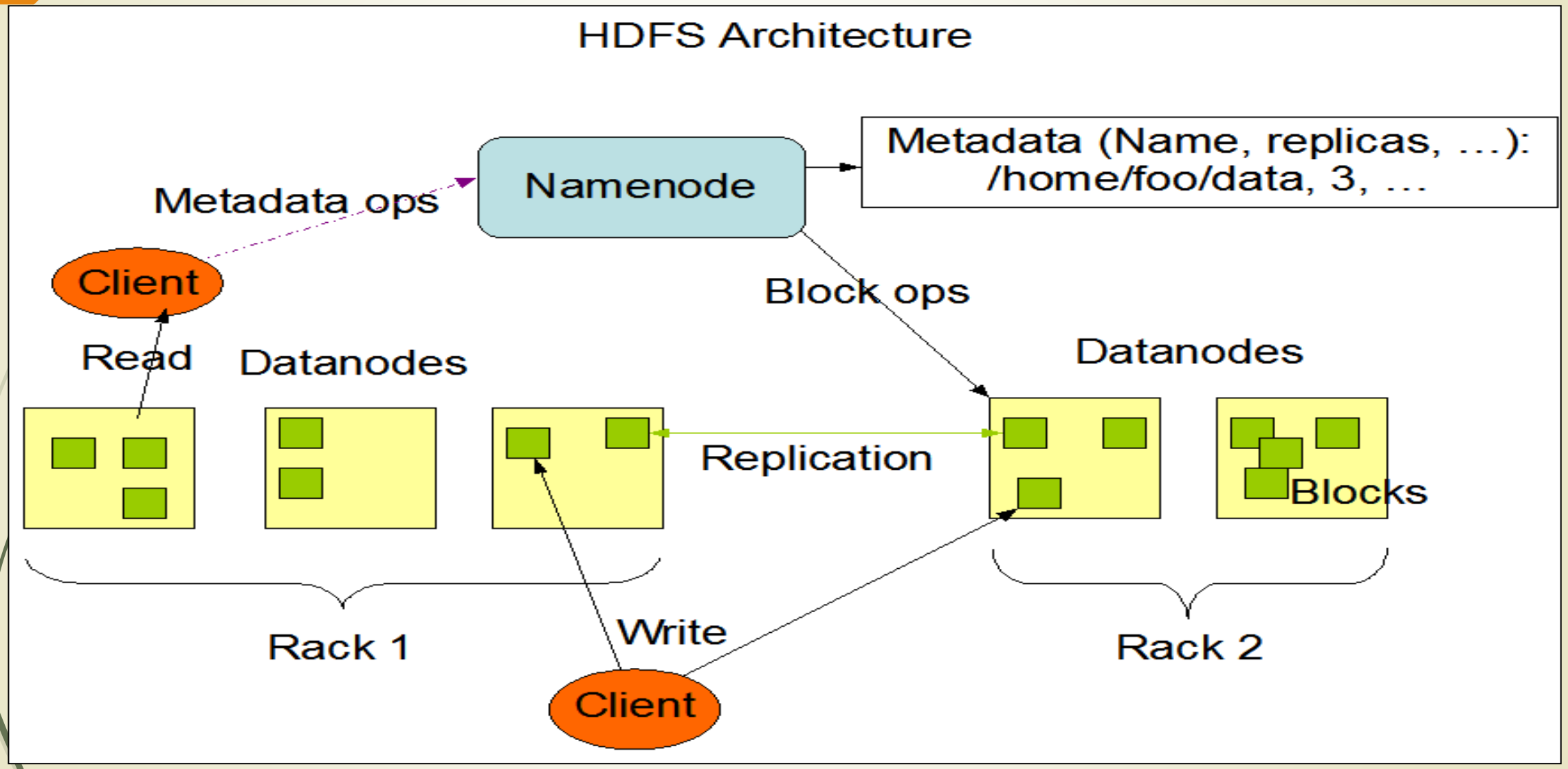


Figure 2.2 JobTracker and TaskTracker interaction. After a client calls the JobTracker to begin a data processing job, the JobTracker partitions the work and assigns different map and reduce tasks to each TaskTracker in the cluster.

HDFS Architecture



HDFS Architecture

- Hadoop Distributed File System ***follows the master-slave architecture***. Each cluster ***comprises a single master*** node and ***multiple slave nodes***.
- Internally the ***files get divided into one or more blocks***, and each block is ***stored on different slave machines*** depending on the replication factor.
- The ***master node*** stores and manages the ***file system namespace***, that is information about blocks of files like block locations, permissions, etc. The ***slave nodes store data blocks of files.***
- The Master node is the ***NameNode*** and ***DataNodes*** are the slave nodes.

What is HDFS NameNode?

- NameNode is the ***centerpiece of the HDFS***. It maintains and manages the ***file system namespace*** and provides the right ***access permission to the clients***.
- The **NameNode stores** information about ***blocks locations, permissions, etc***. on the local disk in the ***form of 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:*** It contains all the recent changes performed to the file system namespace to the most recent Fsimage.

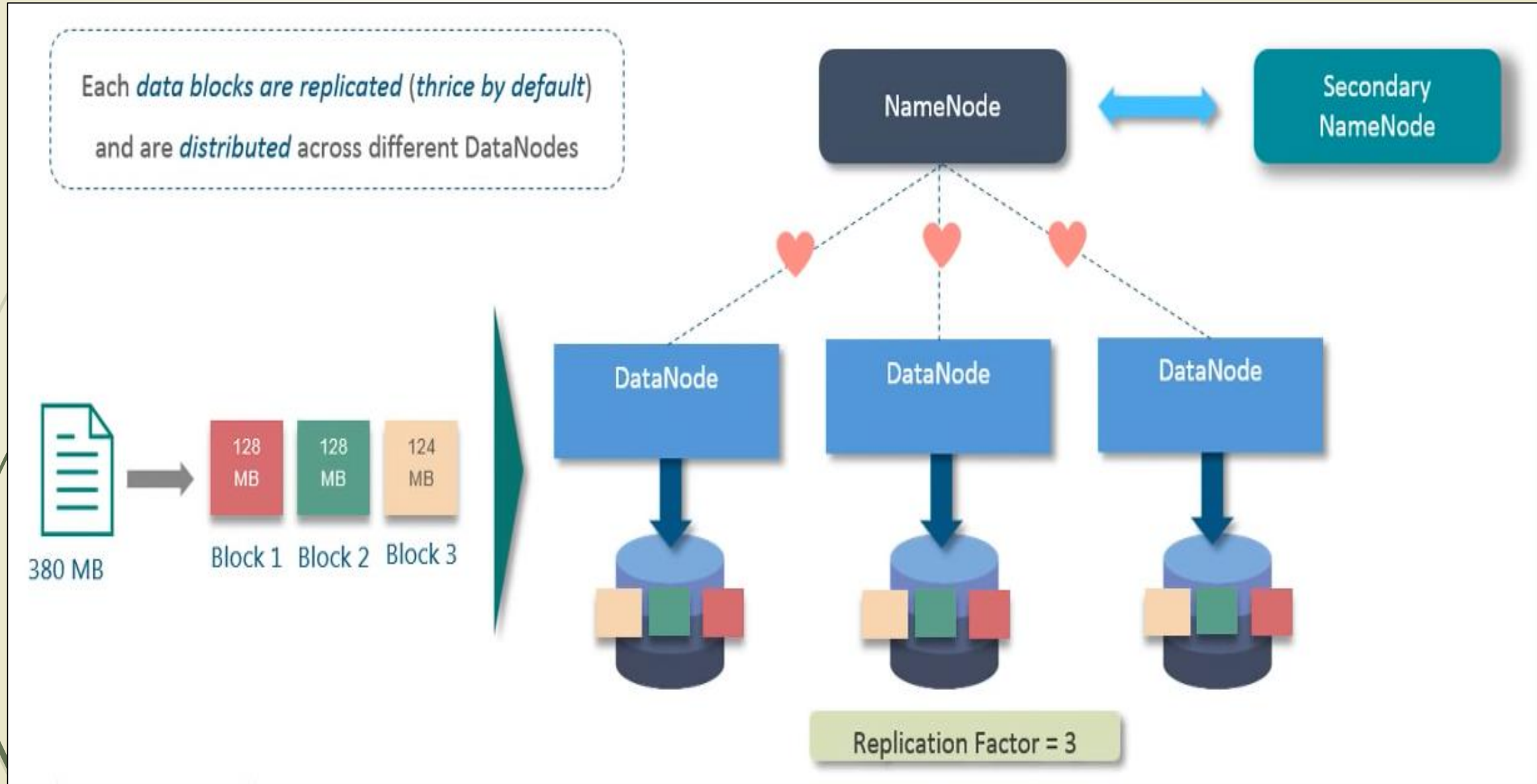
Functions of HDFS NameNode

1. It executes the file system namespace operations like opening, renaming, and closing files and directories.
2. NameNode manages and maintains the DataNodes.
3. It determines the mapping of blocks of a file to DataNodes.
4. NameNode records each change made to the file system namespace.
5. It keeps the locations of each block of a file.
6. NameNode takes care of the replication factor of all the blocks.
7. NameNode receives heartbeat and block reports from all DataNodes that ensure DataNode is alive.
8. If the DataNode fails, the NameNode chooses new DataNodes for new replicas.

What is HDFS DataNode?

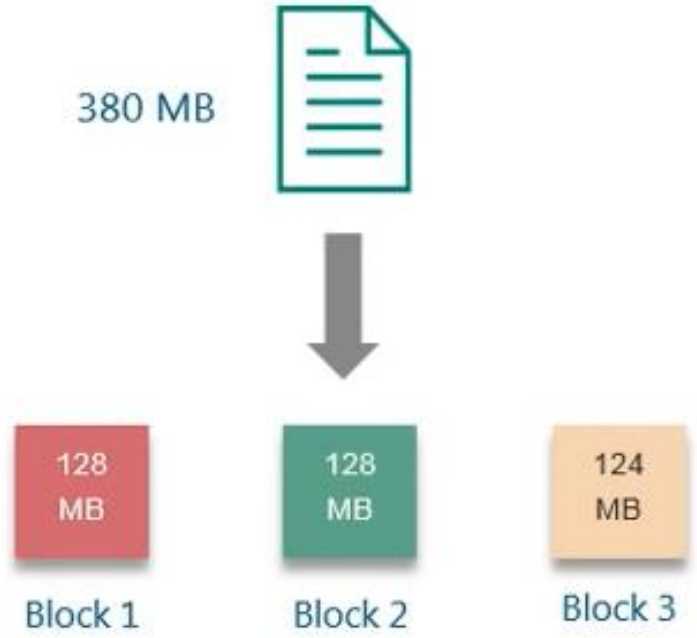
- DataNodes are the ***slave nodes in Hadoop HDFS***. DataNodes are ***inexpensive commodity hardware***. They store blocks of a file.
- ***Functions of DataNode***
  - DataNode is responsible for ***serving the client read/write requests***.
  - Based on the instruction from the NameNode, DataNodes 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.

HDFS: Data Replication

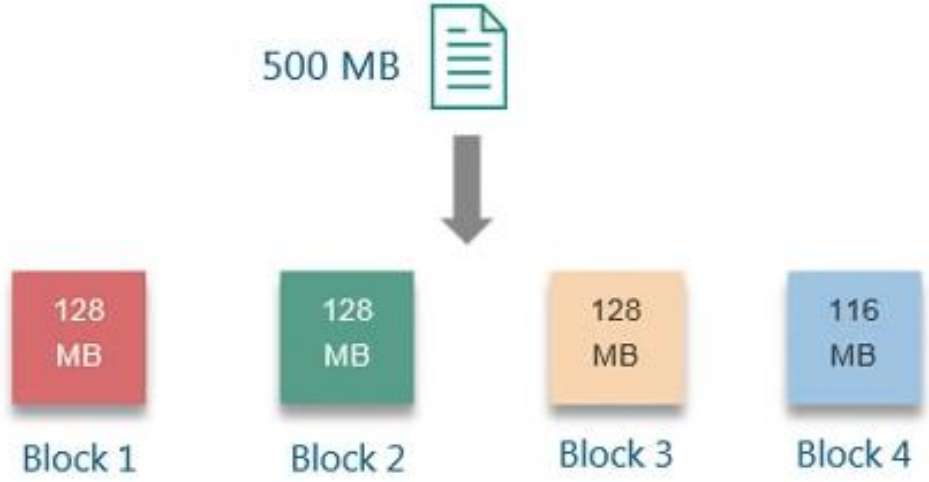


HDFS Data Block

Each file is stored on HDFS as block. The default size of each block is 128 MB

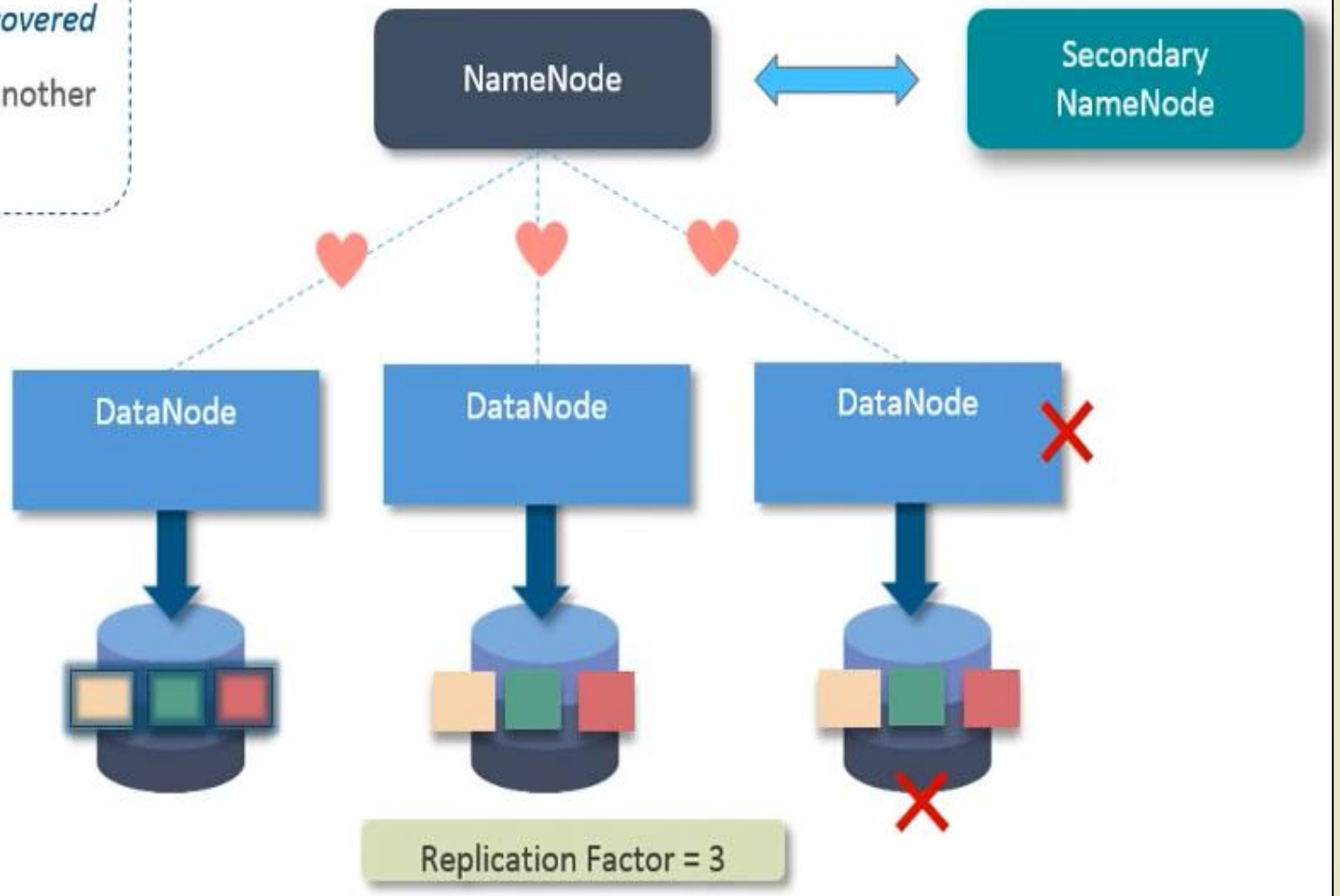


How many blocks will be created if a file of size 500 MB is copied to HDFS?



HDFS Fault Tolerance

If *a DataNode fails*, the *data blocks can be recovered* and retrieved *from the replicas* stored on another DataNodes.



What is YARN?

Hadoop v1.0

MapReduce
Data Processing
& Resource Management

HDFS
Distributed File Storage

Hadoop v2.0

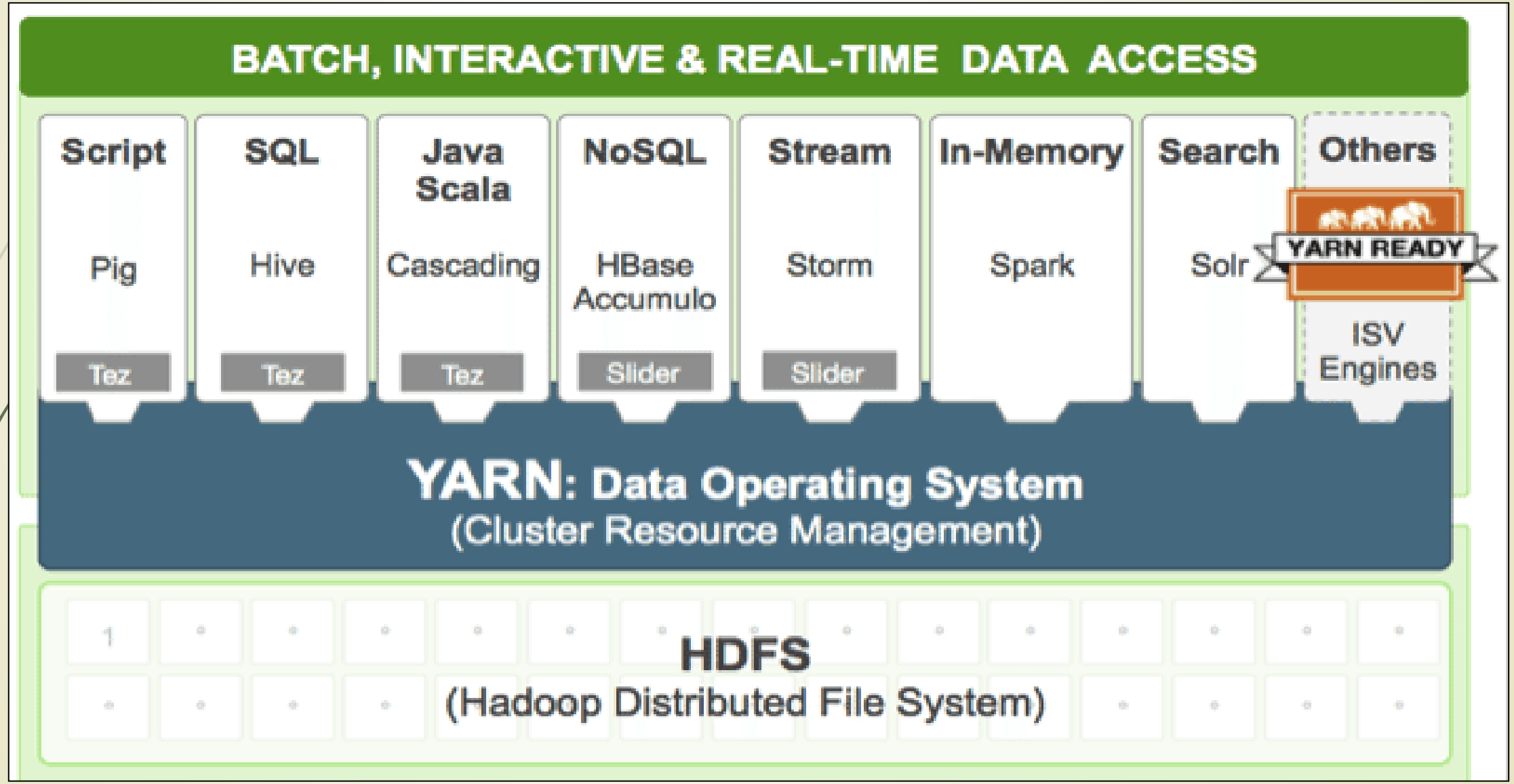
MapReduce

Other Data Processing Frameworks

YARN
Resource Management

HDFS
Distributed File Storage

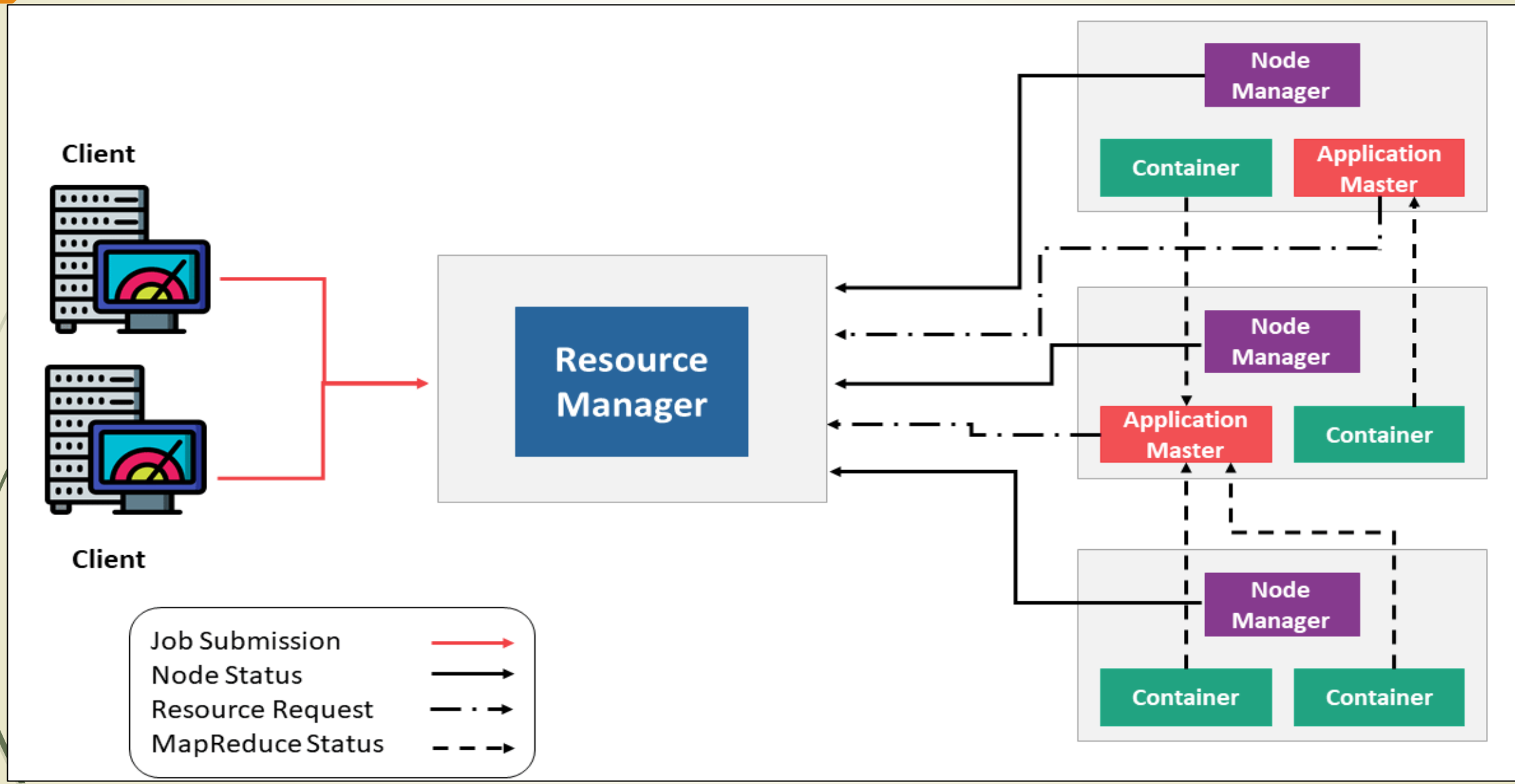
Hadoop YARN


BATCH, INTERACTIVE & REAL-TIME DATA ACCESS
Script
Pig
Tez
SQL
Hive
Tez
Java
Scala
Cascading
Tez
NoSQL
HBase
Accumulo
Slider
Stream
Storm
Slider
In-Memory
Spark
Search
Solr
Others
YARN READY
ISV
Engines
YARN: Data Operating System
(Cluster Resource Management)
HDFS
(Hadoop Distributed File System)

What is YARN?

- YARN stands for "***Yet Another Resource Negotiator***".
- It was introduced in Hadoop 2.0 to remove the ***bottleneck on Job Tracker*** which was present ***in Hadoop 1.0.***
- YARN was described as a ***"Redesigned Resource Manager***" at the time of its launching, but it has now ***evolved to be known as large-scale distributed operating system*** used for Big Data processing.
- YARN ***allows different data processing methods*** like ***graph processing, interactive processing, stream processing as well as batch processing*** to run and process data stored in HDFS. Therefore YARN opens up Hadoop to other types of distributed applications beyond MapReduce.
- YARN enabled the ***users to perform operations*** as per requirement by using a variety of tools like ***Spark*** for real-time processing, ***Hive*** for SQL, ***HBase*** for NoSQL and others.

Components of YARN



Components of YARN

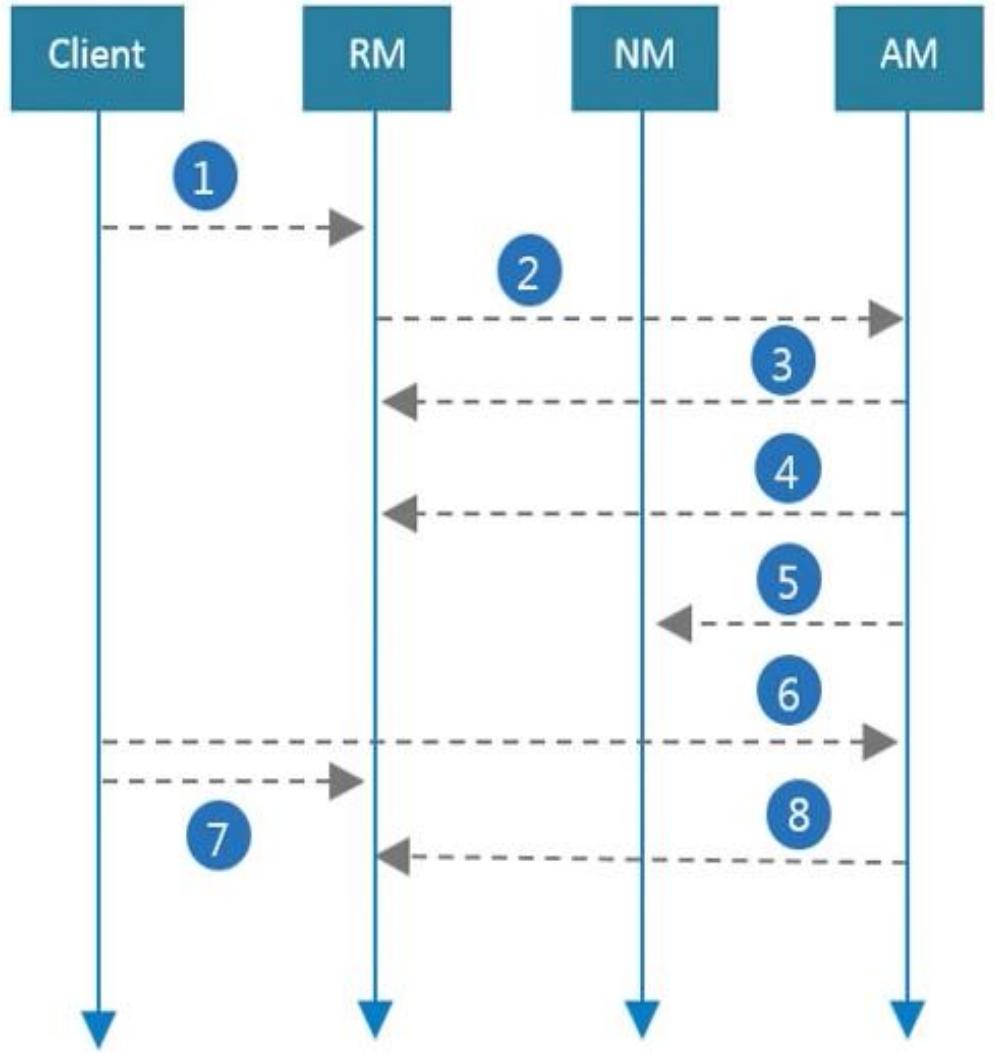
- Apache Hadoop ***YARN Architecture consists of the following main components.***
- ***Resource Manager***
  - It is the ***ultimate authority*** in resource allocation.
  - On receiving the processing requests, it passes parts of requests to ***corresponding node managers accordingly***, where the actual processing takes place.
  - It is the arbitrator of the cluster resources and ***decides the allocation of the available resources for competing applications***.
  - It has ***two major components***: a) Scheduler b) Application Manager
  - The ***scheduler is responsible for allocating resources*** to the various running applications subject to constraints of capacities, queues etc.
  - Application Manager is ***responsible for accepting job submissions***.

Components of YARN

- ***Node Manager***
  - They run on the slave daemons and are responsible for the execution of a task on every single Data Node.
  - It takes care of individual nodes in a Hadoop cluster and manages user jobs and workflow on the given node.
  - It registers with the Resource Manager and sends heartbeats with the health status of the node.
- ***Application Master:*** Manages the user job lifecycle and resource needs of individual applications. It works along with the Node Manager and monitors the execution of tasks.
- ***Container:*** Package of resources including RAM, CPU, Network, HDD etc. on a single node.

YARN Application Workflow

1. Client submits an application
2. RM allocates a container to start AM
3. AM registers with RM
4. AM asks containers from RM
5. AM notifies NM to launch containers
6. Application code is executed in container
7. Client contacts RM/AM to monitor application's status
8. AM unregisters with RM



Hadoop Daemons

- ***Daemons mean Process. Hadoop Daemons are a set of processes that run on Hadoop.*** Hadoop is a framework written in Java, so all these processes are Java Processes. Apache Hadoop 2 consists of the ***following Daemons:***
  1. ***NameNode***
  2. ***DataNode***
  3. ***Secondary Name Node***
  4. ***Resource Manager***
  5. ***Node Manager***
- ***"Running Hadoop"*** means running a set of daemons, or resident programs, on the different servers in your network***.***
- ***These daemons have specific roles;*** some exist only on one server, some exist across multiple servers.

Hadoop Daemons: NameNode

- The NameNode(master) ***directs the slave DataNode daemons*** to perform the ***low-level I/O tasks***.
- The NameNode is ***bookkeeper of HDFS***; it keeps track of how your files are broken down into file blocks, which nodes store those blocks, and the overall health of the distributed file system.
- The function of the ***NameNode is memory and I/O intensive***. As such, the server hosting the NameNode typically doesn't store any user data or perform any computations for a MapReduce program to lower the workload on the machine.
- ***It's a single point of failure of your Hadoop cluster.***
- For any of the other daemons, if their host nodes fail for software or hardware reasons, the Hadoop cluster will likely continue to function smoothly or you can quickly restart it. ***Not so for the NameNode.***

Hadoop Daemons: DataNode

- Each slave machine in cluster ***host a DataNode daemon*** to perform work of the distributed file system***, reading and writing HDFS blocks to actual files on the local file system.*** Read or write a HDFS file, the file is broken into blocks and ***the NameNode will tell your client which DataNode each block resides in.***
- Job communicates directly with the DataNode daemons to process the local files corresponding to the blocks.
- DataNode may communicate with other DataNodes to replicate its data blocks for redundancy.
- DataNodes are ***constantly reporting to the NameNode***. Upon initialization, each of the DataNodes informs the NameNode of the ***blocks it's currently storing***.
- After this mapping is complete, the DataNodes continually poll the NameNode to provide information regarding local changes ***as well as receive instructions to create, move, or delete blocks from the local disk.***

Hadoop Daemons: Secondary Name Node

- The Secondary NameNode (SNN) is ***an assistant daemon for monitoring the state of the cluster HDFS.***
- Like the NameNode, ***each cluster has one SNN***.
- No other DataNode or TaskTracker daemons run on the same server.
- The SNN differs from the NameNode, ***it doesn't receive or record any real-time changes to HDFS***. Instead, it ***communicates with the NameNode to take snapshots of the HDFS metadata*** at intervals defined by the cluster configuration.
- As mentioned earlier, the NameNode is a single point of failure for a Hadoop cluster, and the ***SNN snapshots help minimize the downtime and loss of data.***
- Nevertheless, a NameNode failure requires human intervention to reconfigure the cluster to use the SNN as the primary NameNode.

Hadoop Daemons: Resource Manager

- Resource Manager is also known as the ***Global Master Daemon*** that ***works on the Master System.*** The Resource Manager Manages the ***resources for the applications that are running*** in a Hadoop Cluster.
- The Resource Manager Mainly ***consists of 2 things***.
  1. ***Applications Manager***
  2. ***Scheduler***
- An Application Manager is responsible for ***accepting the request for a client*** and also makes a ***memory resource on the Slaves in a Hadoop cluster*** to host the Application Master.
- The scheduler is utilized for ***providing resources for applications*** in a Hadoop cluster and for ***monitoring this application***.

JobTracker

- JobTracker process runs on a separate node and not usually on a DataNode.
- JobTracker is an essential Daemon for MapReduce execution in MRv1. ***It is replaced by ResourceManager/ApplicationMaster in MRv2.***
- JobTracker receives the requests for MapReduce execution from the client.
- JobTracker talks to the NameNode to determine the location of the data.
- JobTracker finds the best TaskTracker nodes to execute tasks based on the data locality (proximity of the data) and the available slots to execute a task on a given node.
- JobTracker monitors the individual TaskTrackers and the submits back the overall status of the job back to the client.
- JobTracker process is critical to the Hadoop cluster in terms of MapReduce execution.
- When the JobTracker is down, HDFS will still be functional but the MapReduce execution can not be started and the existing MapReduce jobs will be halted.

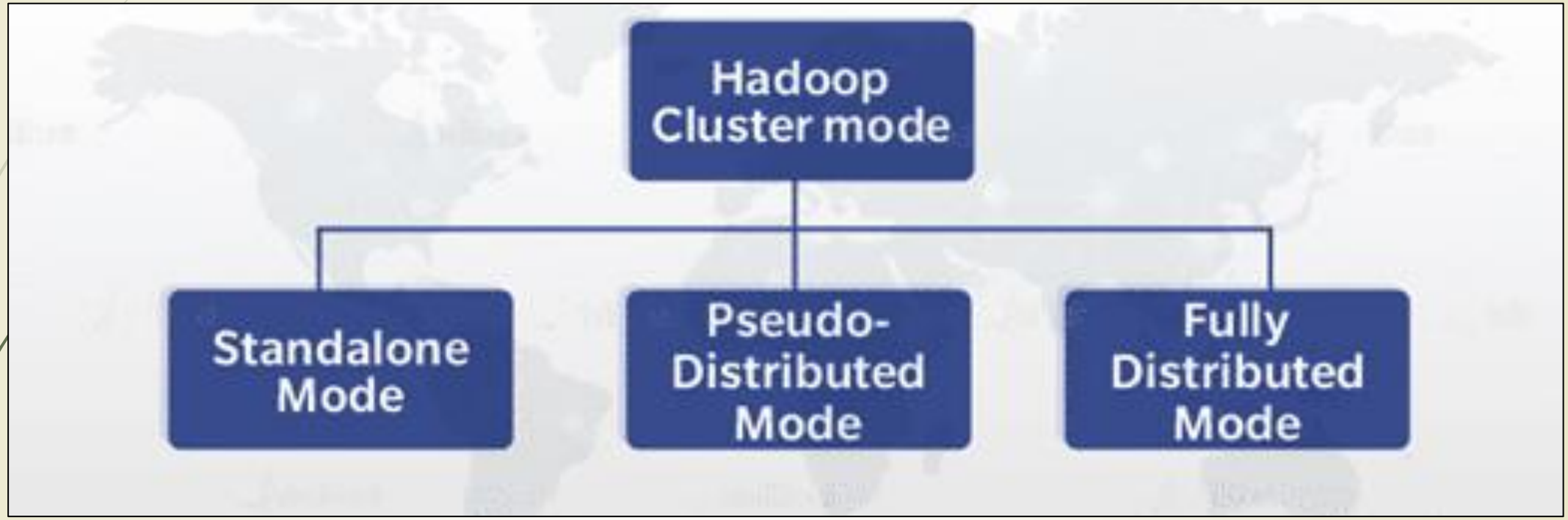
TaskTracker

- TaskTracker runs on DataNode. Mostly on all DataNodes.
- ***TaskTracker is replaced by Node Manager in MRv2.***
- Mapper and Reducer tasks are executed on DataNodes administered by TaskTrackers.
- TaskTrackers will be assigned Mapper and Reducer tasks to execute by JobTracker.
- TaskTracker will be in constant communication with the JobTracker signalling the progress of the task in execution.
- TaskTracker failure is not considered fatal. When a TaskTracker becomes unresponsive, JobTracker will assign the task executed by the TaskTracker to another node.

Hadoop Daemons: Node Manager

- The Node Manager ***works on the Slaves System*** that manages the ***memory resource within the Node and Memory Disk***.
- Each Slave Node in a Hadoop cluster has a single NodeManager Daemon running in it.
- It also ***sends this monitoring information*** to the Resource Manager.

Hadoop Configuration Modes



Standalone Mode

- The standalone mode is the ***default mode for Hadoop***.
- Hadoop chooses to be conservative and assumes a minimal configuration. ***All XML (Configuration) files are empty*** under this default mode. With empty configuration files, ***Hadoop will run completely on the local machine.***
- In Standalone Mode ***none of the Daemon will run*** i.e. Namenode, Datanode, Secondary Name node, Job Tracker, and Task Tracker. We use job-tracker and task-tracker for processing purposes in Hadoop1. For Hadoop2 we use Resource Manager and Node Manager.
- Standalone Mode also means that ***we are installing Hadoop only in a single system.***
- We ***mainly use Hadoop*** in this Mode for the ***Purpose of Learning, testing, and debugging.***

Pseudo Distributed Mode

- The pseudo-distributed mode is ***running Hadoop in a "cluster of one"*** with ***all daemons running on a single machine.*** This mode ***complements the standalone mode*** for debugging your code, ***allowing you to examine*** memory usage, HDFS input/output issues, and other daemon interactions.
- Namenode and Resource Manager are used as Master and Datanode and Node Manager is used as a slave. A secondary name node is also used as a Master. The purpose of the Secondary Name node is to just keep the hourly based backup of the Name node
- ***In this Mode,***
  - Hadoop is used for development and for debugging purposes both.
  - Our HDFS(Hadoop Distributed File System) is utilized for managing the Input and Output processes.
  - We need to change the configuration files ***mapred-site.xml, core-site.xml, hdfs-site.xml*** for setting up the environment.

Fully Distributed Mode

- This is the most important one in which ***multiple nodes are used*** few of them ***run the Master Daemon's*** that are ***Namenode and Resource Manager*** and the ***rest of them run the Slave Daemon's*** that are ***DataNode and Node Manager***.
- Benefits of distributed storage and distributed computation
  - ***Master:*** the server that hosts the namenode and job-tracker daemons
  - ***Backup:*** the server that hosts the secondary namenode daemon
  - ***Slaves:*** the servers that host both datanode and tasktracker daemons

Thank You

???