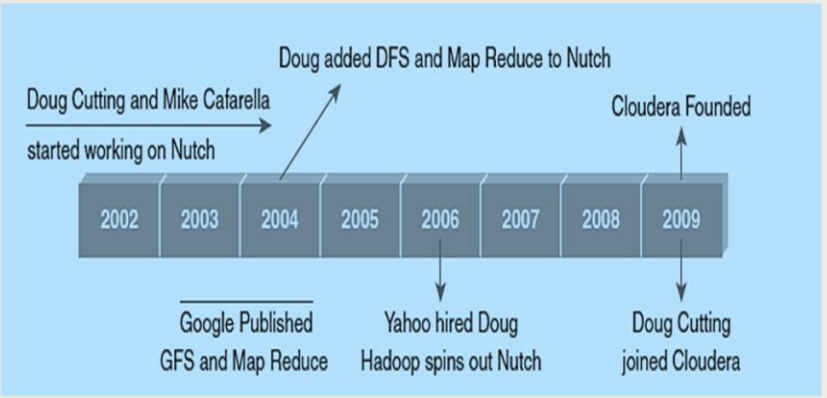
HADOOP

Learning Objectives

- After reading this chapter, you will be able to:
  - *Understand the need for Hadoop.*
  - *Learn the assumptions and goals of Hadoop.*
  - *Learn about the Hadoop components.*
  - *Understand and learn Hadoop Distributed File System.*
  - *Learn how HDFS handles job processing.*
  - *Learn about the MapReduce components.*
  - *Learn the use of Hadoop ecosystem.*
  - *Understand the limitation of Hadoop.*

History of Hadoop



What is Hadoop?

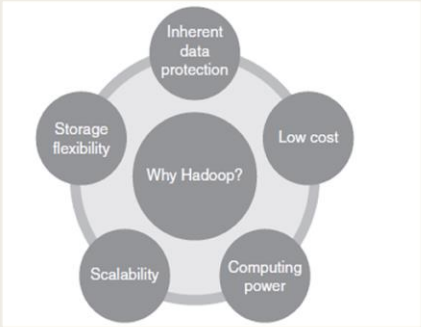
- Apache Hadoop is a framework that allows distributed processing of large datasets across clusters of commodity computers using a simple programming model.
- It is designed to scale-up from single servers to thousands of machines, each providing computation and storage.
- Rather than rely on hardware to deliver high-availability, the framework itself is designed to detect and handle failures at the application layer, thus delivering a highly available service on top of a cluster of computers, each of which may be prone to failures.

What is Hadoop?

- In short, Hadoop is an open-source software framework for storing and processing big data in a distributed way on large clusters of commodity hardware. Basically, it accomplishes the following two tasks:
  - *Massive data storage.*
  - *Faster processing.*

Why Hadoop?

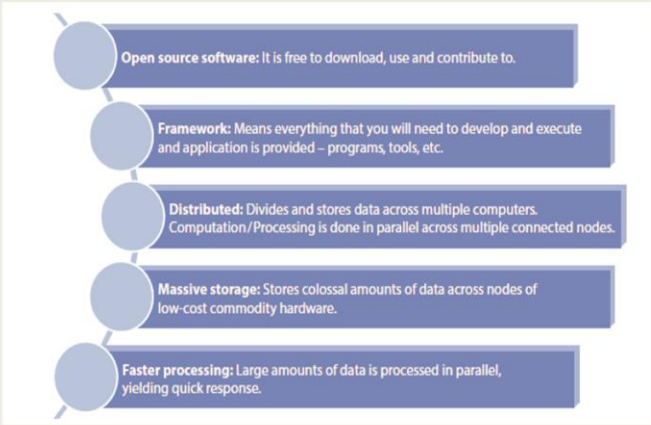
- Problems in data transfer made the organizations to think about an alternate way.



RDBMS versus HADOOP

PARAMETERS	RDBMS	HADOOP
System	Relational Database Management System.	Node Based Flat Structure.
Data	Suitable for structured data.	Suitable for structured, unstructured data. Supports variety of data formats in real time such as XML, JSON, text based flat file formats, etc.
Processing	OLTP	Analytical, Big Data Processing
Choice	When the data needs consistent relationship.	Big Data processing, which does not require any consistent relationships between data.
Processor	Needs expensive hardware or high-end processors to store huge volumes of data.	In a Hadoop Cluster, a node requires only a processor, a network card, and few hard drives.
Cost	Cost around $10,000 to $14,000 per terabytes of storage.	Cost around $4,000 per terabytes of storage.

Key Aspects of Hadoop



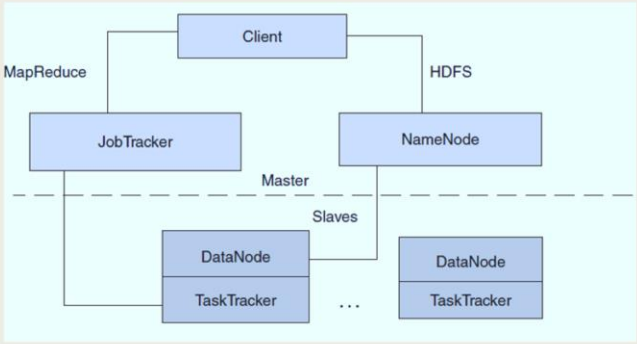
Hadoop Goals

- The main goals of Hadoop are listed below:
- **Scalable**: It can scale up from a single server to thousands of servers.
- **Fault tolerance**: It is designed with very high degree of fault tolerance.
- **Economical**: It uses commodity hardware instead of high-end hardware.
- **Handle hardware failures**: The resiliency of these clusters comes from the software's ability to detect and handle failures at the application layer.

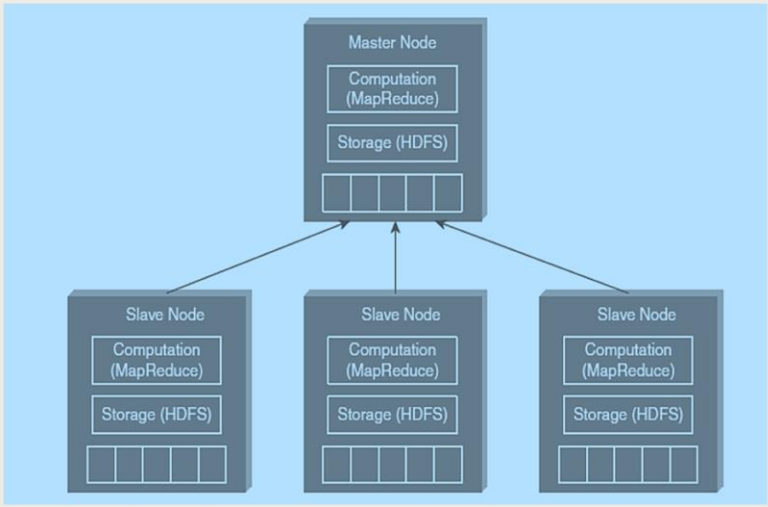
Hadoop Assumptions

- Hadoop was developed with large clusters of computers in mind with the following assumptions:
  - *Hardware will fail, since it considers a large cluster of computers.*
  - *Processing will be run in batches; so aims at high throughput as opposed to low latency.*
  - *Applications that run on Hadoop Distributed File System (HDFS) have large datasets typically from gigabytes to terabytes in size.*
  - *Portability is important.*
  - *Availability of high-aggregate data bandwidth and scale to hundreds of nodes in a single cluster.*
  - *Should support tens of millions of files in a single instance.*
  - *Applications need a write-once-read-many access model.*

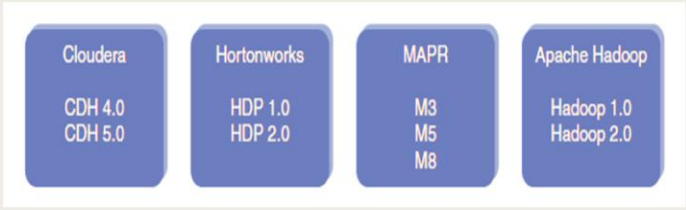
Hadoop High Level Architecture



Hadoop High Level Architecture



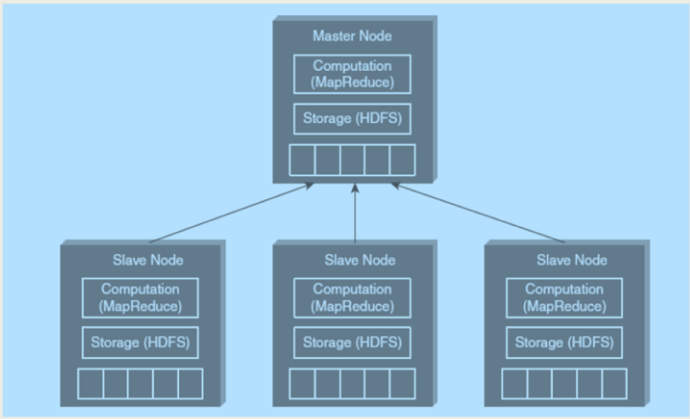
Hadoop Distributors



Core Hadoop Components

- Hadoop consists of the following components:
- Hadoop Common
- Hadoop Distributed File System (HDFS)
- Hadoop MapReduce
- Hadoop *Yet Another Resource Negotiator* (YARN) (MapReduce 2.0)

Hadoop Distributed File System (HDFS)



TEST ME

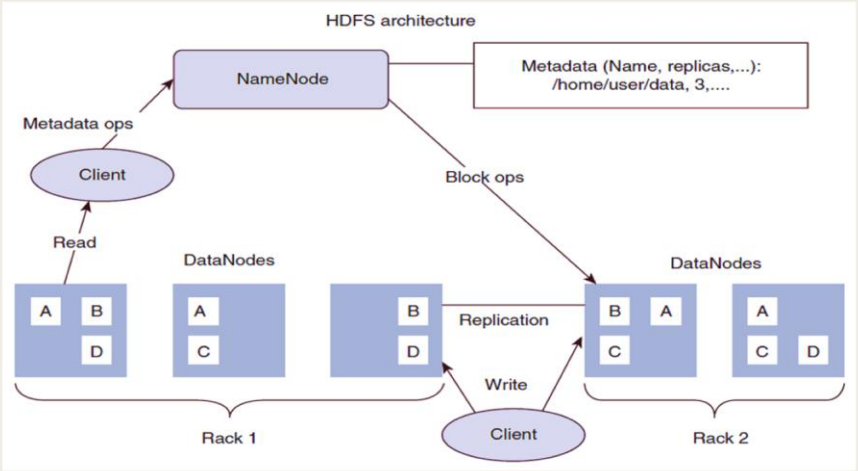
Hadoop Distributed File System (HDFS)

- HDFS is a distributed file system that provides a limited interface for managing the file system to allow it to scale and provide high throughput.
- HDFS creates multiple replicas of each data block and distributes them on computers throughout a cluster to enable reliable and rapid access.
- When a file is loaded into HDFS, it is replicated and fragmented into "blocks" of data, which are stored across the cluster nodes; the cluster nodes are also called the DataNodes.
- The NameNode is responsible for storage and management of metadata, so that when MapReduce or another execution framework calls for the data,the NameNode informs it where the data that is needed resides.

Features of HDFS

- Cost-effective
- Large Datasets/ Variety and volume of data
- Replication
- Fault Tolerance and reliability
- High Availability
- Scalability
- Data Integrity
- High Throughput
- Data Locality

NameNode and DataNode block replication



Hadoop Distributed File System (HDFS)

- Figure shows the NameNode and DataNode block replication in HDFS architecture.
  - *HDFS creates multiple replicas of data blocks for reliability, placing them on the computer nodes around the cluster.*
  - *Hadoop's target is to run on clusters of the order of 10,000 nodes.*
  - *A file consists of many 64 MB blocks.*

Main Components of HDFS

- NameNode
- NameNode is the master that contains the metadata. In general, it maintains the directories and files and manages the blocks which are present on the DataNode. The following are the functions of NameNode:
  - *Manages namespace of the file system in memory.*
  - *Maintains "inode" information.*
  - *Maps inode to the list of blocks and locations.*
  - *Takes care of authorization and authentication.*
  - *Creates checkpoints and logs the namespace changes.*
- So the NameNode maps DataNode to the list of blocks, monitors status (health) of DataNode and replicates the missing blocks.

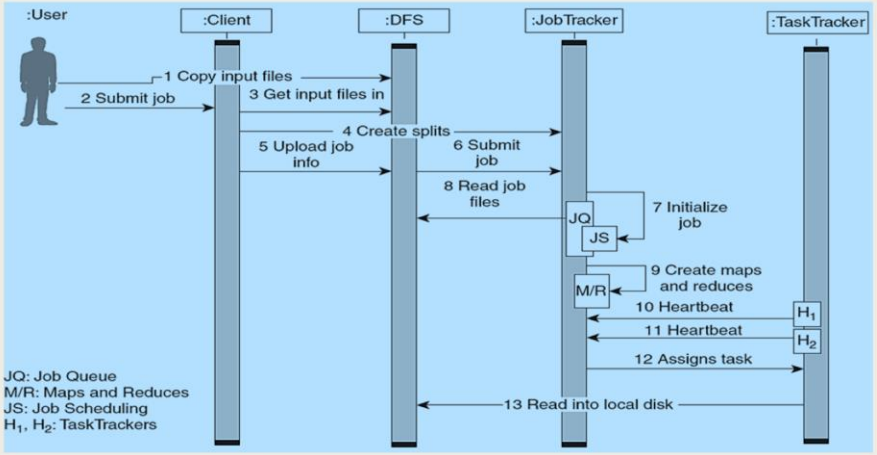
Main Components of HDFS

- Namenode is responsible for executing operations such as opening and closing of files, no data actually flows through the Namenode.
- Namenode executes the read and write operations while the data is transferred directly to/from the Datanodes.
- HDFS splits files into blocks, and the blocks are stored on the Datanodes. For each block,multiple replicas are kept.
- Namenode persistently stores the filesystem meta-data and the mappings of the blocks to the datanodes, on the disk as two files: fsimage and edits files.
- The fsimage contains a complete snapshot of the filesystem meta-data. The edits file stores the incremental updates to the meta-data.
- When the Namenode starts, it loads the fsimage file into the memory and applies the edits file to bring the in-memory view of the filesystem up-to-date. Namenode then writes a new fsimage file to the disk.

Main Components of HDFS

- DataNodes
- DataNodes are the slaves which provide the actual storage and are deployed on each machine. They are responsible for processing read and write requests for the clients. The following are the other functions of DataNode:
  - *Handles block storage on multiple volumes and also maintain block integrity.*
  - *Periodically sends heartbeats and also the block reports to NameNode.*

How HDFS handles job processing requests from the user

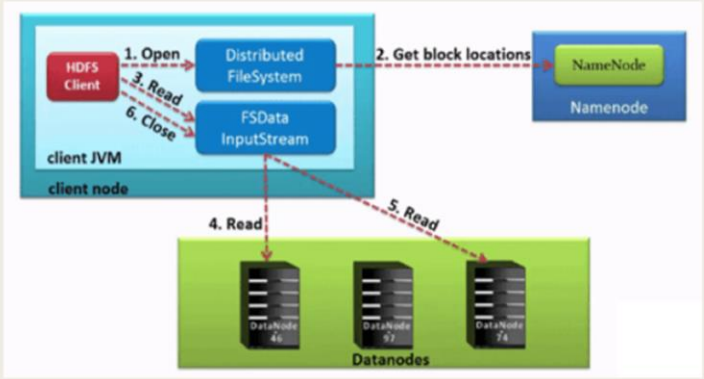


Data Blocks & Replication

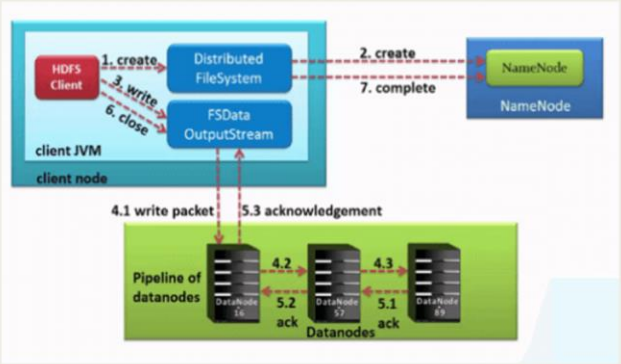
- Blocks are replicated on the Datanodes and by default three replicas are created.
- The placement of replicas on the Datanodes is determined by a rack-aware placement policy. This placement policy ensures reliability and availability of the blocks.
- For a replication factor of three, one replica is placed on a node on a local rack, the second replica is placed on a different node on a remote rack and the third replica is placed on a different node on the same remote rack.
- This ensures that even if the rack becomes unavailable, at least one replica will remain available. Placement of replicas on different nodes in the same rack minimizes the network traffic between the racks.

Test me

Apache HDFS File Read Operation



Apache HDFS File Write Operation



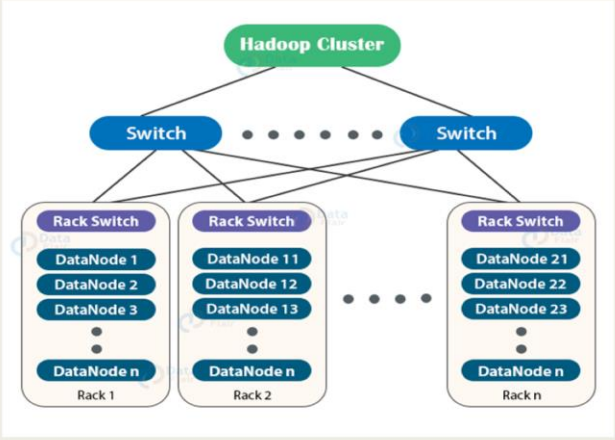
What is a rack?

- The Rack is the collection of around 40-50 DataNodes connected using the same network switch. If the network goes down, the whole rack will be unavailable. A large Hadoop cluster is deployed in multiple racks.
- In a large Hadoop cluster, there are multiple racks. Each rack consists of DataNodes. Communication between the DataNodes on the same rack is more efficient as compared to the communication between DataNodes residing on different racks.

What is Rack Awareness in Hadoop HDFS?

- To reduce the network traffic during file **read/write**, NameNode chooses the closest DataNode for serving the client read/write request.
- NameNode maintains **rack ids** of each DataNode to achieve this rack information. This concept of choosing the closest DataNode based on the rack information is known as **Rack Awareness.**
- A default Hadoop installation assumes that all the DataNodes reside on the same rack.

What is Rack Awareness in Hadoop HDFS?



Why Rack Awareness?

- The reasons for the Rack Awareness in Hadoop are:
- To reduce the network traffic while file read/write, which improves the cluster performance.
- To achieve **fault tolerance**, even when the rack goes down
- Achieve high availability of data so that data is available even in unfavorable conditions.
- To reduce the latency, that is, to make the file read/write operations done with lower delay.
- NameNode uses a rack awareness algorithm while placing the replicas in HDFS.

Test Me

Hadoop HDFS Commands

- With the help of the HDFS command, we can perform Hadoop HDFS file operations like changing the file permissions, viewing the file contents, creating files or directories, copying file/directory from the local file system to HDFS or vice-versa, etc.
- Before starting with the HDFS command, we have to start the Hadoop services. To start the Hadoop services do the following:
- Move to the **~/hadoop-3.1.2** directory
- Start Hadoop service by using the command
  - *sbin/start-dfs.sh*

Hadoop HDFS Commands

- Hadoop HDFS version Command
  - *hadoop version*
- Hadoop HDFS mkdir Command
  - *hadoop fs –mkdir /path/directory_name*
- Using the **ls** command, we can check for the directories in HDFS.
  - *hadoop fs -ls /path*
- Hadoop HDFS put Command
  - *haoop fs -put <localsrc> <dest>*

References

- https://beyondcorner.com/apache-hdfs-read-write-operations/
- https://data-flair.training/blogs/rack-awareness-hadoop-hdfs/