
hadoop

Why Hadoop is Invented?

- **Shortcomings of the traditional approach** which led to the invention of Hadoop –
- **Storage for Large Datasets**

➢ The conventional RDBMS is incapable of storing huge amounts of Data. The cost of data storage in available RDBMS is very high. As it incurs the cost of hardware and software both.

- **Handling data in different formats**

➢ The RDBMS is capable of storing and manipulating data in a structured format. But in the real world we have to deal with data in a structured, unstructured and semi-structured format.

Why Hadoop is Invented?

- **Data getting generated with high speed:**

➢ The data in oozing out in the order of tera to peta bytes daily. Hence we need a system to process data in real-time within a few seconds. The traditional RDBMS fail to provide real-time processing at great speeds.

Why Hadoop?

- Apache Hadoop is not only a storage system but is a platform for data storage as well as processing. It is **scalable** (as we can add more nodes on the fly), **Fault-tolerant** (Even if nodes go down, data processed by another node).
- Following characteristics of Hadoop make it a unique platform:

➢ Flexibility to store and mine any type of data whether it is structured, semi-structured or unstructured. It is not bounded by a single schema.

➢ Excels at processing data of complex nature. Its scale-out architecture divides workloads across many nodes. Another added advantage is that its flexible file-system eliminates ETL bottlenecks.

➢ Scales economically, as discussed it can deploy on commodity hardware. Apart from this its open-source nature guards against vendor lock.

Fault Tolerance

- Failures are detected by the master program which reassigns the work to a different node
- Restarting a task does not affect the nodes working on other portions of the data
- If a failed node restarts, it is added back to the system and assigned new tasks
- The master can redundantly execute the same task to avoid slow running nodes

What is Hadoop?

- Hadoop is the solution to above Big Data problems. It is the technology to **store massive datasets** on a cluster of cheap machines in a **distributed manner.** Not only this it provides Big Data analytics through distributed computing framework.
- It is an **open-source software** developed as a project by Apache Software Foundation. **Doug Cutting** created Hadoop. In the year 2008 Yahoo gave Hadoop to Apache Software Foundation. Since then three versions of Hadoop has come. Version 1.0 in the year 2011, version 2.0.6 in the year 2013 and Version 3.1.x – released on 21 October 2019. Hadoop comes in various flavors like Cloudera, IBM BigInsight, MapR and Hortonworks.

Hadoop Advantages

- Unlimited data storage
  1. Server Scaling Mode
     a) Vertical Scale
     b) Horizontal Scale
- High speed processing system
- All varieties of data processing
  1. Structural
  2. Unstructural
  3. semi-structural

Uses for Hadoop

- Data-intensive text processing
- Graph mining
- Machine learning and data mining
- Large scale social network analysis

Who Uses Hadoop?

Prerequisites to Learn Hadoop

- **Familiarity with some basic Linux Command –** Hadoop is set up over Linux Operating System preferable Ubuntu. So one must know certain ***basic Linux commands***. These commands are for uploading the file in HDFS, downloading the file from HDFS and so on.
- **Basic Java concepts –** Folks want to learn Hadoop can get started in Hadoop while simultaneously grasping basic concepts of JAVA. We can write **map and reduce functions** in Hadoop using other languages too. And these are **Python, Perl, C, Ruby**, etc. This is possible via streaming API. It supports reading from standard input and writing to standard output. Hadoop also has high-level abstractions tools like Pig and Hive which do not require familiarity with Java.

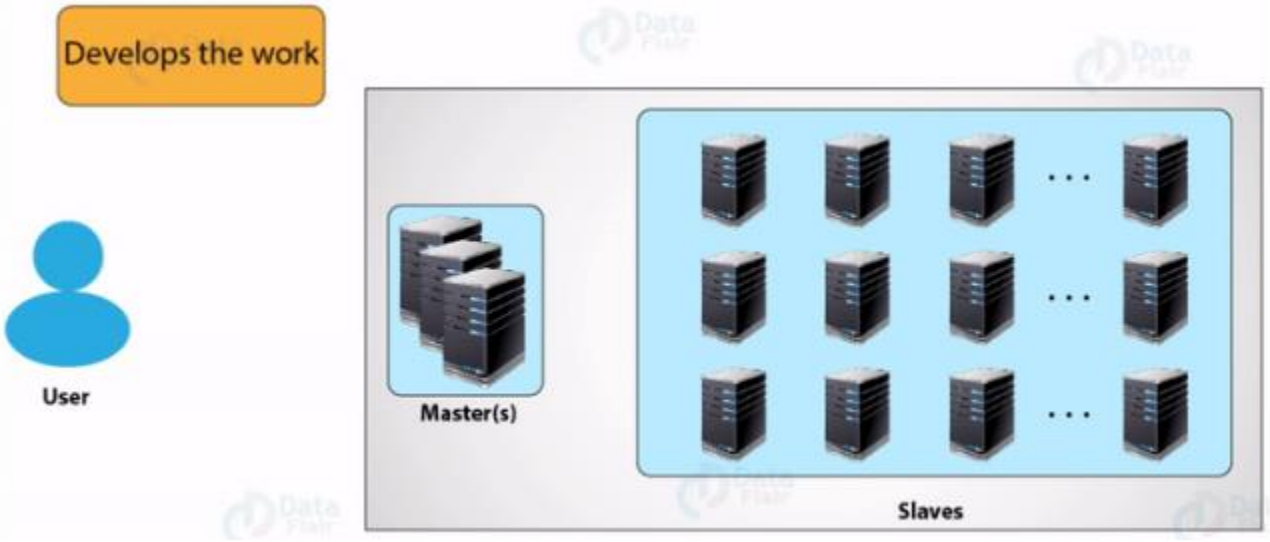
Hadoop

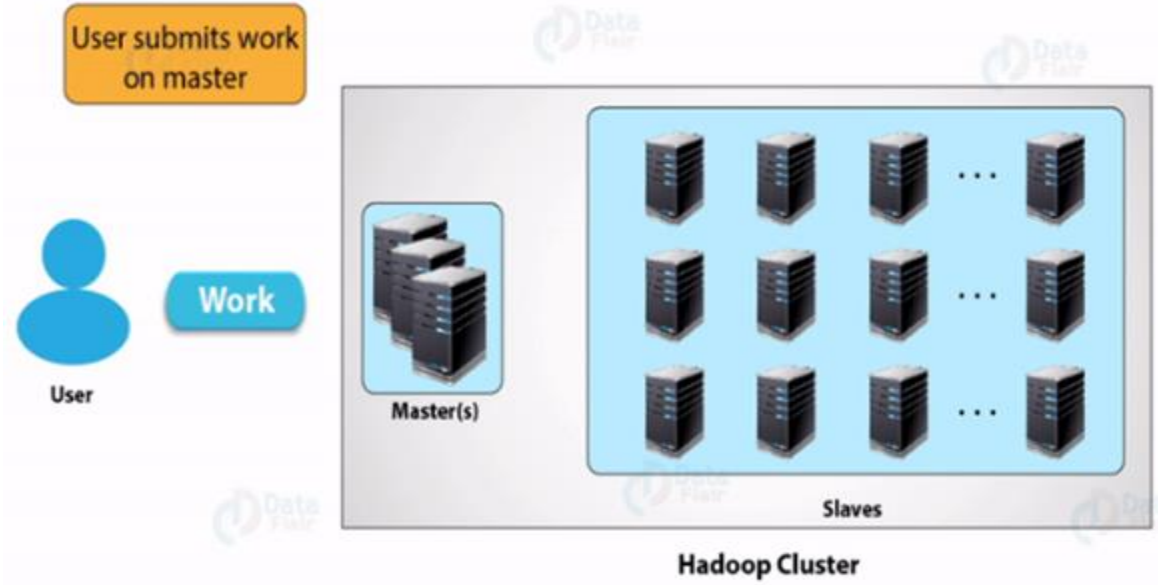
- Hadoop is Open Source Framework and Distributed Under Apache
- When should we go for Hadoop?
  - Data is too huge
  - Processes are independent
  - Better scalability
  - Parallelism
  - Unstructured data
- Hadoop Roles
  - Hadoop Admin
  - Hadoop Developer
- Hadoop Support
  - Java, Scala, Python, C++, C, Ruby

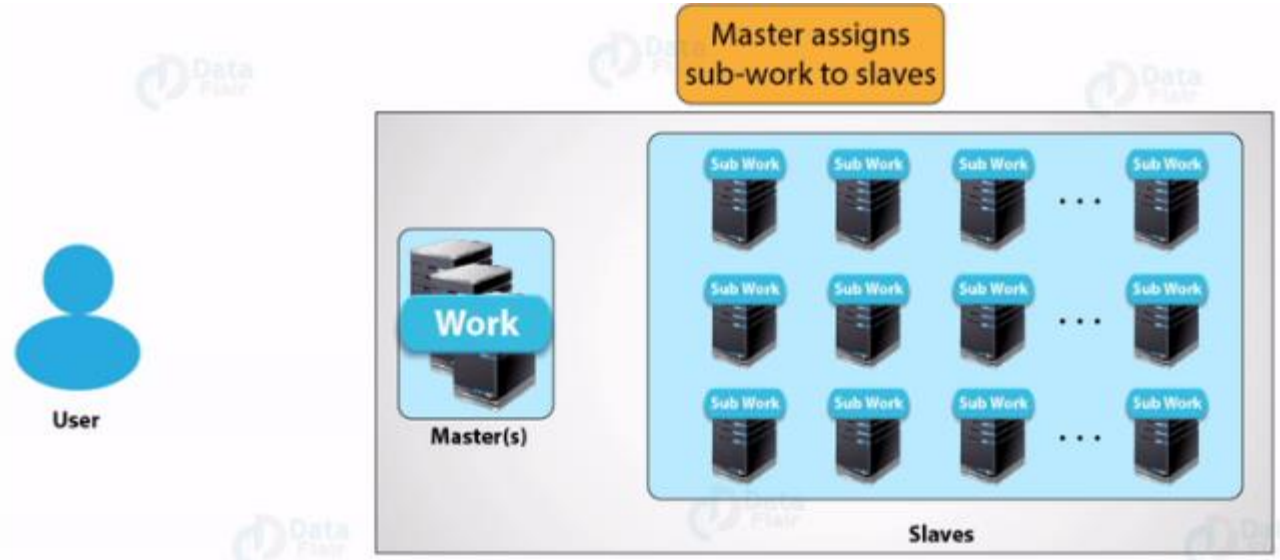
What is Hadoop Architecture?

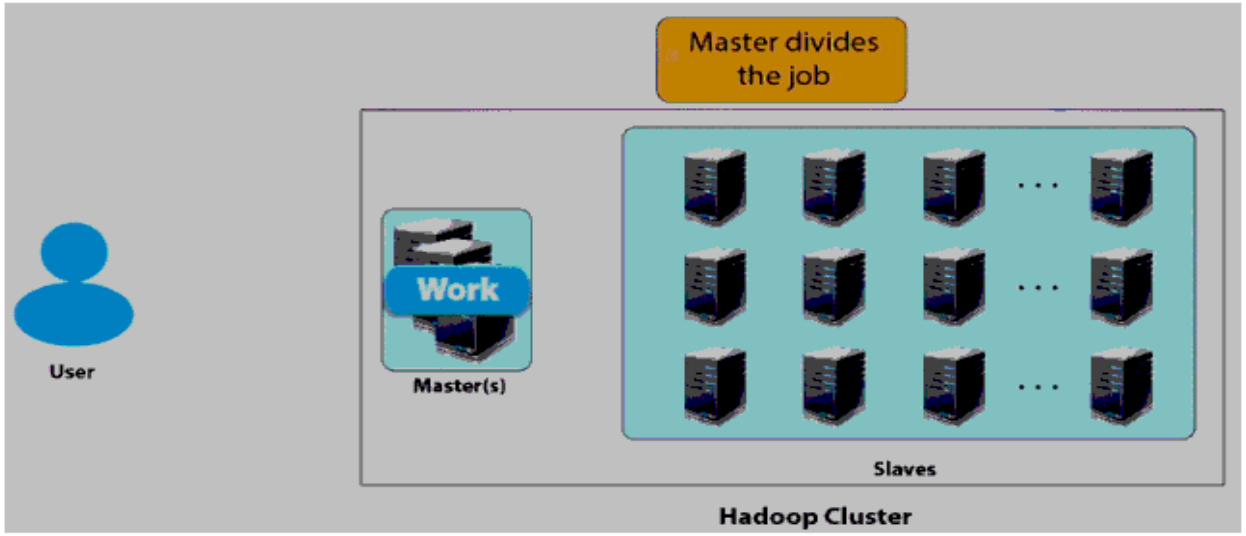
- Hadoop works in master-slave fashion. There is a **master node** and there are **n numbers of slave nodes** where n can be 1000s. Master manages, maintains and monitors the slaves while slaves are the actual worker nodes. In Hadoop architecture, the Master should deploy on good configuration hardware, not just commodity hardware. As it is the centrepiece of Hadoop cluster.
- Master stores the metadata (data about data) while slaves are the nodes which store the data. Distributedly data stores in the cluster. The client connects with the master node to perform any task.

What is Hadoop Architecture?









Core Components of Hadoop

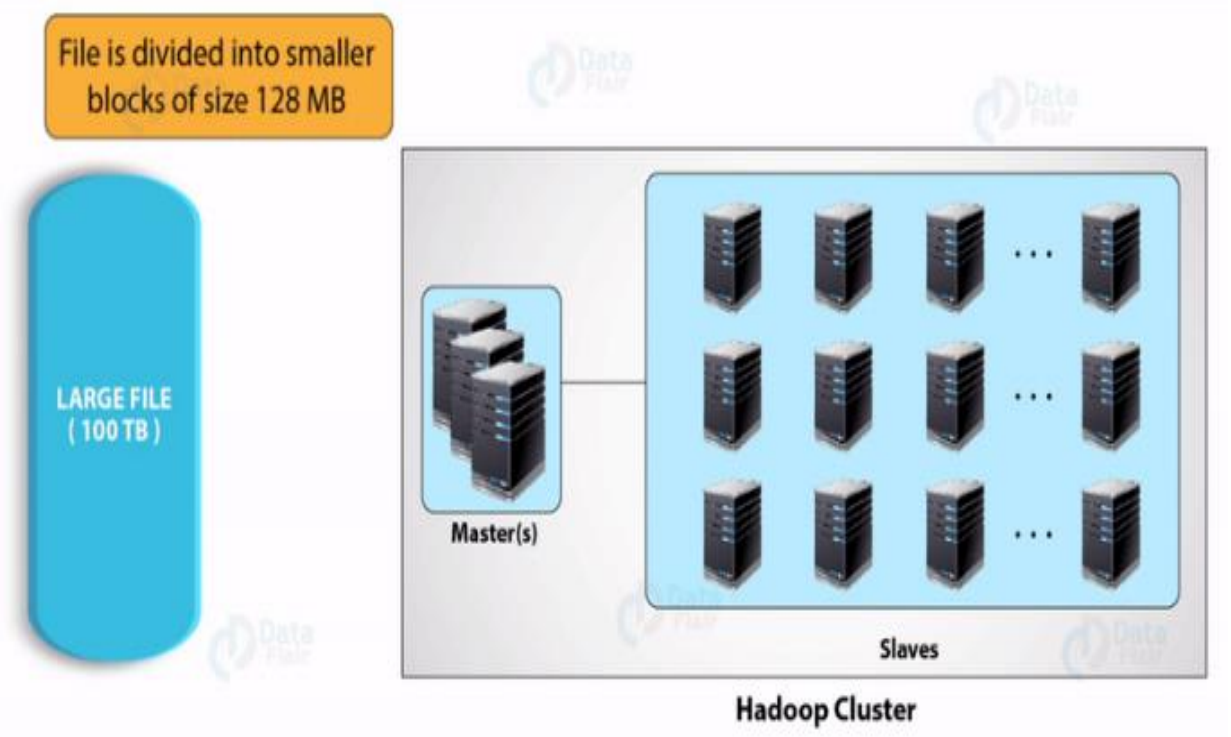
- Hadoop consists of three core components –

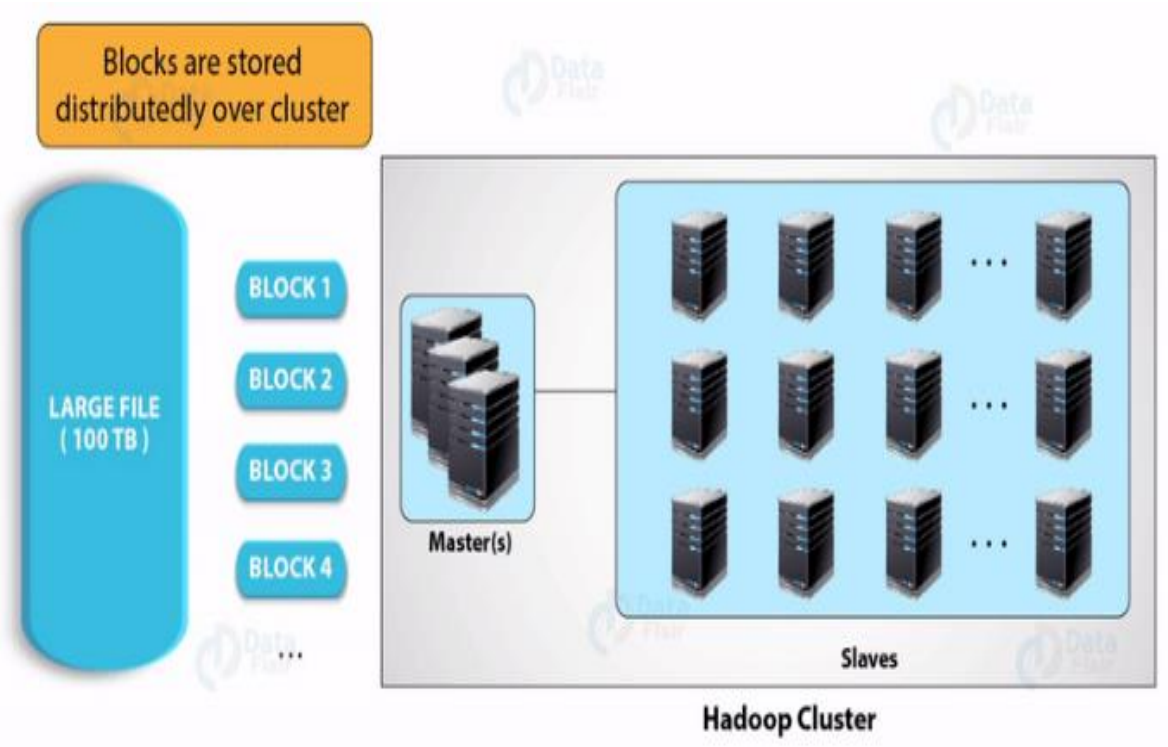
- **Hadoop Distributed File System (HDFS) –** It is the storage layer of Hadoop.
- **Map-Reduce –** It is the data processing layer of Hadoop.
- **YARN –** It is the resource management layer of Hadoop.

Core Components of Hadoop

1. HDFS

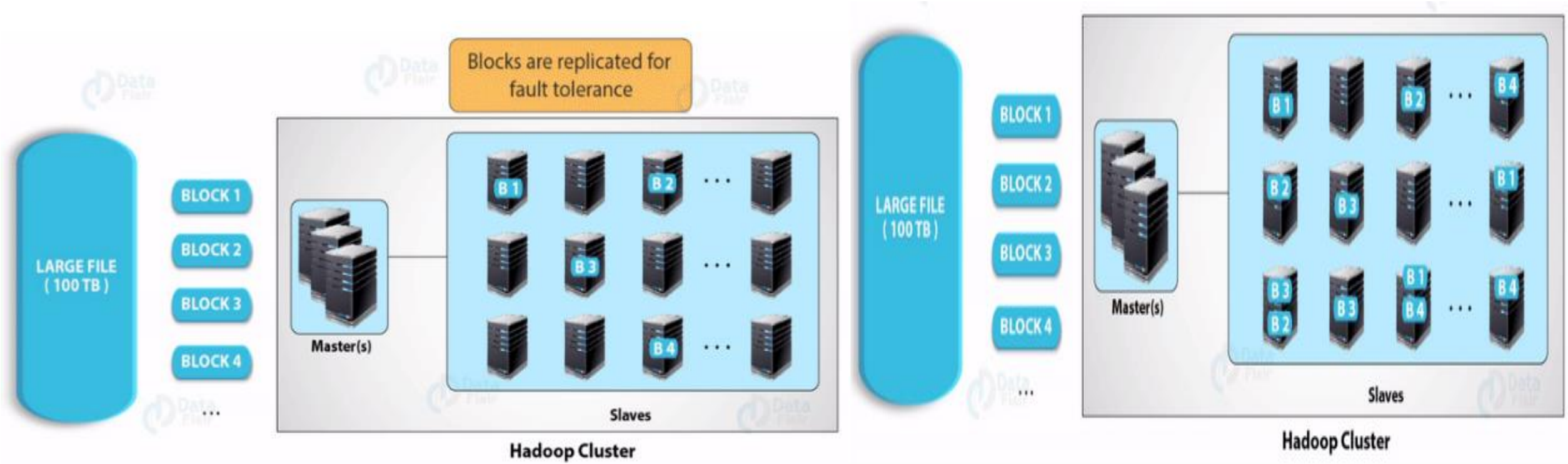
- Short for Hadoop Distributed File System provides for distributed storage for Hadoop. HDFS has a master-slave topology.





Core Components of Hadoop

- Master is a high-end machine where as slaves are inexpensive computers. The Big Data files get divided into the number of **blocks**. Hadoop stores these blocks in a **distributed fashion** on the cluster of **slave nodes**. On the master, we have metadata stored.



Core Components of Hadoop

- HDFS has two daemons running for it. They are : NameNode and DataNode
- **NameNode :** Name Node which is used to store metadata about the Data node, placed with the Master Node. They contain details like the details about the slave note, indexing and their respective locations along with timestamps for timelining.
- NameNode is nothing but the master node. The NameNode is responsible for managing file system namespace, controlling the client's access to files. Also, it executes tasks such as opening, closing and naming files and directories. NameNode has two major files – FSImage and Edits log

Core Components of Hadoop

- **FSImage –** FSImage is a point-in-time snapshot of HDFS's metadata. It contains information like file permission, disk quota, modification timestamp, access time, location of the data on the Data Blocks and which blocks are stored on which node, etc.
- **Edits log –** It contains modifications on FSImage. It records incremental changes like renaming the file, appending data to the file, addition of a new block, replication, deletion etc. In short, it records the changes since the last FsImage was created.
- Whenever the NameNode restarts it applies Edits log to FSImage and the new FSImage gets loaded on the NameNode.

Core Components of Hadoop

- Every time the NameNode restarts, EditLogs are applied to FsImage to get the latest snapshot of the file system. But NameNode restarts are rare in production clusters. Because of this, you may encounter the following issues: .

➢ EditLog grows unwieldy in size, particularly where the NameNode runs for a long period of time without a restart;

➢ NameNode restart takes longer, as too many changes now have to be merged

➢ If the NameNode fails to restart (i.e., crashes), there will be significant data loss, as the FsImage used at the time of the restart is very old

Core Components of Hadoop

- **Secondary NameNode**
- Secondary Namenode helps to overcome the above issues by taking over the responsibility of merging EditLogs with FsImage from the NameNode.
  - The Secondary NameNode obtains the FsImage and EditLogs from the NameNode at regular intervals.
  - Secondary NameNoide loads both the FsImage and EditLogs to main memory and applies each operation from the EditLogs to the FsImage.
  - Once a new FsImage is created, Secondary NameNode copies the image back to the NameNode.
  - Namenode will use the new FsImage for the next restart, thus reducing startup time.

Core Components of Hadoop

- However, this seemingly fail-proof process is not without issues. Delays in the aforesaid process can cause a NameNode to startup without the latest FsImage at its disposal. Such delays can occur if:

➢ The Secondary NameNode takes too long to download the EditLogs from the NameNode;

➢ The NameNode is slow in uploading FsImages to the Secondary NameNode and/or in downloading the updated FsImages from the Secondary NameNode

- To avoid such delays, administrators will have to closely monitor the communication between the NameNode and Secondary NameNode, proactively detect any slowness in the upload and/or download of FsImages / EditLogs, and promptly initiate measures to isolate and remove the source of the slowness. This is where the Hadoop FS Image EditLogs test helps!
- This test monitors the following:

➢ How quickly the Secondary NameNode downloads EditLogs from the NameNode;

➢ How quickly the NameNode uploads and downloads FsImages from the Secondary NameNode

Core Components of Hadoop

- NameNode performs following functions –

- NameNode Daemon runs on the master machine.
- It is responsible for maintaining, monitoring and managing DataNodes.
- It records the metadata of the files like the location of blocks, file size, permission, hierarchy etc.
- Namenode captures all the changes to the metadata like deletion, creation and renaming of the file in edit logs.
- It regularly receives heartbeat and block reports from the DataNodes.

Core Components of Hadoop

- **DataNode:** Data Nodes used for storage of data related to the applications in use placed in the Slave Nodes.
- The various functions of DataNode are as follows –

➢ DataNode runs on the slave machine.

➢ It stores the actual business data.

➢ It serves the read-write request from the user.

➢ DataNode does the ground work of creating, replicating and deleting the blocks on the command of NameNode.

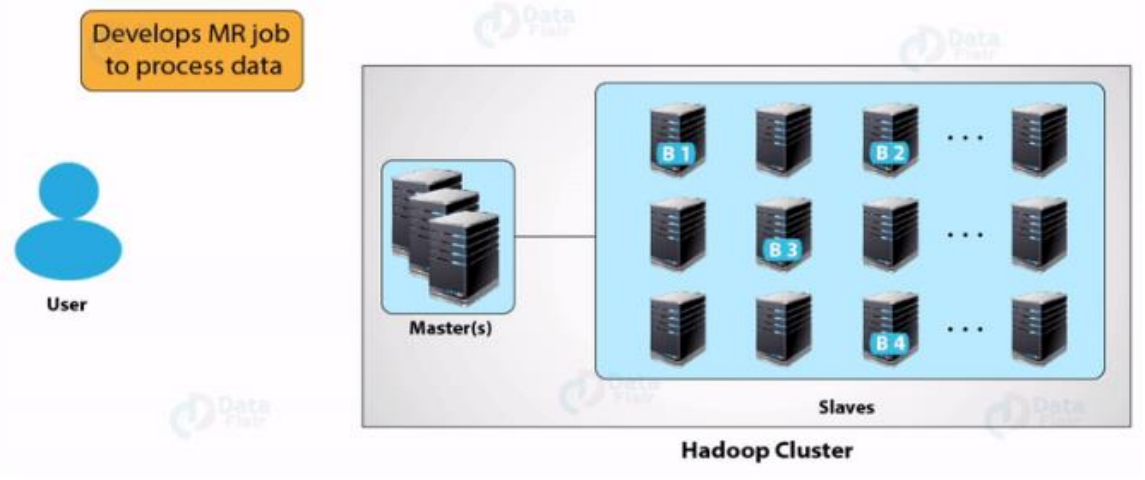
➢ After every 3 seconds, by default, it sends heartbeat to NameNode reporting the health of HDFS.

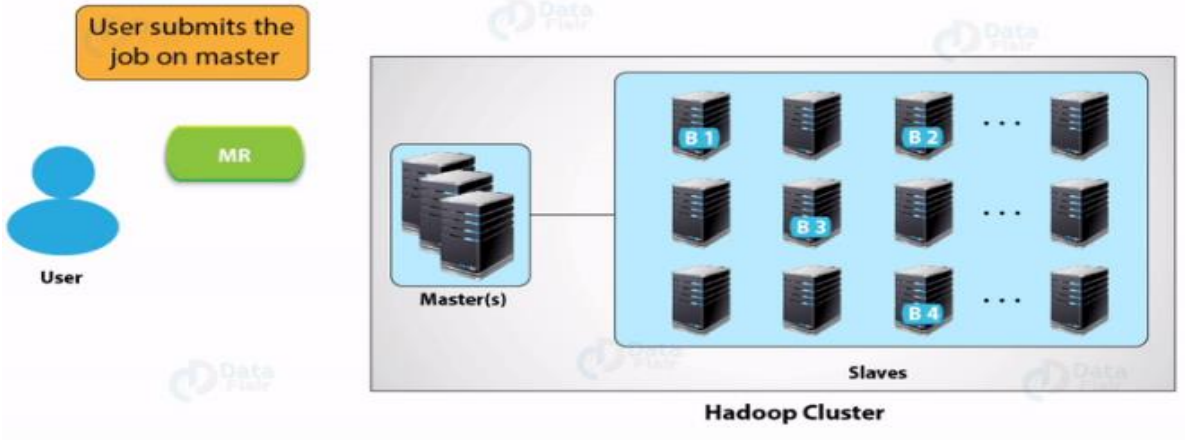
- In other words, a node which knows where the files are to be found in hdfs are Namenode, and the node which have the data of the files are Datanodes.

Core Components of Hadoop

2. MapReduce

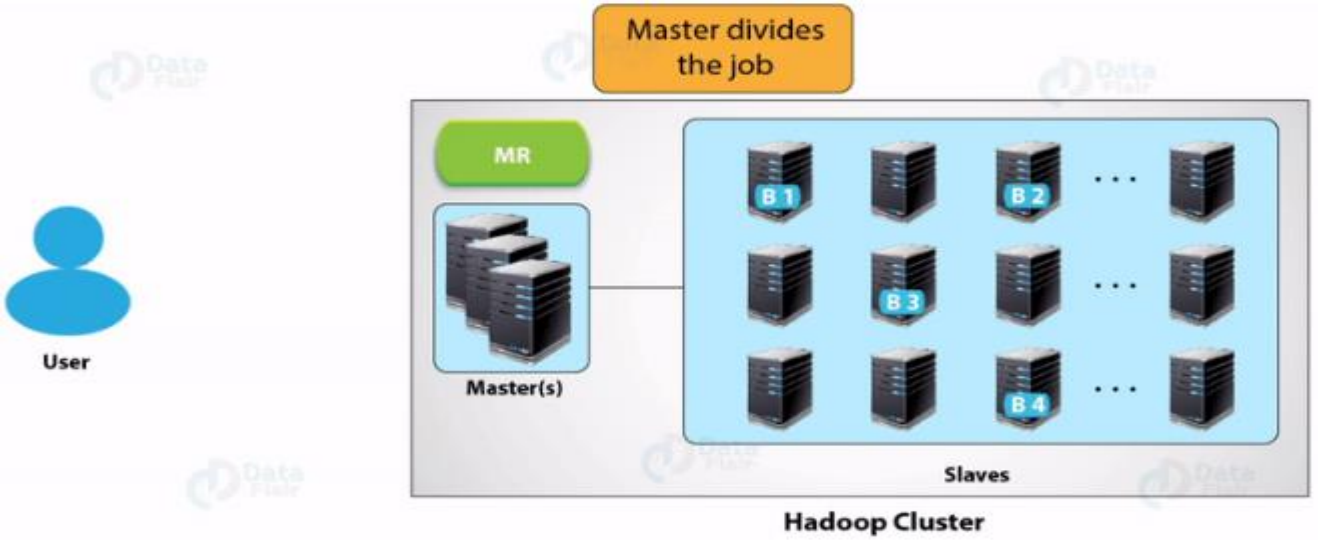
- It is the data processing layer of Hadoop. It processes data in two phases. They are:-
- **Map Phase-** This phase applies business logic to the data. The input data gets converted into key-value pairs.
- **Reduce Phase-** The Reduce phase takes as input the output of Map Phase. It applies aggregation based on the key of the key-value pairs.

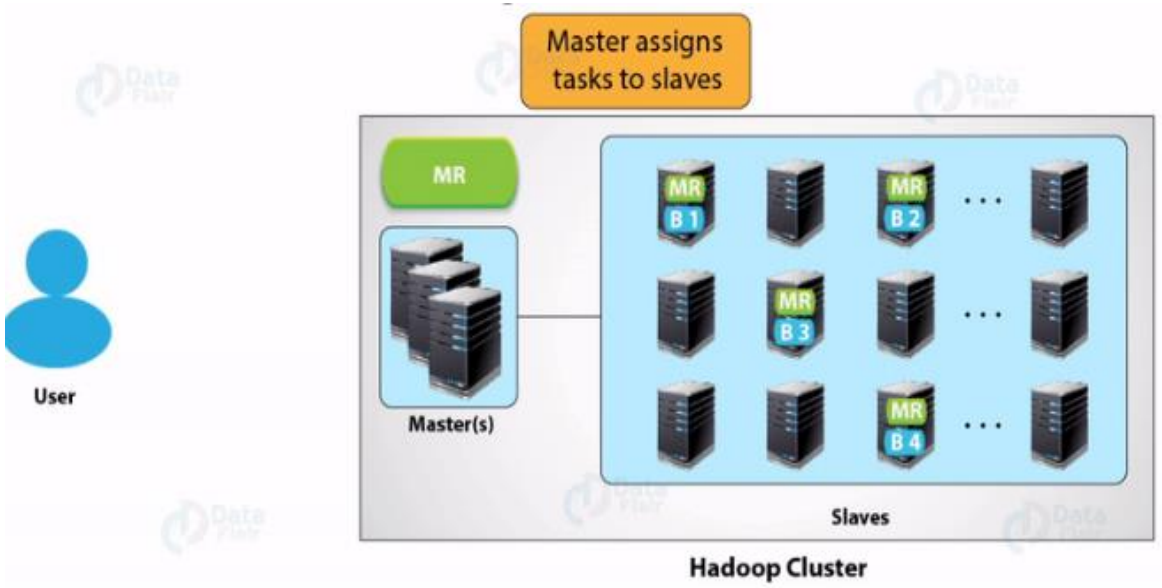




Core Components of Hadoop

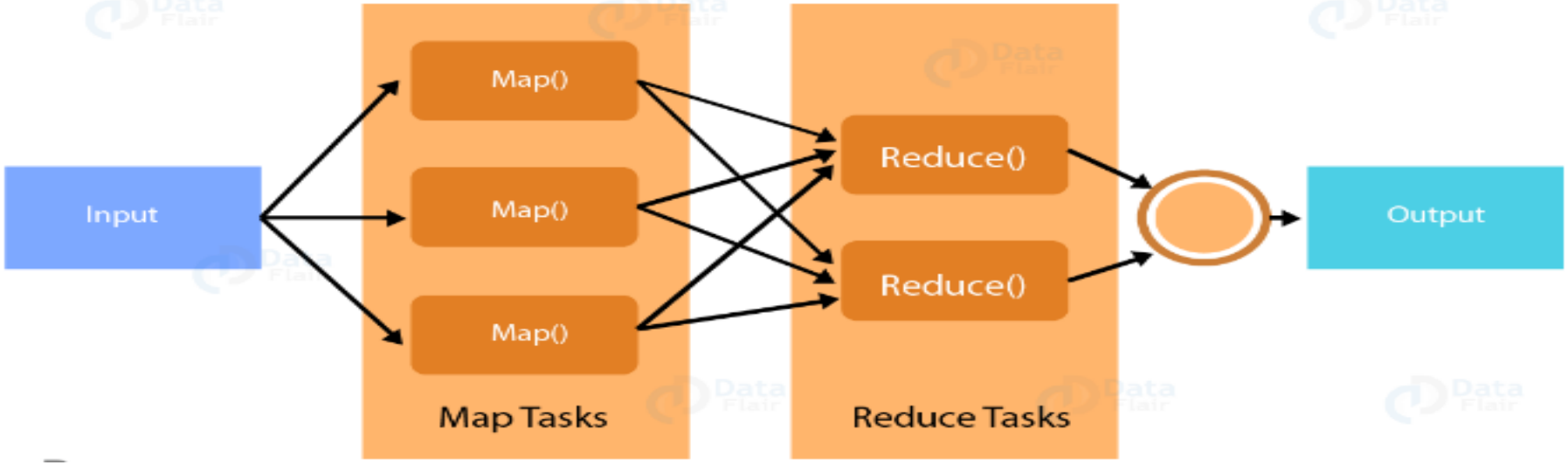
- **MapReduce**





Core Components of Hadoop

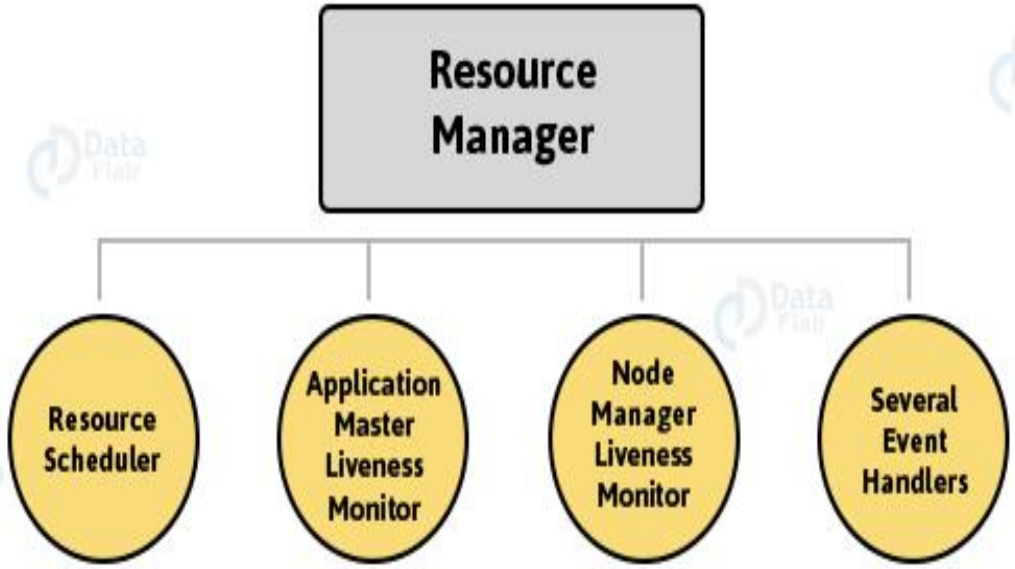
- **MapReduce**



- Mapper reads the block of data and converts it into key-value pairs.
- Now, these key-value pairs are input to the reducer.
- The reducer receives data tuples from multiple mappers.
- Reducer applies aggregation to these tuples based on the key.
- The final output from reducer gets written to HDFS.
- MapReduce framework takes care of the failure. It recovers data from another node in an event where one node goes down.

Core Components of Hadoop

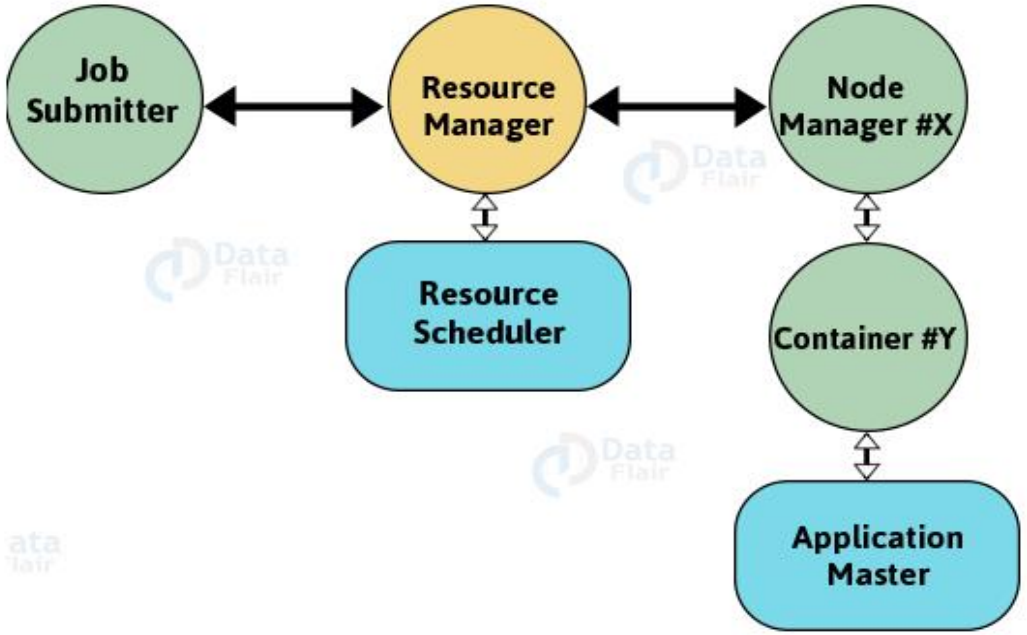
3. YARN (Yet Another Resource Negotiator)



- **Resource Manager** runs on the **master** node.
- It knows where the location of slaves.
- It is aware about how much resources each slave have.
- **Resource Scheduler** and **Application Manager** is one of the important **service** run by the **Resource Manager**.
- Resource Scheduler decides how the resources get assigned to various tasks.
- Application Manager negotiates the first container for an application.
- Resource Manager keeps track of the heart beats from the Node Manager.

Core Components of Hadoop

3. YARN (Yet Another Resource Negotiator)



- The application start up process is as follows:-

➢ The client submits the **job** to **Resource Manager**.

➢ Resource Manager contacts **Resource Scheduler** and allocates **container**.

➢ Now Resource Manager contacts the relevant **Node Manager** to **launch** the **container**.

➢ Container runs Application Master.

Core Components of Hadoop

3. YARN (Yet Another Resource Negotiator)

- The basic idea of YARN was to **split the task** of resource management and job scheduling. It has one global Resource Manager and per-application Application Master. An application can be either one job or DAG of jobs.
- The Resource Manager's job is to assign resources to various competing applications. **Node Manager** runs on the **slave** nodes. It is responsible for **containers, monitoring resource utilization and informing** about the same to Resource Manager.
- The job of Application master is to negotiate resources from the Resource Manager. It also works with NodeManager to execute and monitor the tasks.

Core Components of Hadoop

- Daemons are the processes that run in the background. The Hadoop Daemons are:-

  a) Namenode – It runs on master node for HDFS.

  b) Datanode – It runs on slave nodes for HDFS.

  c) Resource Manager – It runs on YARN master node for MapReduce.

  d) Node Manager – It runs on YARN slave node for MapReduce.

These 4 daemons run for Hadoop to be functional.

How Hadoop Works?

- Summarize how Hadoop works step by step:

➢ Input data is broken into blocks of size 128 Mb and then blocks are moved to different nodes.

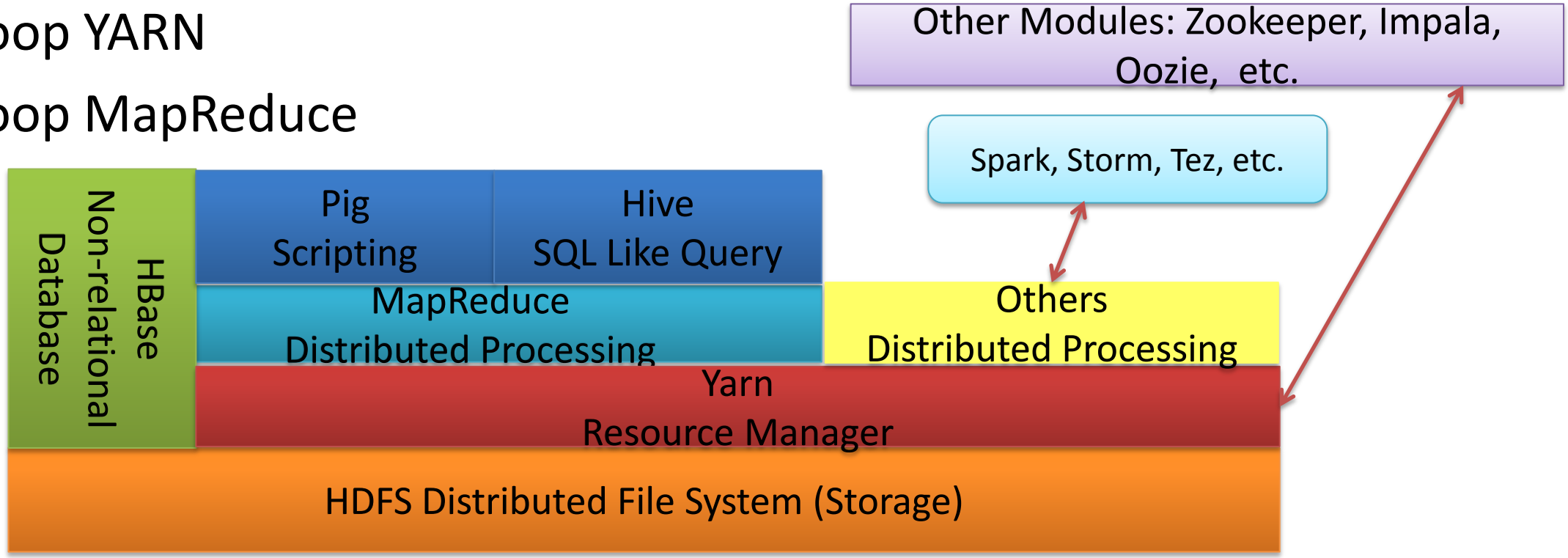
➢ Once all the blocks of the data are stored on data-nodes, the user can process the data.

➢ Resource Manager then schedules the program (submitted by the user) on individual nodes.

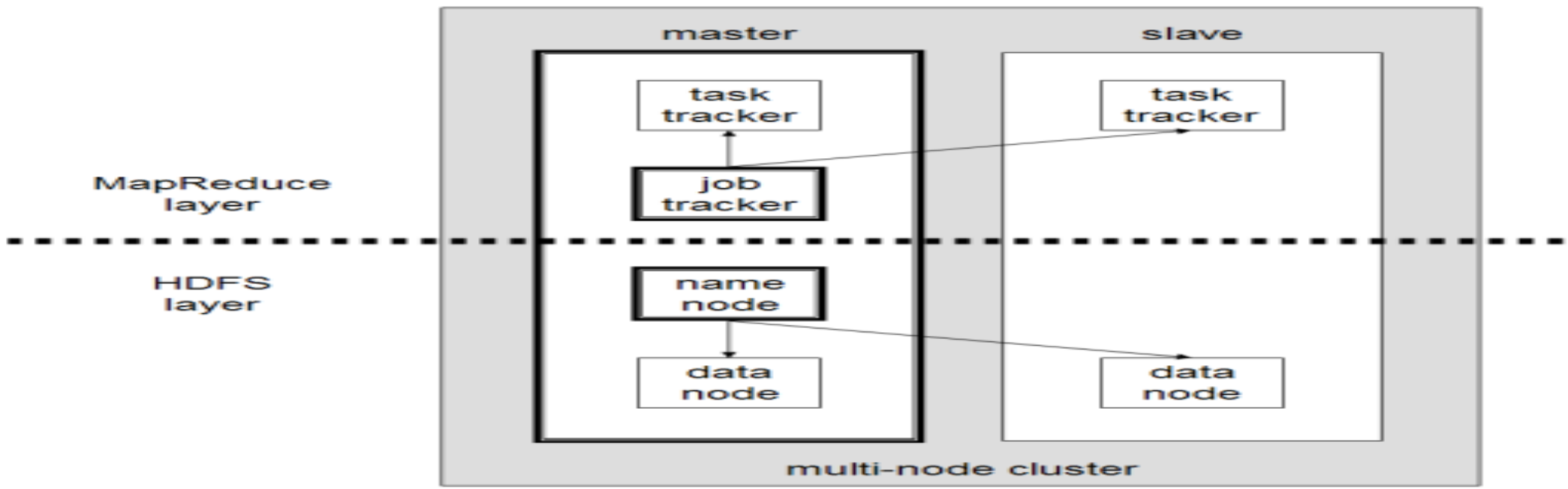
➢ Once all the nodes process the data, the output is written back to HDFS.

Apache Hadoop Basic Modules

- Hadoop Common
- Hadoop Distributed File System (HDFS)
- Hadoop YARN
- Hadoop MapReduce



HADOOP MASTER/SLAVE ARCHITECTURE



Hadoop Core Components:
HDFS
MapReduce
YARN (MapReduce 2.0)

Hadoop Essential
Pig (Pig Latin + Pig Runtime) Hive or HQL, HBase (Google's Big table), Cassandra (Amazon's Dynamo) Avro, Zookeeper, Sqoop, Mahout (machine Learning) ...

Hadoop Incubator
Chukwa, Ambari, HDT Hcatalog, Knox, Spark, Kafka (Data Ingestion layer) Storm (Data analytic layer) Samza, Hama, Nutch ...

Training Data Sets | Algorithms | Dashboard | Administration

Platform ETL Tools | Platform Functions | Platform Core Units | Data Processing Modules | Platform Admin Modules

hadoop 3 Hadoop Framework, V1 and V2
Map Reduce Hadoop File Distributed System (HFDS)
YARN (Map Reduce 2.0)
Apache

Sqoop & Hiho (Structured Data)
Flume & Scribe (Unstructured data)
Chukwa (log)
Pentaho, Talend ETL
Knox Security, AAA
1

Data Integration

Map Reduce programming — hadoop MapReduce — 2 — Pig, Hive, Java, Perl

Data Storage, Data Library, Meta Data Store — 4 — Parquet HBase, Cassandra DataFu, Whirr (cloud), Sentry, Nutch, Solr, Gora MongoDB

Data Interaction Visualization Execution Development — **Hcatalog,** Lucene Hama, Crunch

Data Serialization — 4a — Avro, Thrift

Data Intelligence — 4b — Drill (Dremel), **Mahout**

Data Analysis with SQL — 4c — Impala, **Spark**, Hive

MAHOUT · Spark · H_2O · Flink

Oozie (workflow)
Zookeeper (configure)
Ambari, CDH (provisioning)
Hue (User Experiences)
Nagios, Ganglia Monitoring
Splunk, Talend Report
HDT, development

Management, Monitoring, Orchestration
5

Operation Systems (Windows, Linux)

Virtualized Infrastructure (Virtual Machine, J2EE container)

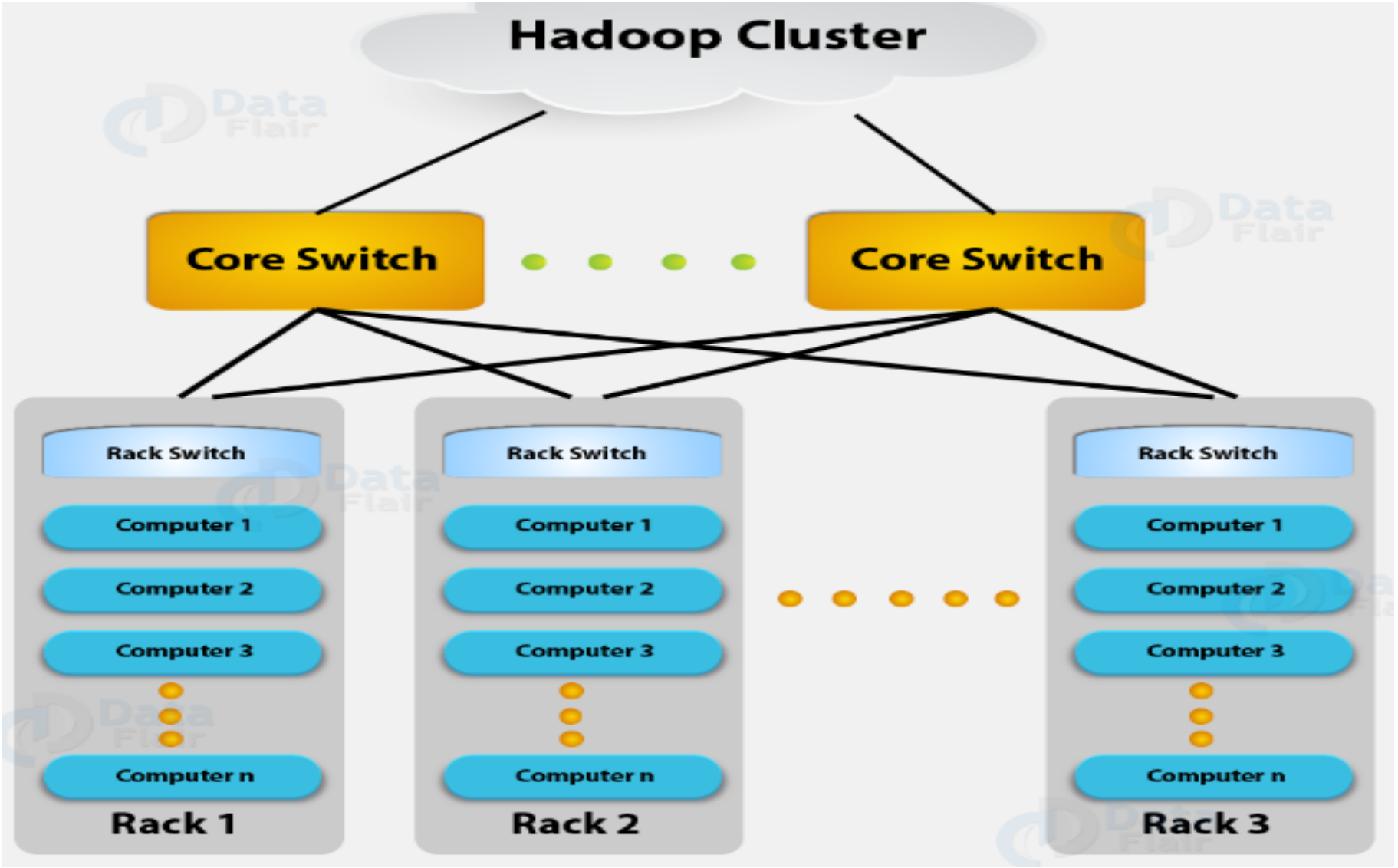
Hadoop Cluster

- Cluster is a set of connected computers which work together as a single system. Similarly, the Hadoop cluster is just a computer cluster which we use for Handling huge volume of data distributedly.
- **Hadoop Cluster Architecture**
- Basically, for the purpose of **storing** as well as **analyzing huge amounts of unstructured data** in a **distributed computing environment**, a special type of computational cluster is designed that what we call as Hadoop Clusters.
- Hadoop Clusters: two main terms come up, they are cluster and node, so on defining them:

➢ A collection of nodes is what we call the cluster.

➢ A node is a point of intersection/connection within a network, ie a server

Hadoop Cluster Architecture



Hadoop Cluster

- There is nothing shared between the nodes in a Hadoop cluster except for the network which connects them (Hadoop follows shared-nothing architecture). This feature **decreases** the **processing latency** so the cluster-wide latency is minimized when there is a need to process queries on huge amounts of data.
- In addition, Hadoop clusters have two types of machines, such as Master and Slave, where:

➢ **Master: HDFS NameNode, YARN ResourceManager.**

➢ **Slaves: HDFS DataNodes, YARN NodeManagers**.

Hadoop Cluster

- However, it is recommended to separate the master and slave node, because:

➢ Task/application workloads on the slave nodes should be isolated from the masters.

➢ Slaves nodes are frequently decommissioned for maintenance.

- Moreover, it is possible to scale out a Hadoop cluster. Here, Scaling means to add more nodes. That's why we also call it **linearly scalable**. Hence, we get a corresponding boost in throughput, for every node we add.

Hadoop Cluster

- **The Communication Protocols**

- For inter-node communication, Hadoop uses **tcp ip**. Basically, on top of the TCP/IP protocol, all HDFS communication protocols are layered and on the NameNode machine, a client establishes a connection to a configurable TCP port. Moreover, with the NameNode it talks the ClientProtocol. And, by using the DataNode Protocol, the DataNodes talk to the NameNode.
- In addition, the abstraction Remote Procedure Call (RPC) wraps both the DataNode Protocol as well as the Client Protocol. Although, the NameNode never starts any RPCs, by design. Rather than that it only responds to RPC requests which are issued by DataNodes or clients.

Hadoop Cluster

- **Hadoop Nodes Configuration**

➢ However, by two types of important configuration files, Hadoop's Java configuration is driven:

- Read-only default configuration : core-default.xml, hdfs-default.xml, yarn-default.xml and mapred-default.xml.
- Site-specific configuration : etc/hadoop/core-site.xml, etc/hadoop/hdfs-site.xml, etc/hadoop/yarn-site.xml and etc/hadoop/mapred-site.xml.

➢ Moreover, by setting site-specific values via above files and etc/hadoop/hadoop-env.sh and etc/hadoop/yarn-env.sh, we can control the Hadoop scripts found in the bin/ directory of the distribution.

Hadoop Cluster

- **Advantages of a Hadoop Cluster**

- The cluster helps in increasing the speed of the analysis process.
- It is inexpensive.
- These clusters are failure resilient.
- One more benefit to Hadoop clusters is “scalability”, it means Hadoop offers Scalable and flexible Data Storage. Here Scalability means, we can scale a Hadoop cluster by adding new servers to the cluster if needed.
- Hadoop Clusters deal with data from many sources and formats in a very quick, easy manner.
- It is possible to deploy Hadoop using a single-node installation, for evaluation purposes.

Hadoop Features

- **Reliability**

➢ In the Hadoop cluster, if any node goes down, it will not disable the whole cluster. Instead, another node will take the place of the failed node. Hadoop cluster will continue functioning as nothing has happened. Hadoop has built-in fault tolerance feature.

- **Scalable**

➢ Hadoop gets integrated with cloud-based service. If you are installing Hadoop on the cloud you need not worry about scalability. You can easily procure more hardware and expand your Hadoop cluster within minutes.

Hadoop Features

- **Economical**

➢ Hadoop gets deployed on commodity hardware which is cheap machines. This makes Hadoop very economical. Also as Hadoop is an open system software there is no cost of license too.

- **Distributed Processing**

➢ In Hadoop, any job submitted by the client gets divided into the number of sub-tasks. These sub-tasks are independent of each other. Hence they execute in parallel giving high throughput.

Hadoop Features

- **Distributed Storage**

➢ Hadoop splits each file into the number of blocks. These blocks get stored distributedly on the cluster of machines.

- **Fault Tolerance**

➢ Hadoop replicates every block of file many times depending on the replication factor. Replication factor is 3 by default. In Hadoop suppose any node goes down then the data on that node gets recovered. This is because this copy of the data would be available on other nodes due to replication. Hadoop is fault tolerant.

Thank you.