



Introduction to Hadoop and MapReduce Programming - Session 1

Course: Data Engineering - I

Lecture On: Introduction to
Hadoop

Instructor: Vishwa Mohan



Segment - 01

Module Introduction

Module Introduction

Session 1

- Introduction to distributed systems
- GFS and MapReduce
- Introduction to Hadoop and components of HDFS
- Hadoop 2.x
- YARN and task processing
- Tools for Hadoop

Session 2

- File storage in HDFS
- Demonstration of basic commands in HDFS
- Write operation and rack awareness
- Read operation in HDFS
- Features and limitations of HDFS

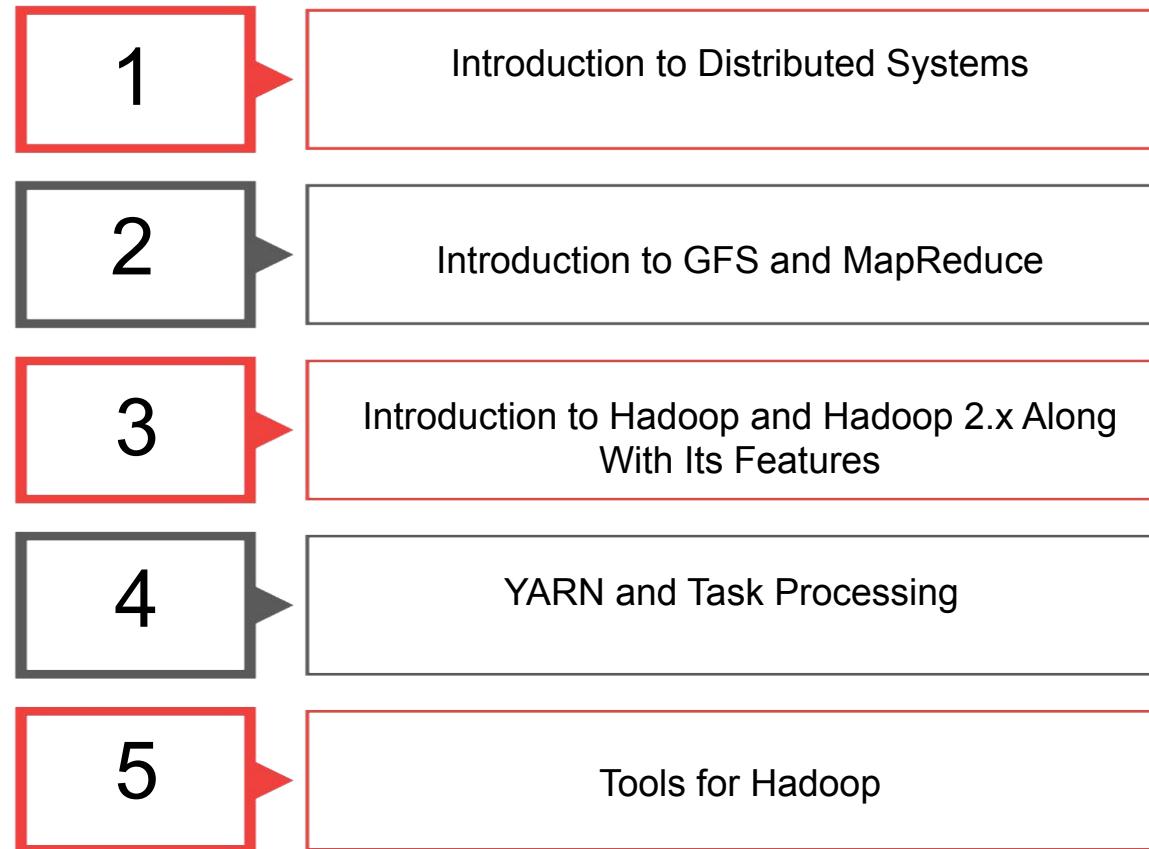
Session 3

- Introduction to the MapReduce Framework
- Basic implementation of MapReduce using Python
- Hadoop streaming with demonstration
- The Combiner
- The Partitioner
- Job scheduling and fault tolerance in MapReduce

Segment - 02

Introduction: Introduction to Hadoop

Session Overview



Segment - 03

Introduction to Distributed Systems

1

Introduction to distributed systems

2

The need for distributed systems and their features

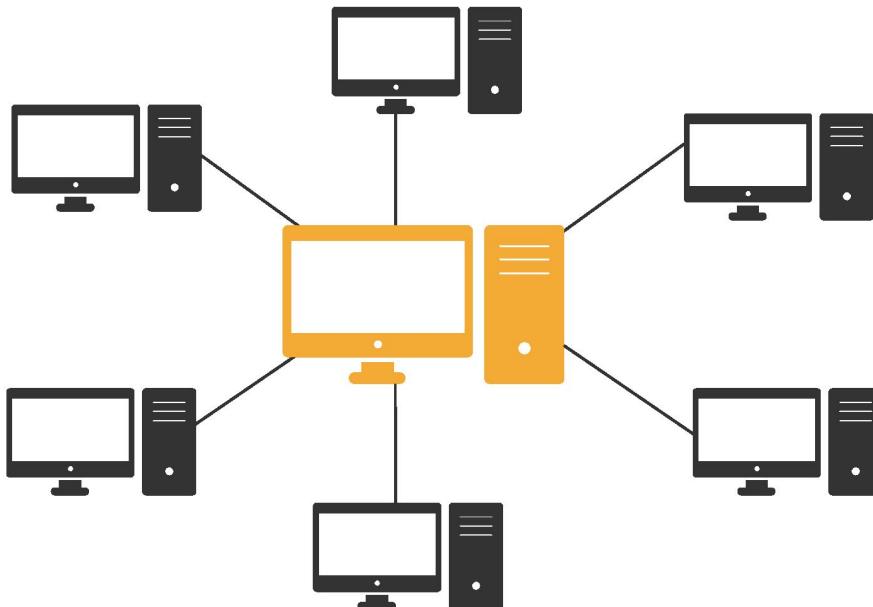
Why are distributed systems needed?



COFFEE
- house -

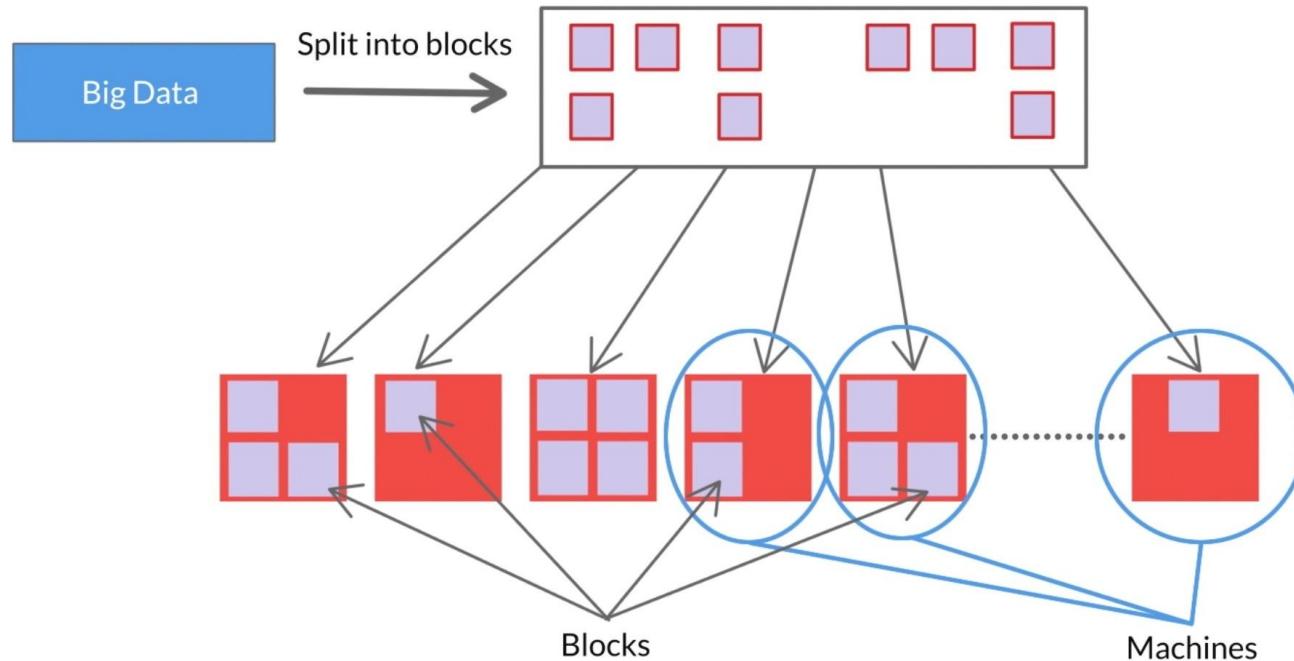
- A single store of Coffee-house may take 100s of order each day
- If suppose there are 10 stores in a small city, data grows more or less proportionately.
- At the global level, if Coffee-house has 10000 stores across the world, the amount of data generated and processed can't be handled by a single machine.

Introduction to Distributed Systems



A **distributed system** is a system in which several independent computers are connected to each other over a network with middleware to look like a single machine.

DISTRIBUTED COMPUTING



Primary Challenges faced by Distributed Systems

- **Performance** - Also includes challenges like communication fault delay or computation fault delay.
- **Fault tolerance** - Tolerate faults and function normally.
- **Scalability** - System must remain effective even under heavy load.
- **Security** - Probable threats like information leakage, integrity violation, denial of services and illegitimate use
- **Concurrency** - Shared access to resources must be made available to the required processes.
- **Migration** - Tasks should be allowed to move within the system without affecting other operations.
- **Load balancing** - Load must be distributed among available resources for better performance.

1

Introduced to the concept of distributed systems

2

Discussed why they are needed and what their features are

Segment - 04

Introduction to GFS and MapReduce

1

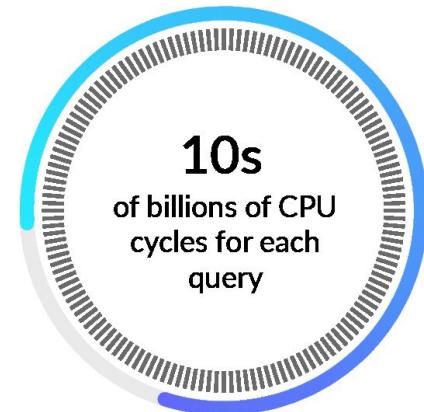
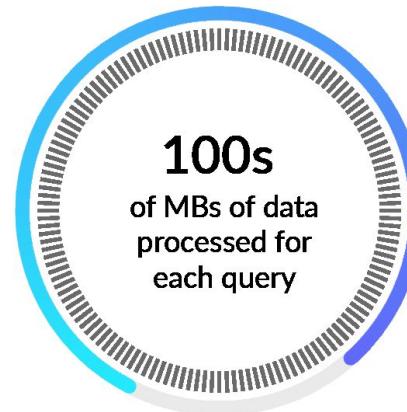
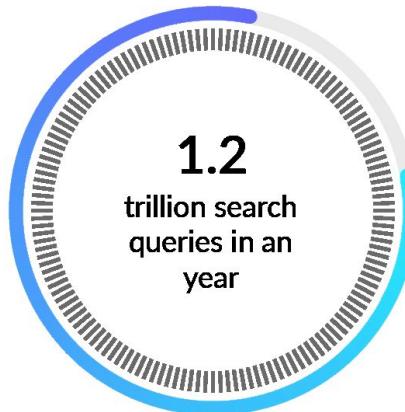
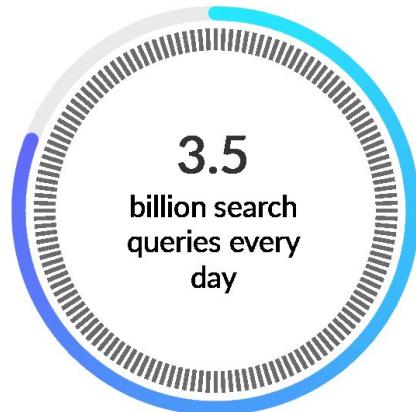
GFS and its master/slave
architecture

2

Considerations for GFS and the
origin of MapReduce

Introduction to GFS and MapReduce

Today, on average, Google handles...



Introduction to GFS and MapReduce



Google File System (GFS)

GFS is a proprietary distributed file system made by Google for storing and processing large amounts of data in multiple commodity machines in large clusters, which ensures reliability and scalability and also maintains efficiency.

Considerations made while developing GFS and their effects

1 Uses commodity hardware

2 Can be easily scaled horizontally

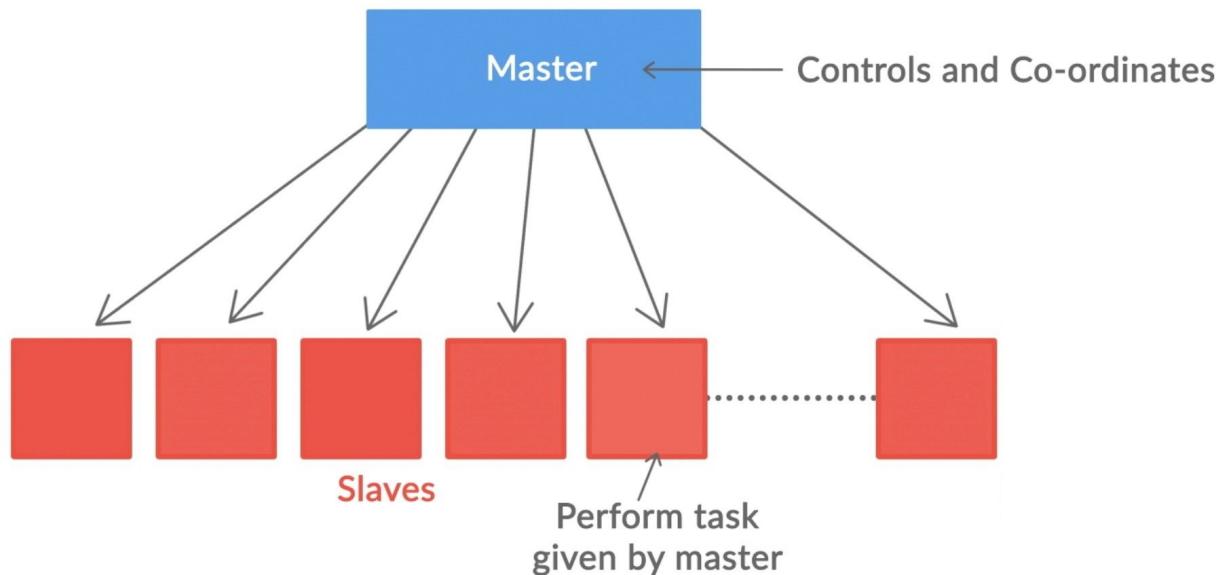
3 Commodity hardware always fails

4 Fault tolerance and automatic
error recovery are crucial

How GFS solved the problems of traditional Distributed systems?

- Very **high availability** and **fault tolerance** maintained through **replication**
- **Automatic and efficient data recovery**
- **High aggregate throughput**
- **Data integrity** is ensured as each chunkserver verifies integrity of their own copies using checksums
- Each chunk server is constantly monitored by GFS master
- **Modularity** allows GFS to easily expand to account for increased loads
- Master Node ensures **load balancing** is maintained

MASTER - SLAVE LAYOUT



Introduction to GFS and MapReduce

Google in their paper on “**MapReduce: Simplified Data Processing on Large Clusters**” first introduced MapReduce, a programming model and an associated implementation for processing & generating large data sets



MapReduce was then used in the **Apache Nutch project**. It was highly scalable and supported unstructured data. MapReduce jobs were able to withstand hardware failures, thus ensuring fault-tolerance.



Introduction to GFS and MapReduce

Google implemented GFS in their white paper titled “**The Google File System**” and defined it as a scalable distributed file system for large distributed data-intensive applications

Later on, Development of **Hadoop** started as a sub-project of **Apache Nutch** when the Apache community realised that the implementation of MapReduce and Nutch DFS could be used for other tasks as well.

1

Discussed GFS and its
master/slave architecture

2

Discussed the considerations
made while using GFS and the
origin of MapReduce

Segment - 05

Introduction to Hadoop

1

Introduction to Hadoop and its components

2

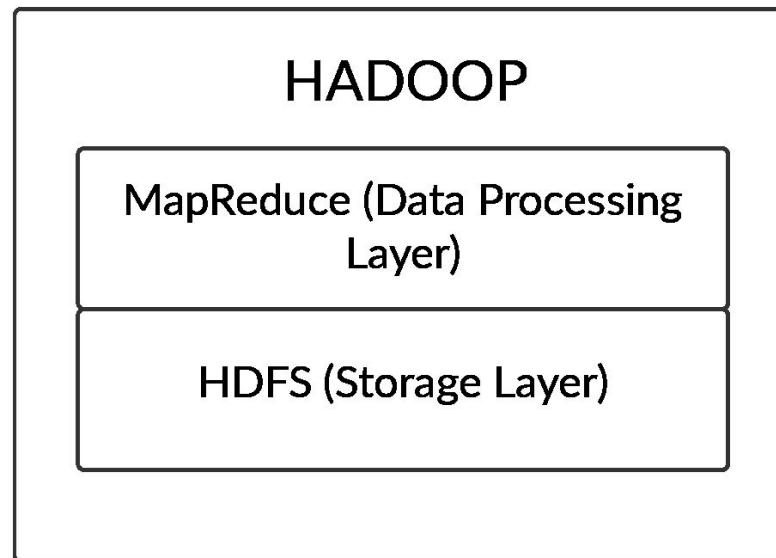
HDFS, its components and basic architecture

Introduction to Hadoop

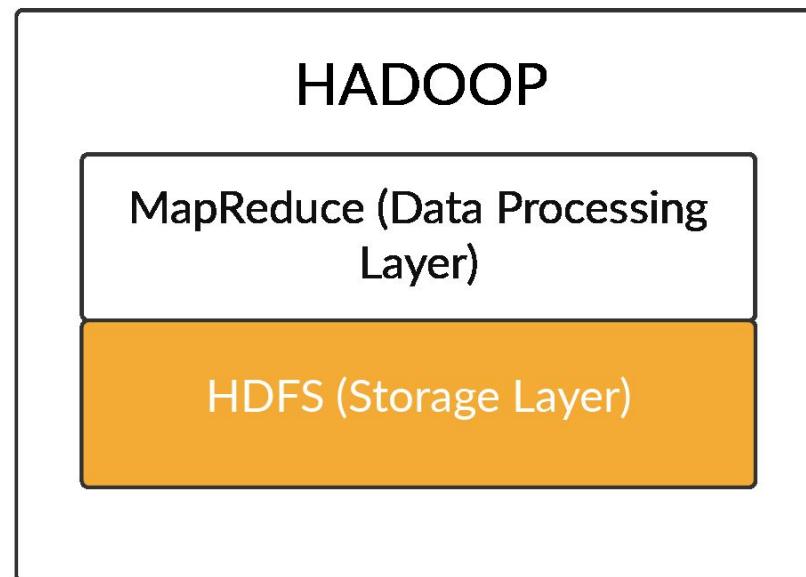


Hadoop is an open source framework that is used for storing and processing big data on the clusters of commodity hardware.

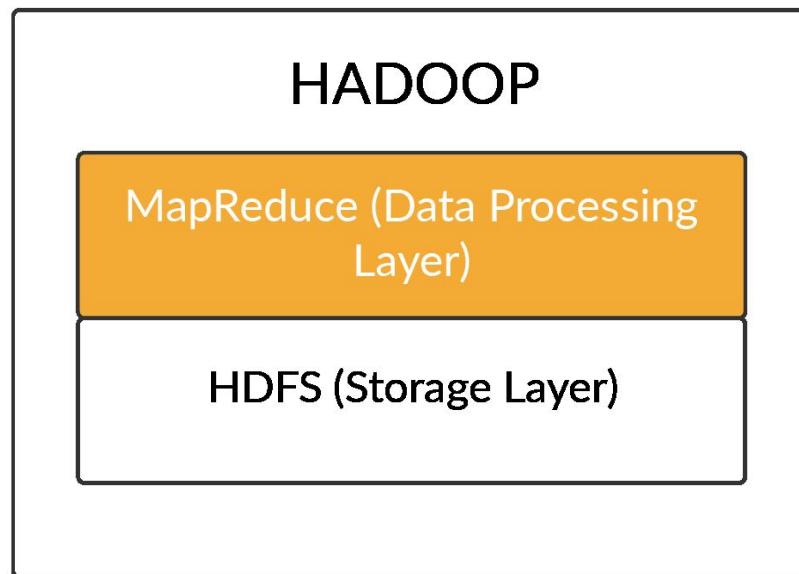
Hadoop contains two main components: The
Hadoop Distributed File System (HDFS) and
MapReduce



Hadoop contains two main components: The
Hadoop Distributed File System (HDFS) and
MapReduce

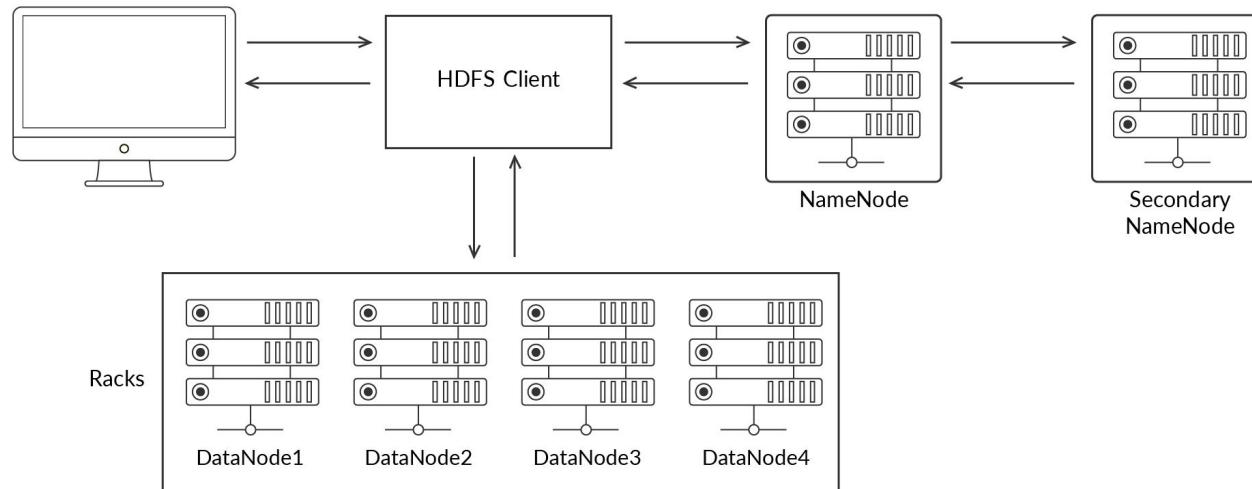


Hadoop contains two main components: The
Hadoop Distributed File System (HDFS) and
MapReduce



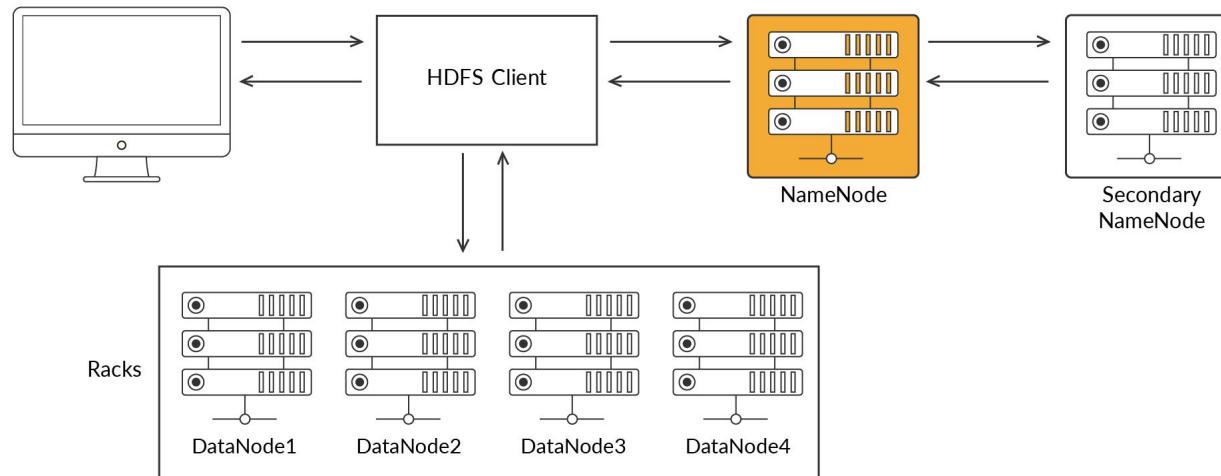
Introduction to Hadoop

HDFS has three main components: The NameNode, the Secondary NameNode and DataNodes



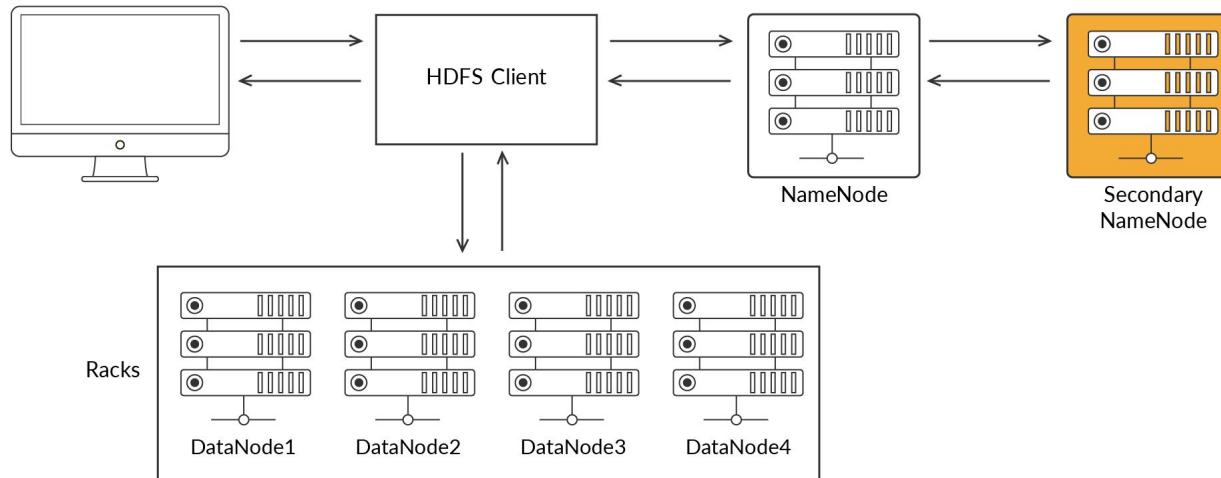
Introduction to Hadoop

HDFS has three main components: The NameNode, the Secondary NameNode and DataNodes



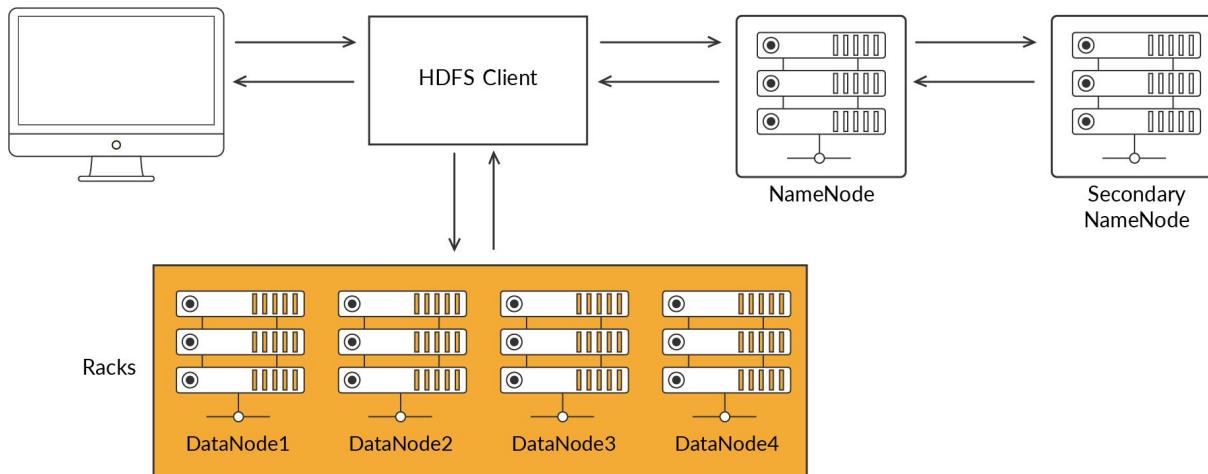
Introduction to Hadoop

HDFS has three main components: The NameNode, the Secondary NameNode and DataNodes



Introduction to Hadoop

HDFS has three main components: The NameNode,
the Secondary NameNode and DataNodes



1

Introduced to Hadoop and its components

2

Discussed HDFS, its components and architecture

Segment - 06

Hadoop 2.x

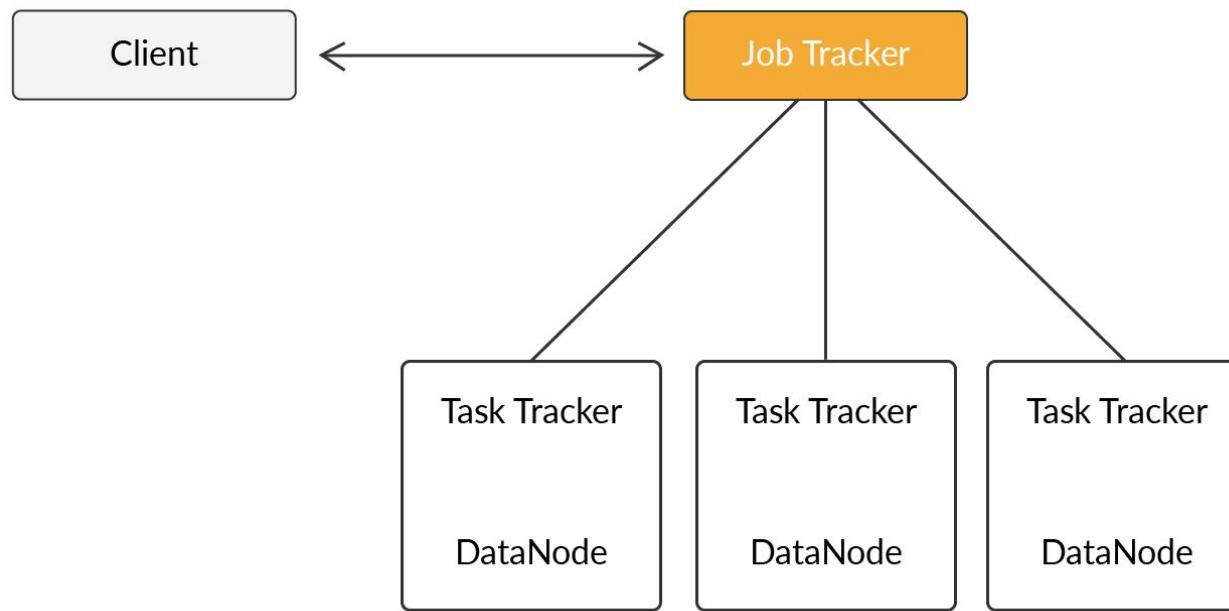
1

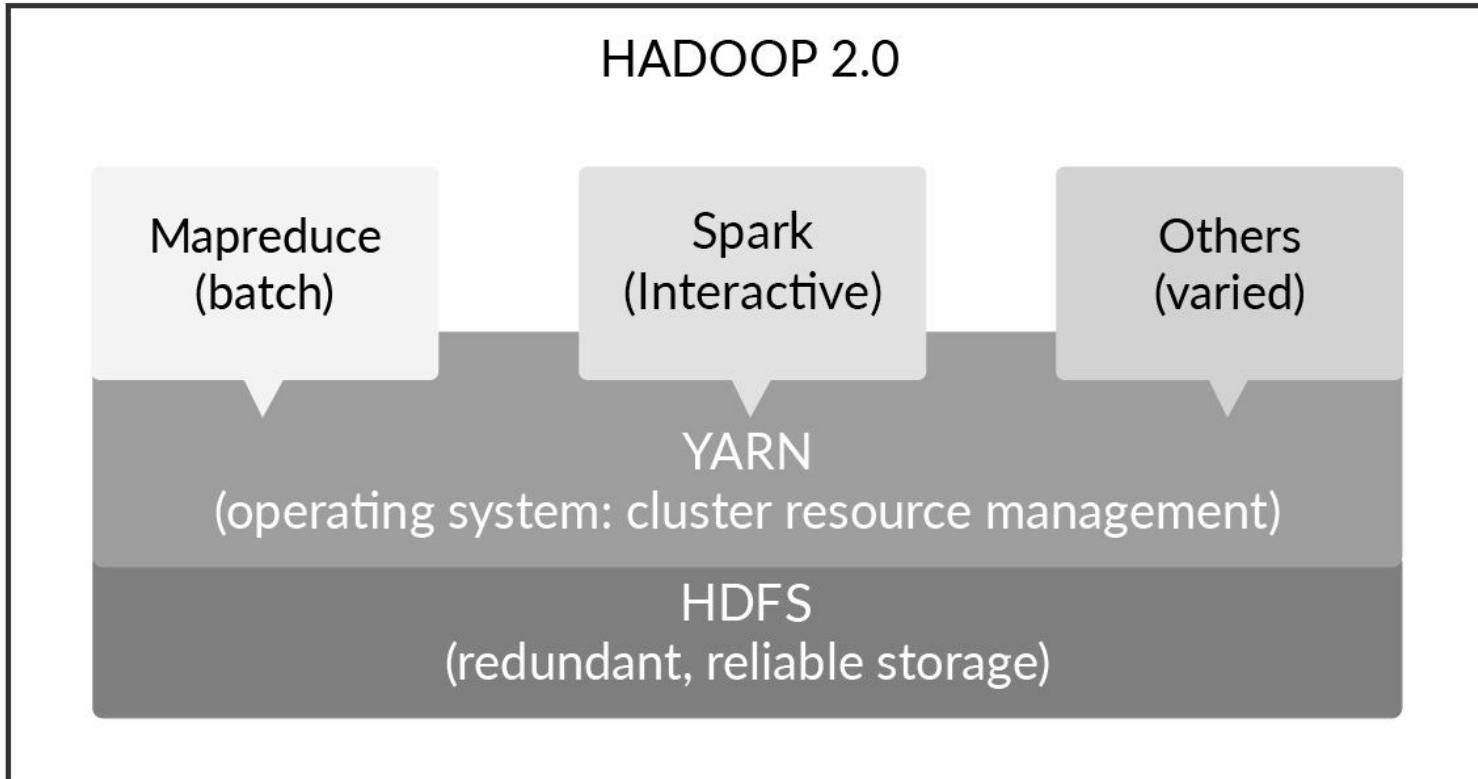
Motivation behind the development of Hadoop 2.x

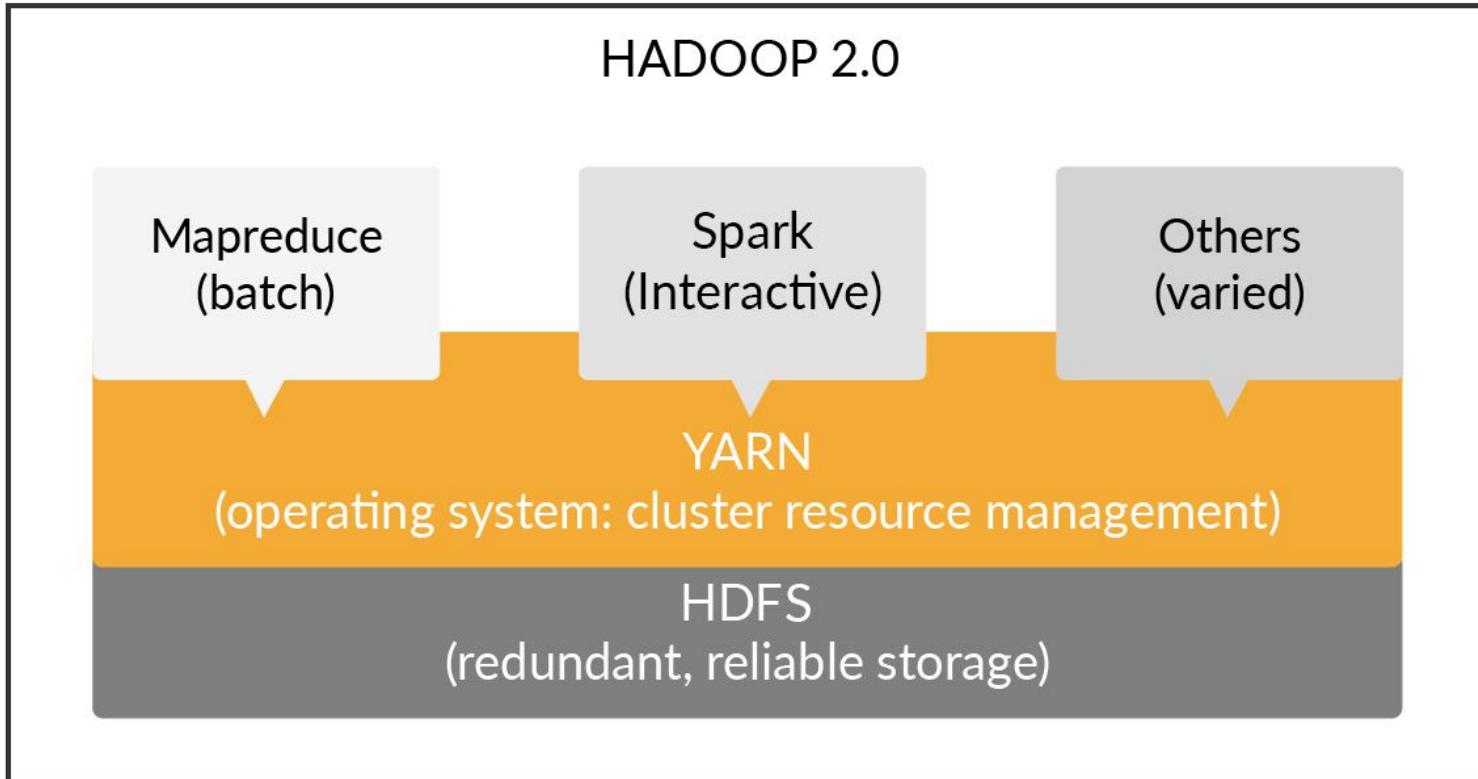
2

Introduction to Hadoop 2.x and its improvements

Main motivation for developing Hadoop 2.x









Improvements in Hadoop 3.x

- The runtime version requirement of Java 8
- Erasure coding support
- Improved YARN Timeline Service v.2
- Support for more than two NameNodes
- Support for GPUs in YARN
- Intra-node disk balancing.

1

Discussed the motivation for
Hadoop 2.x

2

Introduced Hadoop 2.x and its
improvements

Segment - 07

YARN

1

Introduction to YARN

2

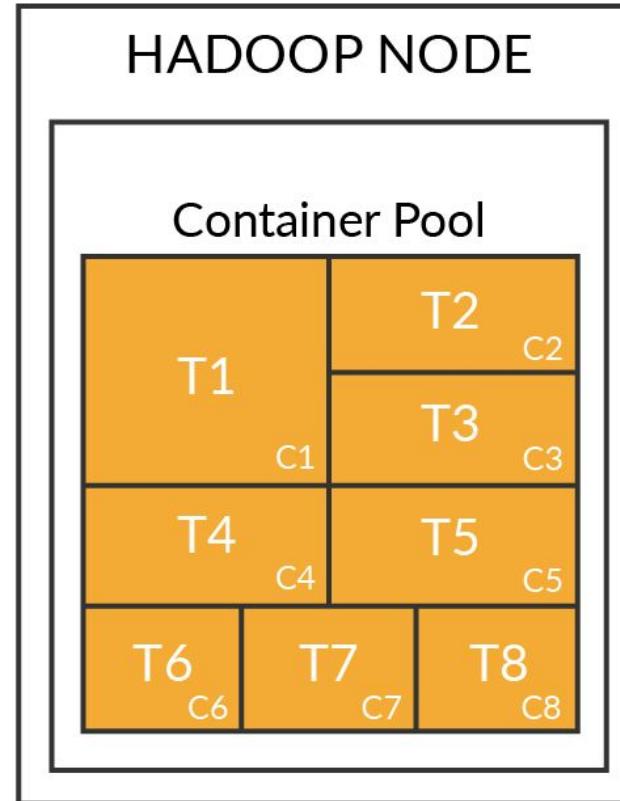
Components of YARN

YARN was introduced in Hadoop 2.x mainly to split the responsibilities of the JobTracker in Hadoop 1.x into separate components, thus improving the scalability and reliability of the whole system.

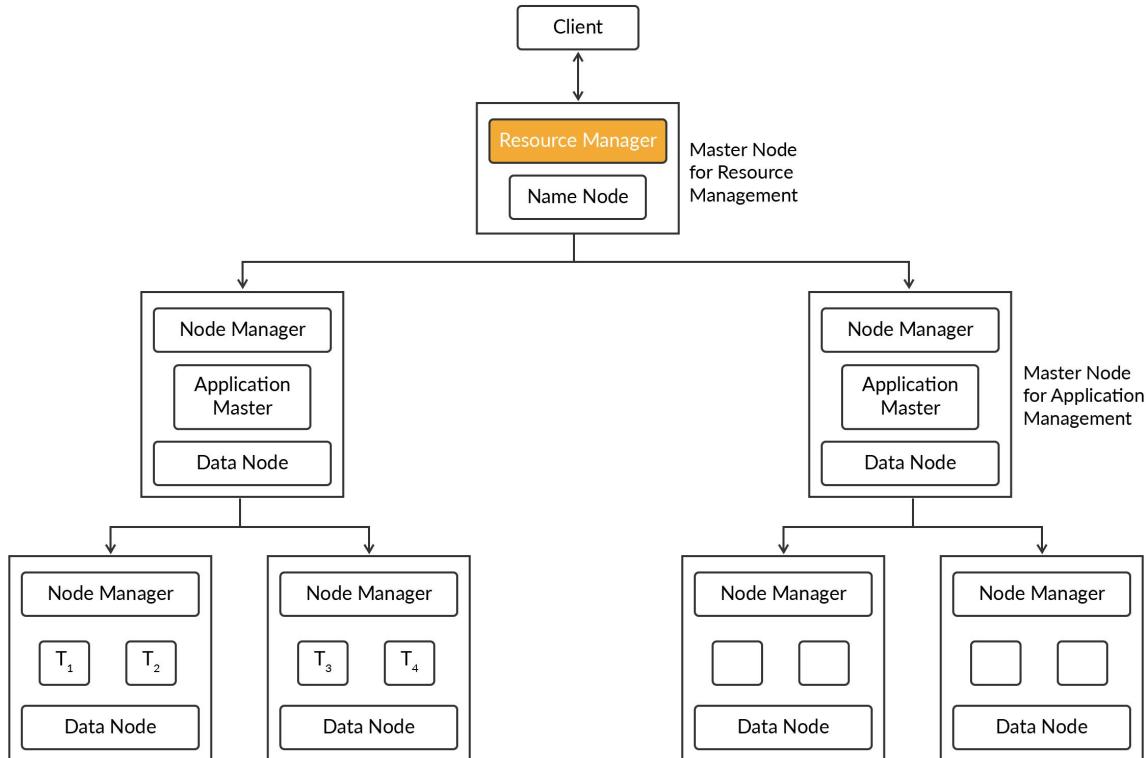
Components of YARN

- 1 Resource Manager
- 2 Node Manager
- 3 Application Master
- 4 Containers

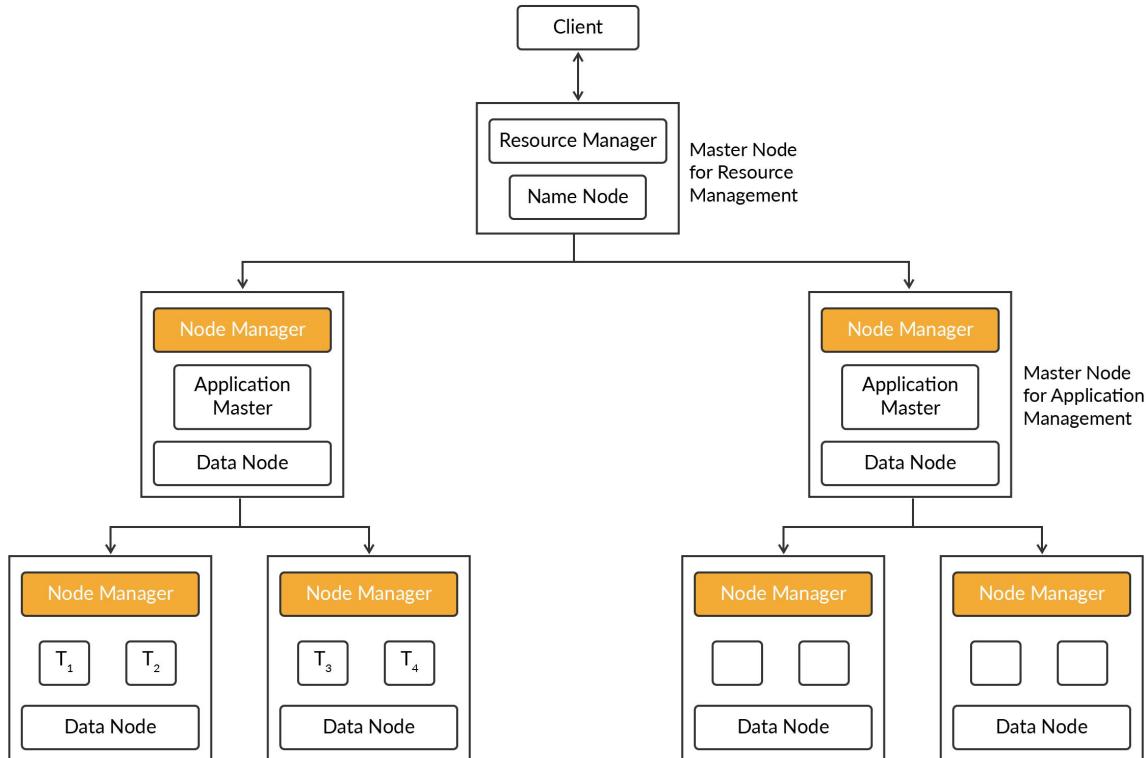
Containers



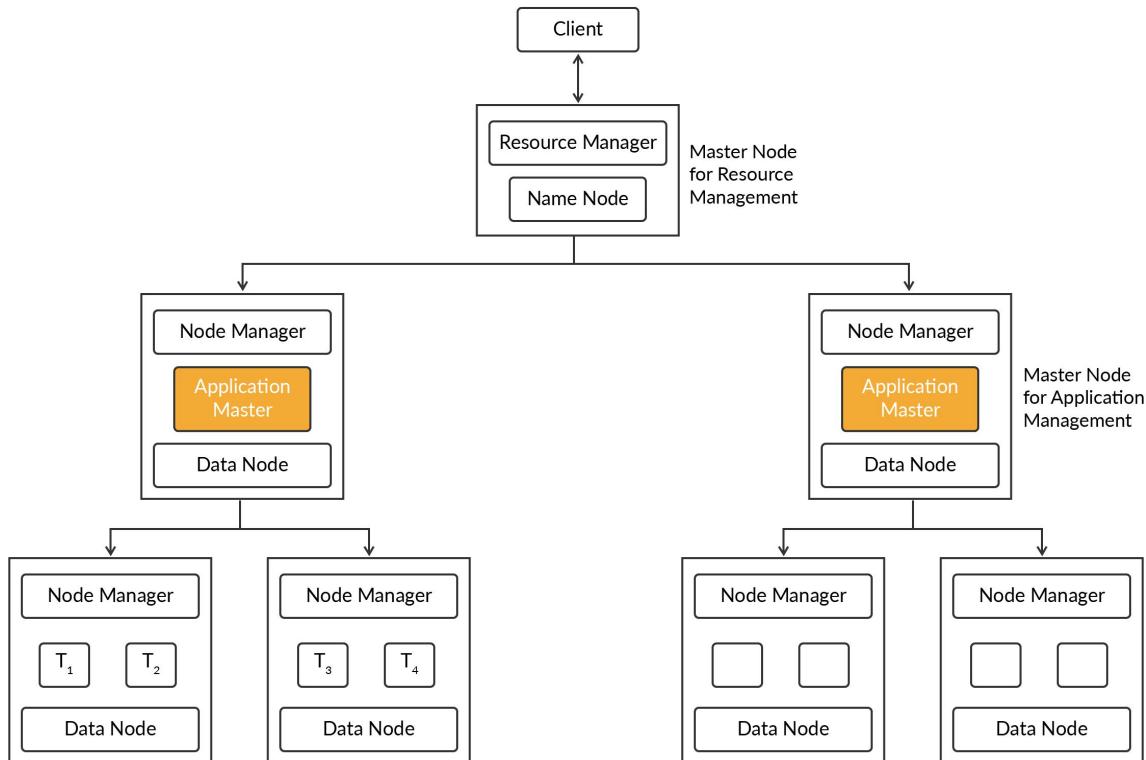
Resource Manager



Node Manager



Application Master



1

Introduced YARN

2

Discussed the different
components of YARN

Segment - 08

Task Processing in Hadoop

1

A recap of YARN components
and their functions

2

Task processing in Hadoop

A quick recap of the components of YARN

Resource Manager

- One per cluster
- Runs in the master node

Application Master

- Installed in slave nodes
- Monitors jobs and requests containers from the Scheduler

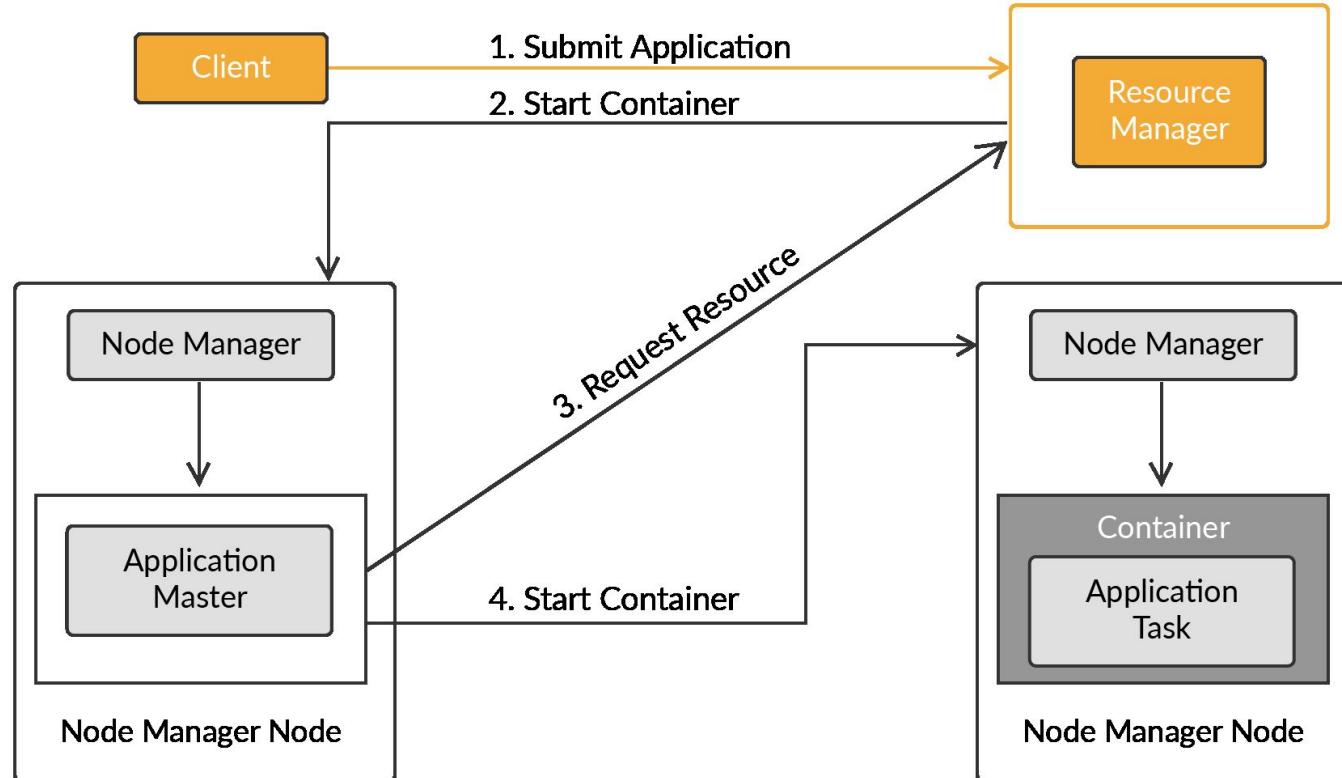
Node Manager

- Installed in slave nodes
- Launches and monitors containers

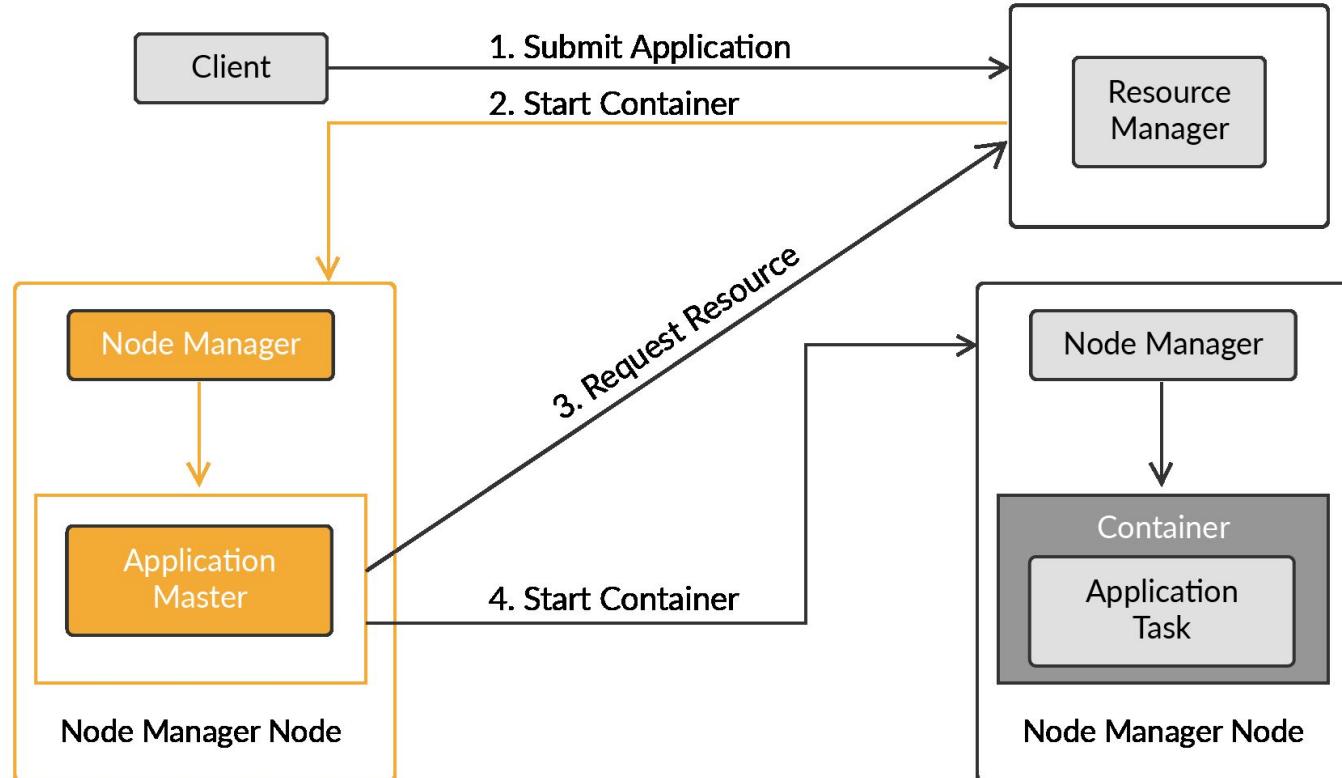
Containers

- Consist of processors and memory hardware required for processing tasks

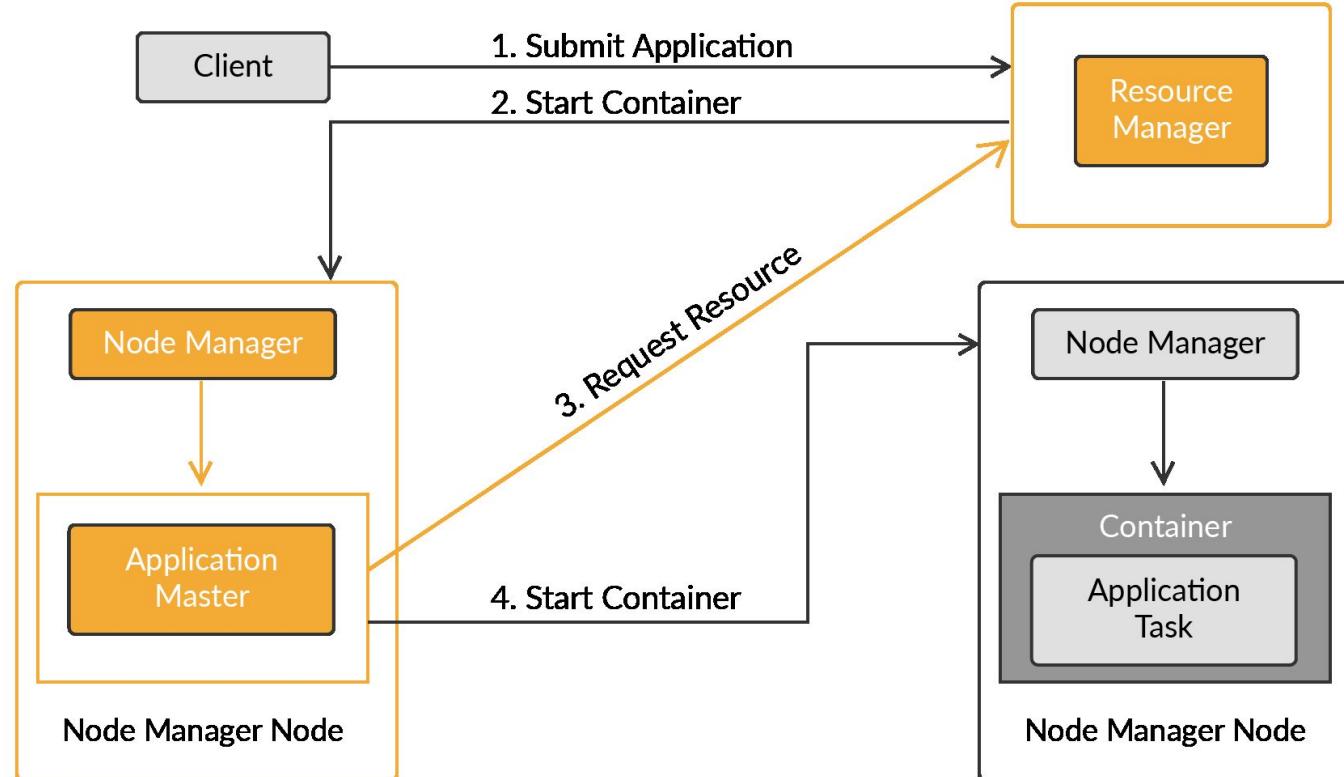
Task Processing in Hadoop



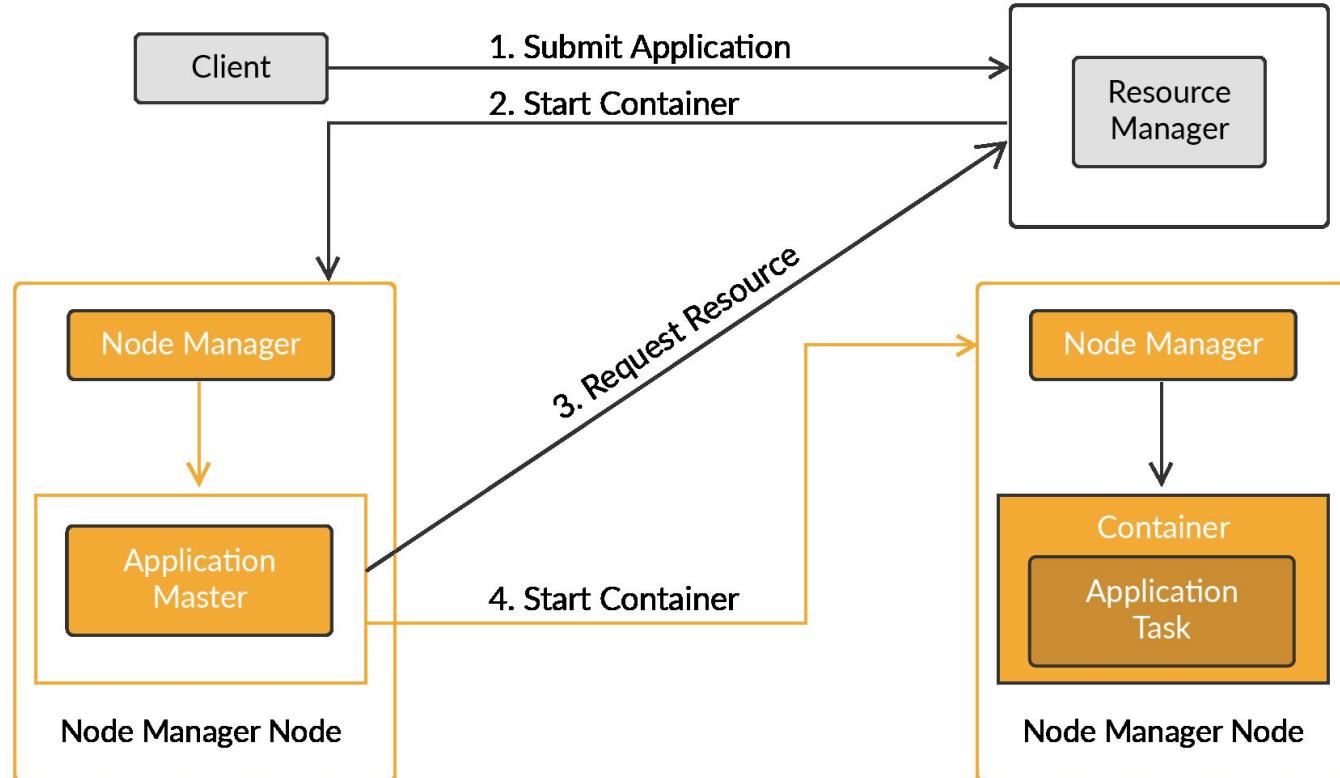
Task Processing in Hadoop



Task Processing in Hadoop



Task Processing in Hadoop



1

Recap of the components of
YARN

2

Discussed the task processing
steps of YARN

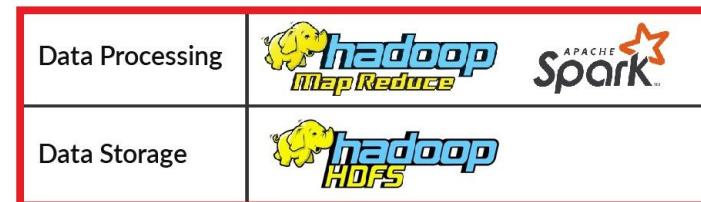
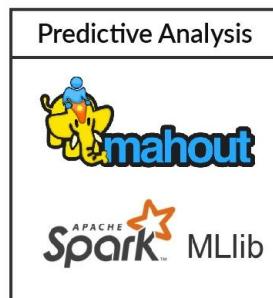
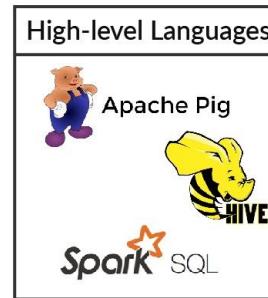
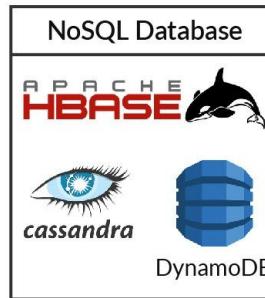
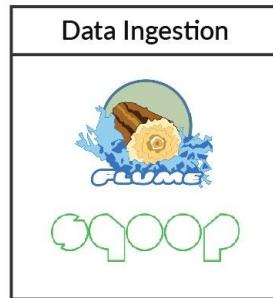
Segment - 09

Tools for Hadoop

1

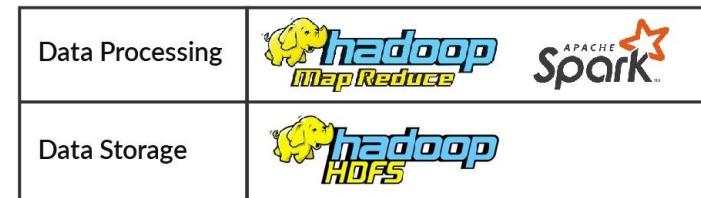
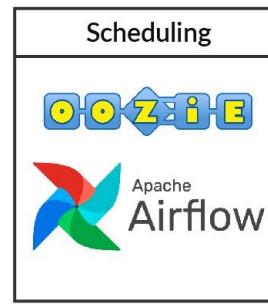
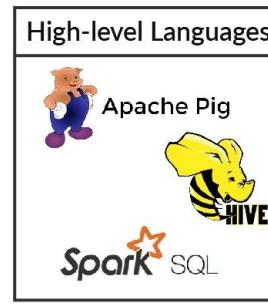
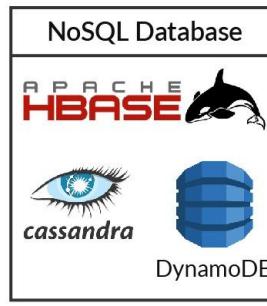
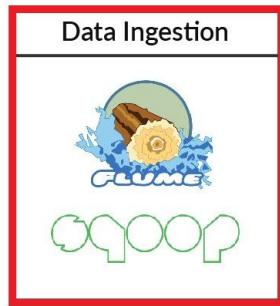
**Discuss the various tools used in
the Hadoop Ecosystem**

HADOOP ECOSYSTEM



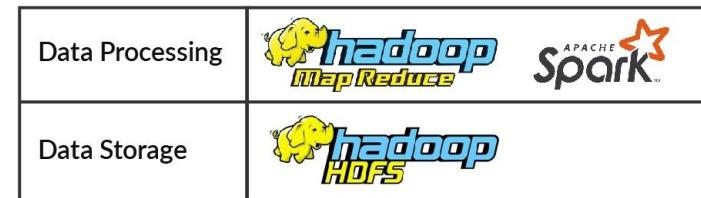
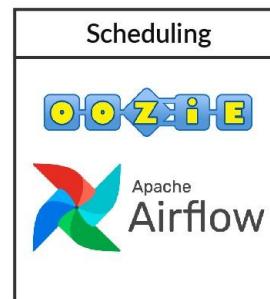
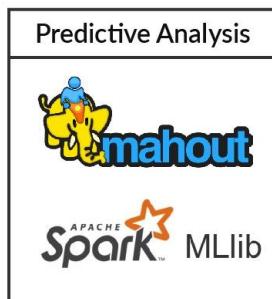
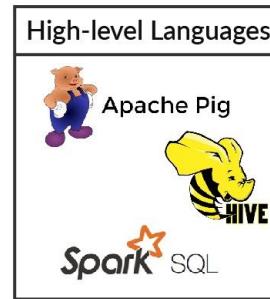
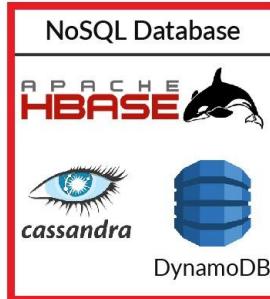
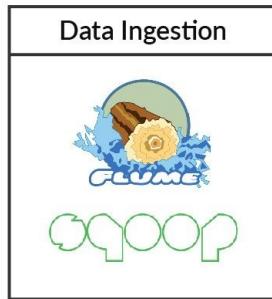
Tools for Hadoop

HADOOP ECOSYSTEM

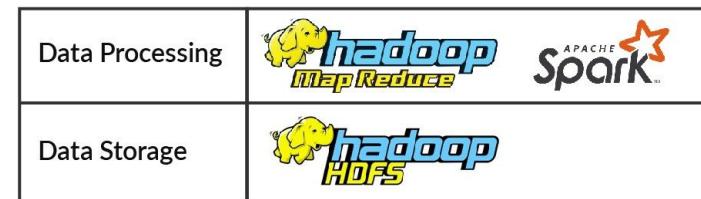
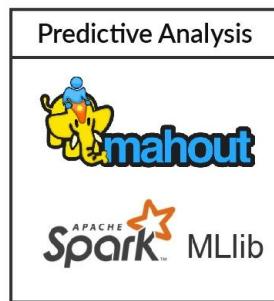
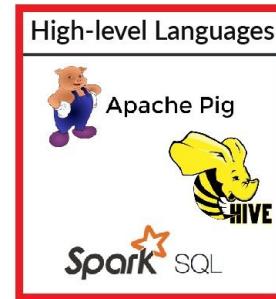
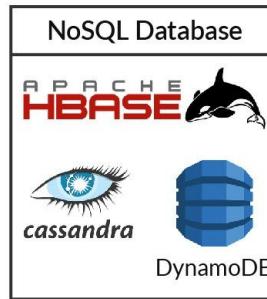
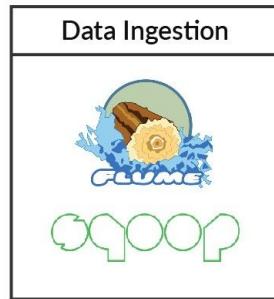


Tools for Hadoop

HADOOP ECOSYSTEM

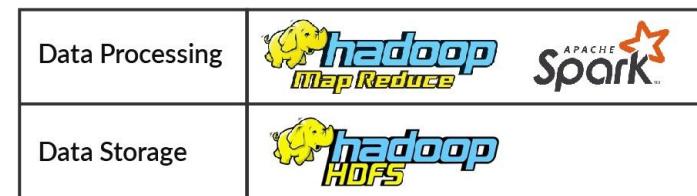
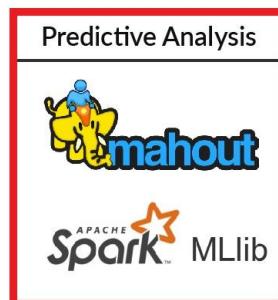
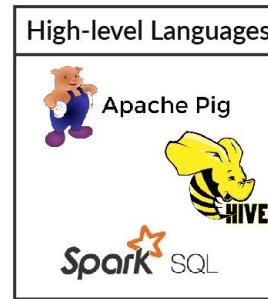
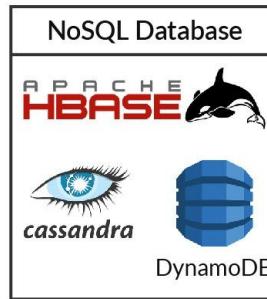
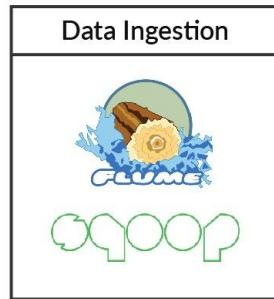


HADOOP ECOSYSTEM



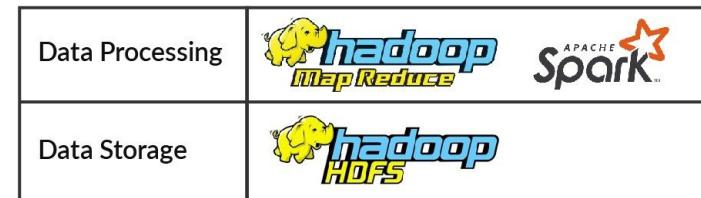
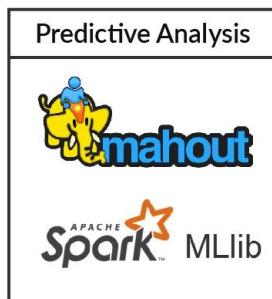
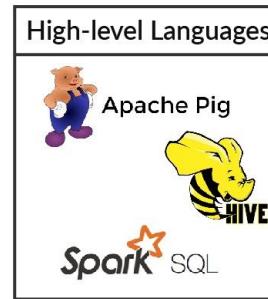
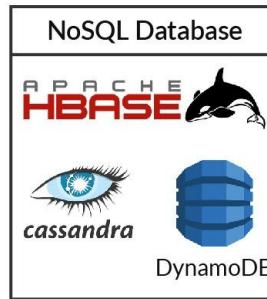
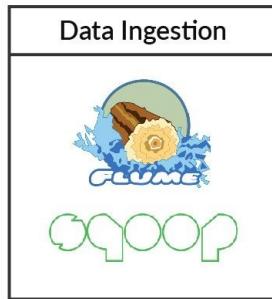
Tools for Hadoop

HADOOP ECOSYSTEM

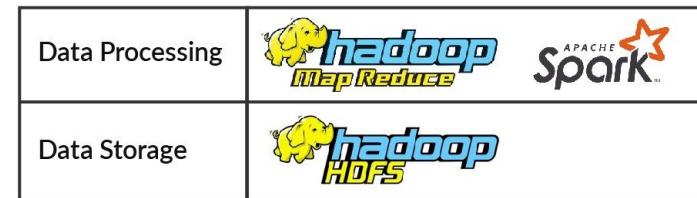
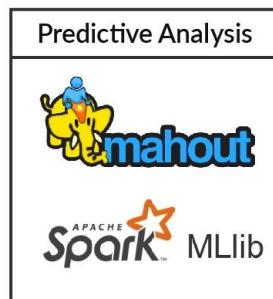
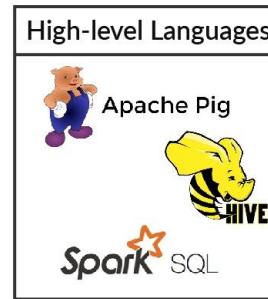
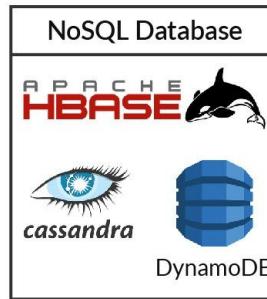
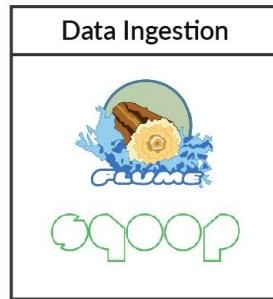


Tools for Hadoop

HADOOP ECOSYSTEM



HADOOP ECOSYSTEM



1

**Discussed the various tools used
in the Hadoop Ecosystem**

Session Summary

1

Introduced to distributed systems and the need for the same

2

Introduced to GFS and MapReduce on which Hadoop is based

3

Introduced Hadoop and Hadoop 2.x along with their features and the basic architecture

4

Introduced the concept of YARN and task processing in Hadoop

5

Discussed the various tools used in the Hadoop Ecosystem

Thank you