

Hadoop

Hadoop is an open source framework. It is provided by Apache to process and analyze very huge volume of data. It is written in Java and currently used by Google, Facebook, LinkedIn, Yahoo, Twitter etc.

What is Big Data

Data which are very large in size is called Big Data. Normally we work on data of size MB(WordDoc ,Excel) or maximum GB(Movies, Codes) but data in Peta bytes i.e. 10^{15} byte size is called Big Data. It is stated that almost 90% of today's data has been generated in the past 3 years.

Sources of Big Data

These data come from many sources like

- **Social networking sites:** Facebook, Google, LinkedIn all these sites generates huge amount of data on a day to day basis as they have billions of users worldwide.
- **E-commerce site:** Sites like Amazon, Flipkart, Alibaba generates huge amount of logs from which users buying trends can be traced.
- **Weather Station:** All the weather station and satellite gives very huge data which are stored and manipulated to forecast weather.
- **Telecom company:** Telecom giants like Airtel, Vodafone study the user trends and accordingly publish their plans and for this they store the data of its million users.
- **Share Market:** Stock exchange across the world generates huge amount of data through its daily transaction.
- **Black Box Data** – It is a component of helicopter, airplanes, and jets, etc. It captures voices of the flight crew, recordings of microphones and earphones, and the performance information of the aircraft.
- **Power Grid Data** – The power grid data holds information consumed by a particular node with respect to a base station.
- **Transport Data** – Transport data includes model, capacity, distance and availability of a vehicle.
- **Search Engine Data** – Search engines retrieve lots of data from different databases.

3V's of Big Data

1. **Velocity:** The data is increasing at a very fast rate. It is estimated that the volume of data will double in every 2 years.
2. **Variety:** Now a days data are not stored in rows and column. Data is structured as well as unstructured. Log file, CCTV footage is unstructured data. Data which can be saved in tables are structured data like the transaction data of the bank.
3. **Volume:** The amount of data which we deal with is of very large size of Peta bytes.

Thus Big Data includes huge volume, high velocity, and extensible variety of data. The data in it will be of three types.

Structured data – Relational data.

Semi Structured data – XML data.

Unstructured data – Word, PDF, Text, Media Logs.

Benefits of Big Data

Using the information kept in the social network like Facebook, the marketing agencies are learning about the response for their campaigns, promotions, and other advertising mediums.

Using the information in the social media like preferences and product perception of their consumers, product companies and retail organizations are planning their production.

Using the data regarding the previous medical history of patients, hospitals are providing better and quick service.

Big Data Challenges

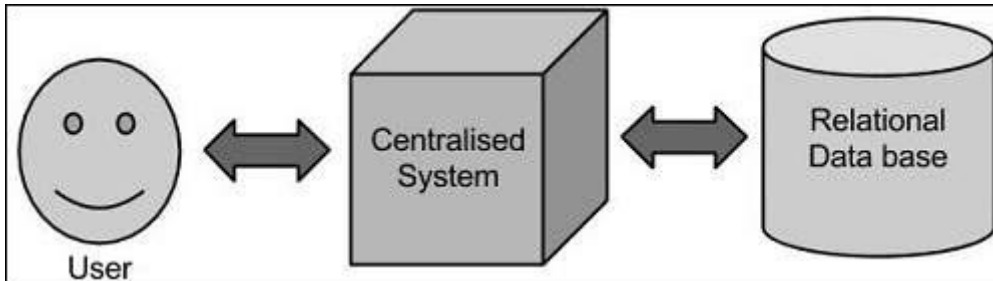
The major challenges associated with big data are as follows –

- Capturing data
- Storage
- Searching
- Sharing
- Transfer
- Analysis
- Presentation

Big Data Solutions

Traditional Approach

In this approach, an enterprise will have a computer to store and process big data. For storage purpose, the programmers will take the help of their choice of database vendors such as Oracle, IBM, etc. In this approach, the user interacts with the application, which in turn handles the part of data storage and analysis.

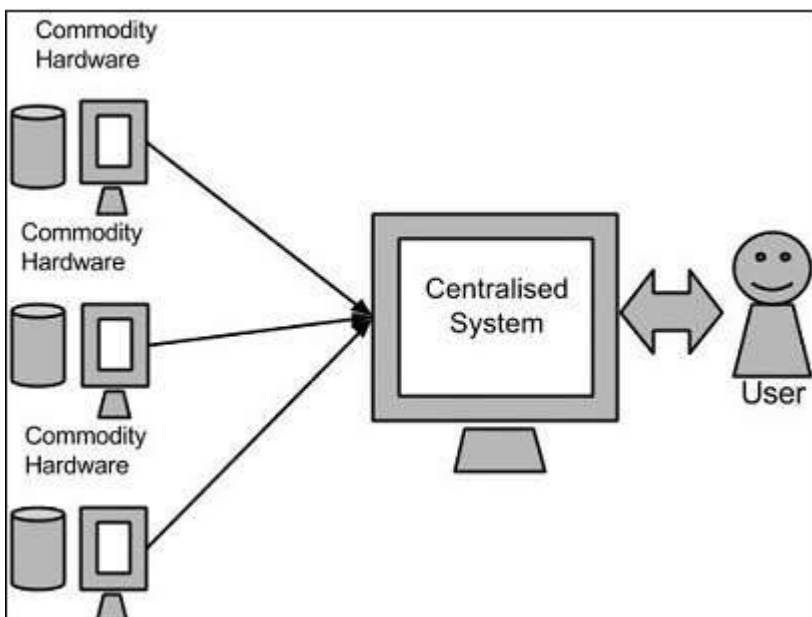


Limitation

This approach works fine with those applications that process less voluminous data that can be accommodated by standard database servers, or up to the limit of the processor that is processing the data. But when it comes to dealing with huge amounts of scalable data, it is a hectic task to process such data through a single database bottleneck.

Google's Solution

Google solved this problem using an algorithm called MapReduce. This algorithm divides the task into small parts and assigns them to many computers, and collects the results from them which when integrated, form the result dataset.



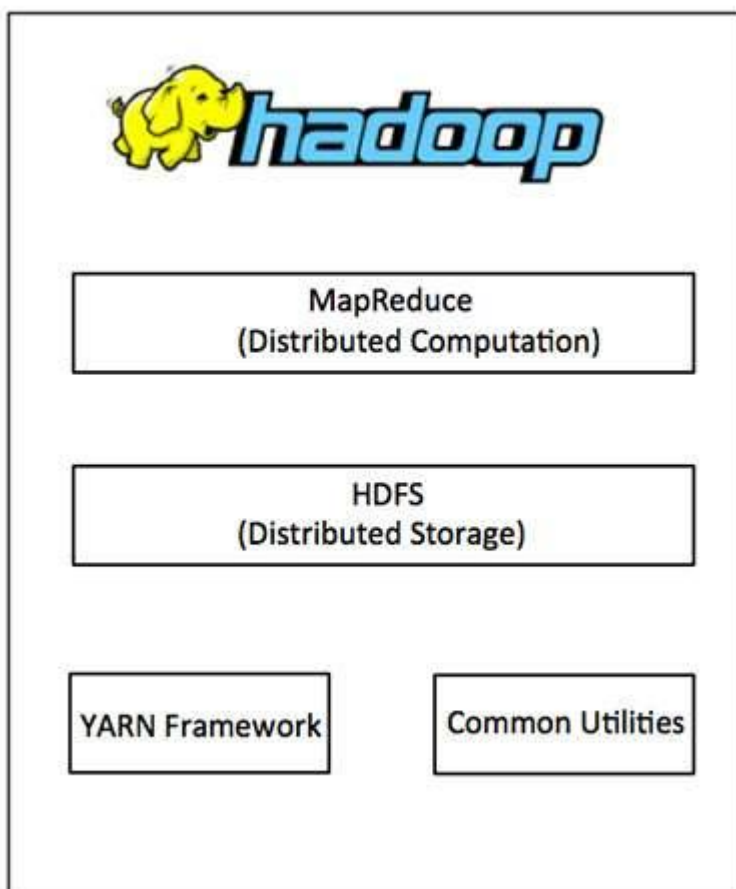
Hadoop - Introduction

Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models. The Hadoop framework application works in an environment that provides distributed *storage* and *computation* across clusters of computers. Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.

Hadoop Architecture

At its core, Hadoop has two major layers namely –

- Processing/Computation layer (MapReduce), and
- Storage layer (Hadoop Distributed File System).



MapReduce

MapReduce is a parallel programming model for writing distributed applications devised at Google for efficient processing of large amounts of data (multi-terabyte data-sets), on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. The MapReduce program runs on Hadoop which is an Apache open-source framework.

Hadoop Distributed File System

The Hadoop Distributed File System (HDFS) is based on the Google File System (GFS) and provides a distributed file system that is designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. It is highly fault-tolerant and is designed to be deployed on low-cost hardware. It provides high throughput access to application data and is suitable for applications having large datasets.

Apart from the above-mentioned two core components, Hadoop framework also includes the following two modules –

Hadoop Common – These are Java libraries and utilities required by other Hadoop modules.

Hadoop YARN – This is a framework for job scheduling and cluster resource management.

How Does Hadoop Work?

It is quite expensive to build bigger servers with heavy configurations that handle large scale processing, but as an alternative, you can tie together many commodity computers with single-CPU, as a single functional distributed system and practically, the clustered machines can read the dataset in parallel and provide a much higher throughput. Moreover, it is cheaper than one high-end server. So this is the first motivational factor behind using Hadoop that it runs across clustered and low-cost machines.

Hadoop runs code across a cluster of computers. This process includes the following core tasks that Hadoop performs –

- Data is initially divided into directories and files. Files are divided into uniform sized blocks of 128M and 64M (preferably 128M).
- These files are then distributed across various cluster nodes for further processing.
- HDFS, being on top of the local file system, supervises the processing.
- Blocks are replicated for handling hardware failure.
- Checking that the code was executed successfully.
- Performing the sort that takes place between the map and reduce stages.
- Sending the sorted data to a certain computer.
- Writing the debugging logs for each job.

Advantages of Hadoop

- Hadoop framework allows the user to quickly write and test distributed systems. It is efficient, and it automatically distributes the data and work across the machines and in turn, utilizes the underlying parallelism of the CPU cores.
- Hadoop does not rely on hardware to provide fault-tolerance and high availability (FTHA), rather Hadoop library itself has been designed to detect and handle failures at the application layer.

- Servers can be added or removed from the cluster dynamically and Hadoop continues to operate without interruption.
- Another big advantage of Hadoop is that apart from being open source, it is compatible on all the platforms since it is Java based.

What is HDFS

Hadoop comes with a distributed file system called HDFS. In HDFS data is distributed over several machines and replicated to ensure their durability to failure and high availability to parallel application.

It is cost effective as it uses commodity hardware. It involves the concept of blocks, data nodes and node name.

Where to use HDFS

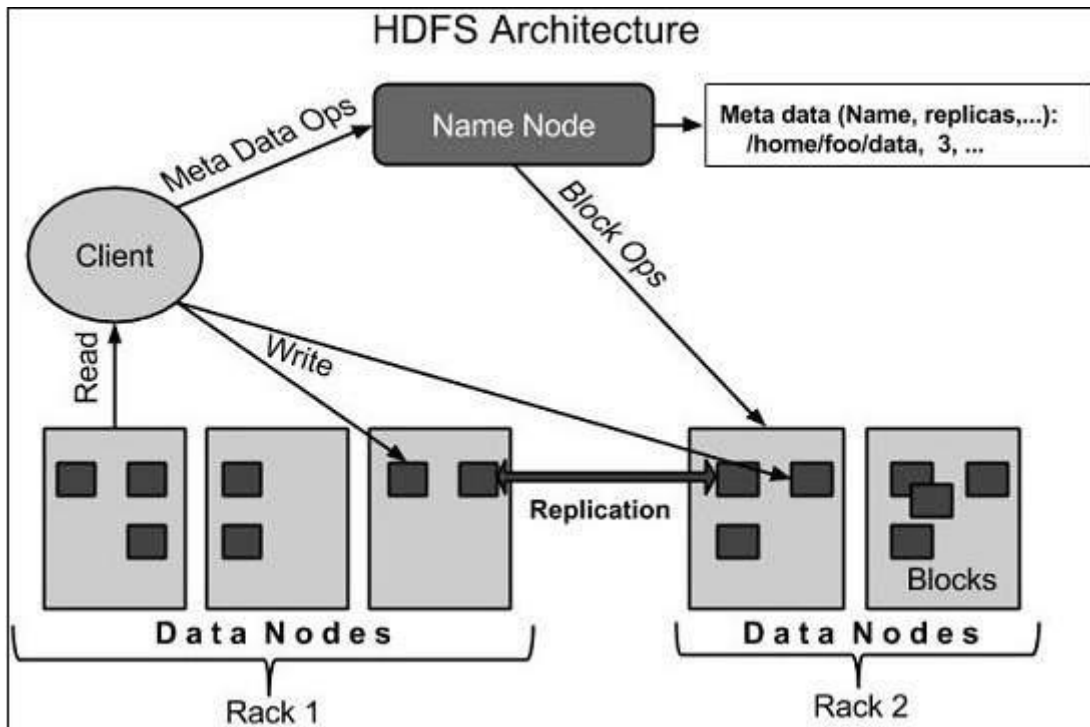
- **Very Large Files:** Files should be of hundreds of megabytes, gigabytes or more.
- **Streaming Data Access:** The time to read whole data set is more important than latency in reading the first. HDFS is built on write-once and read-many-times pattern.
- **Commodity Hardware:** It works on low cost hardware.

Where not to use HDFS

- **Low Latency data access:** Applications that require very less time to access the first data should not use HDFS as it is giving importance to whole data rather than time to fetch the first record.
- **Lots Of Small Files:** The name node contains the metadata of files in memory and if the files are small in size it takes a lot of memory for name node's memory which is not feasible.
- **Multiple Writes:** It should not be used when we have to write multiple times.

HDFS Architecture

Given below is the architecture of a Hadoop File System.



HDFS follows the master-slave architecture and it has the following elements.

Namenode

The namenode is the commodity hardware that contains the GNU/Linux operating system and the namenode software. It is a software that can be run on commodity hardware. The system having the namenode acts as the master server and it does the following tasks –

- Manages the file system namespace.
- Regulates client's access to files.
- It also executes file system operations such as renaming, closing, and opening files and directories.

HDFS works in master-worker pattern where the name node acts as master. Name Node is controller and manager of HDFS as it knows the status and the metadata of all the files in HDFS; the metadata information being file permission, names and location of each block. The metadata are small, so it is stored in the memory of name node, allowing faster access to data. Moreover the HDFS cluster is accessed by multiple clients concurrently, so all this information is handled by a single machine. The file system operations like opening, closing, renaming etc. are executed by it.

Datanode

The datanode is a commodity hardware having the GNU/Linux operating system and datanode software. For every node (Commodity hardware/System) in a cluster, there will be a datanode. These nodes manage the data storage of their system.

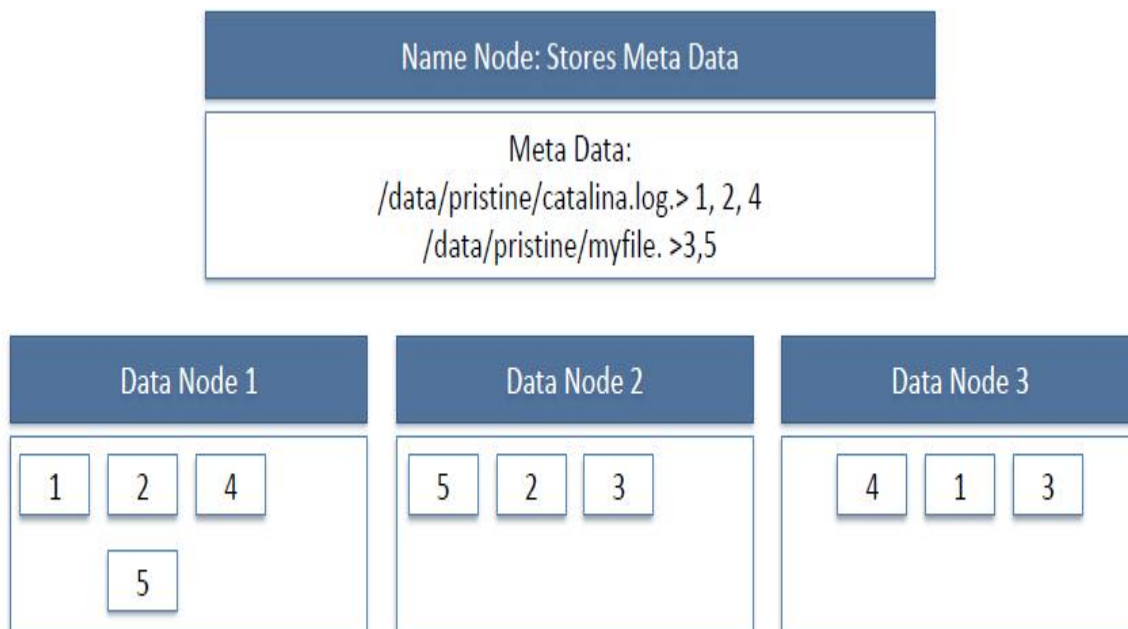
- Datanodes perform read-write operations on the file systems, as per client request.
- They also perform operations such as block creation, deletion, and replication according to the instructions of the namenode.

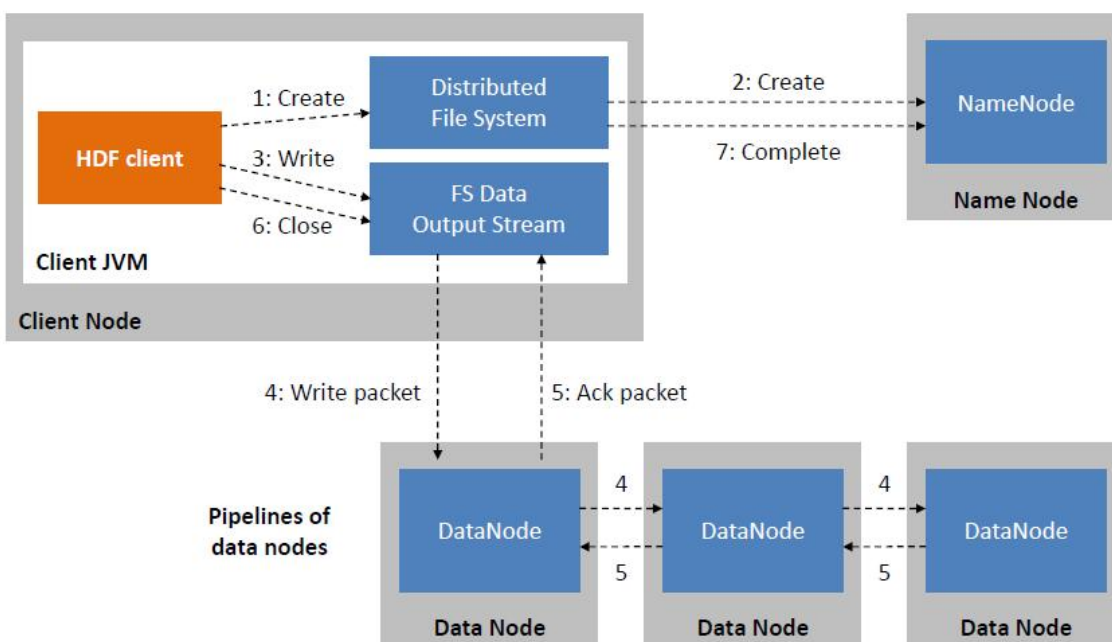
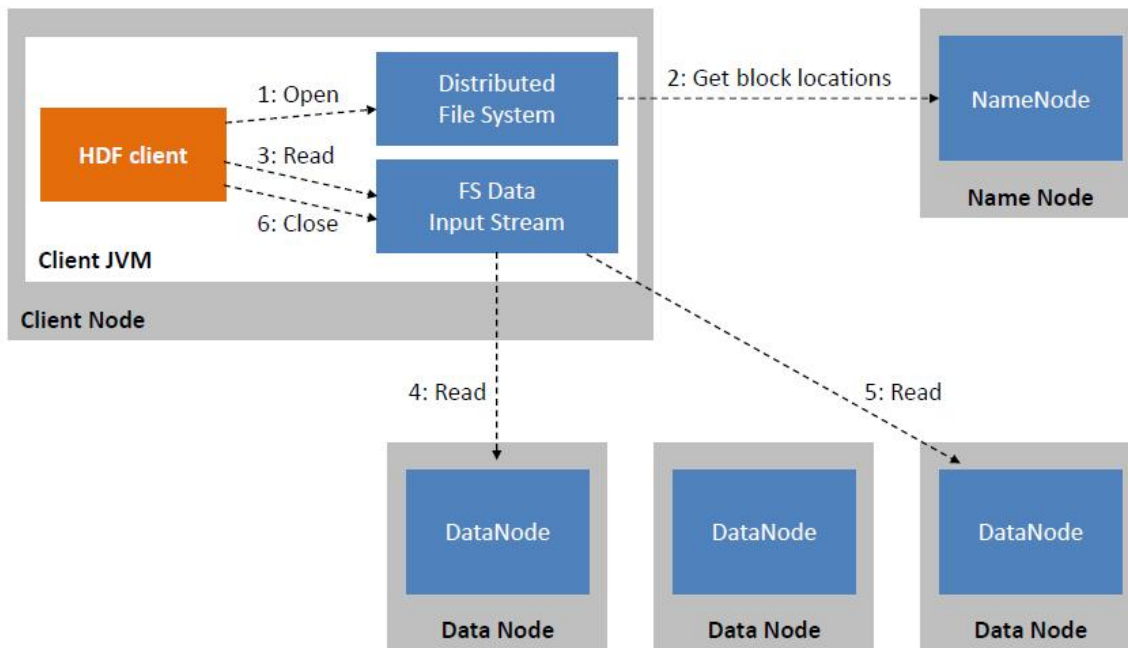
They store and retrieve blocks when they are told to; by client or name node. They report back to name node periodically, with list of blocks that they are storing. The data node being a commodity hardware also does the work of block creation, deletion and replication as stated by the name node.

Block

Generally the user data is stored in the files of HDFS. The file in a file system will be divided into one or more segments and/or stored in individual data nodes. These file segments are called as blocks. In other words, the minimum amount of data that HDFS can read or write is called a Block. The default block size is 64MB, but it can be increased as per the need to change in HDFS configuration.

A Block is the minimum amount of data that it can read or write. HDFS blocks are 128 MB by default and this is configurable. Files in HDFS are broken into block-sized chunks, which are stored as independent units. Unlike a file system, if the file is in HDFS is smaller than block size, then it does not occupy full block's size, i.e. 5 MB of file stored in HDFS of block size 128 MB takes 5MB of space only. The HDFS block size is large just to minimize the cost of seek.





Components Of YARN

- **Client:** For submitting MapReduce jobs.
- **Resource Manager:** To manage the use of resources across the cluster
- **Node Manager:** For launching and monitoring the computer containers on machines in the cluster.

- **Map Reduce Application Master:** Checks tasks running the MapReduce job. The application master and the MapReduce tasks run in containers that are scheduled by the resource manager, and managed by the node managers.

Benefits of YARN

- **Scalability:** Map Reduce 1 hits scalability bottleneck at 4000 nodes and 40000 task, but Yarn is designed for 10,000 nodes and 1 lakh tasks.
- **Utilization:** Node Manager manages a pool of resources, rather than a fixed number of the designated slots thus increasing the utilization.
- **Multitenancy:** Different version of MapReduce can run on YARN, which makes the process of upgrading MapReduce more manageable.

A MapReduce is a data processing tool which is used to process the data parallelly in a distributed form. It was developed in 2004, on the basis of paper titled as "MapReduce: Simplified Data Processing on Large Clusters," published by Google.

The MapReduce is a paradigm which has two phases, the mapper phase, and the reducer phase. In the Mapper, the input is given in the form of a key-value pair. The output of the Mapper is fed to the reducer as input. The reducer runs only after the Mapper is over. The reducer too takes input in key-value format, and the output of reducer is the final output.

MapReduce

MapReduce is a framework using which we can write applications to process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner.

What is MapReduce?

MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This

simple scalability is what has attracted many programmers to use the MapReduce model.

The Algorithm

Generally MapReduce paradigm is based on sending the computer to where the data resides!

MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.

Map stage – The map or mapper's job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.

Reduce stage – This stage is the combination of the **Shuffle** stage and the **Reduce** stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.

During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.

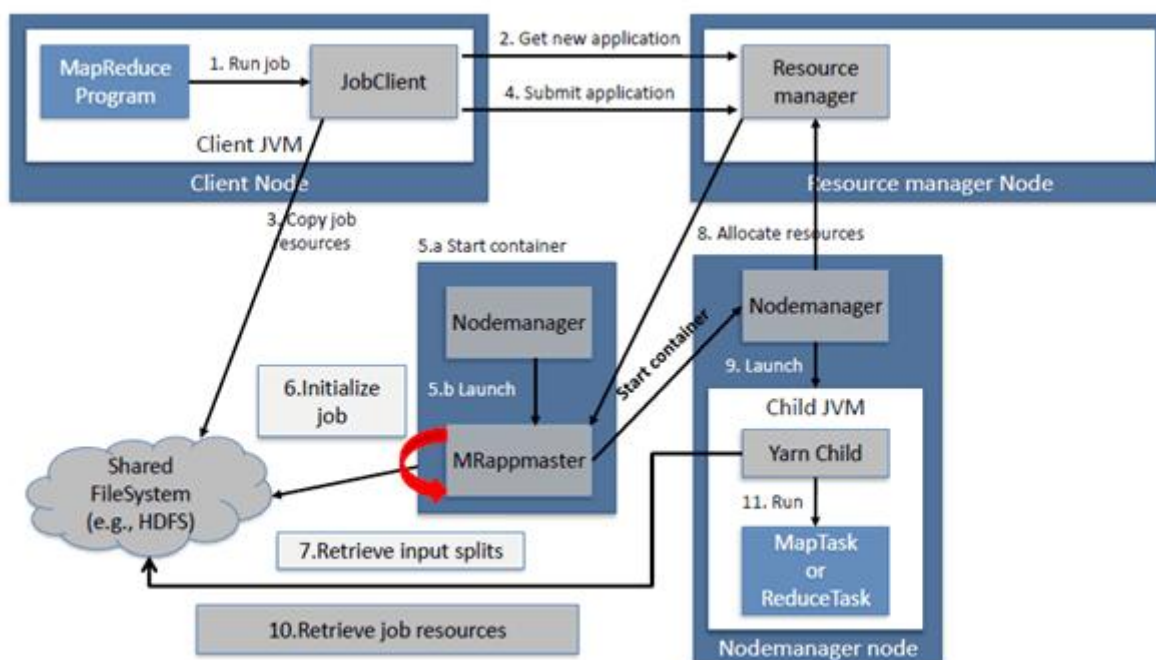
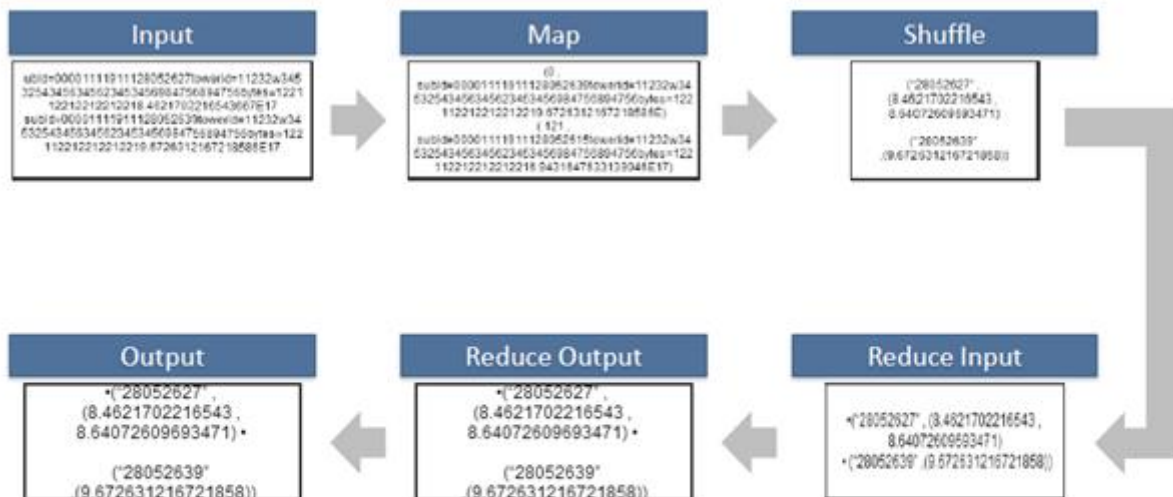
The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.

Most of the computing takes place on nodes with data on local disks that reduces the network traffic.

After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.

Steps in Map Reduce

- The map takes data in the form of pairs and returns a list of <key, value> pairs. The keys will not be unique in this case.
- Using the output of Map, sort and shuffle are applied by the Hadoop architecture. This sort and shuffle acts on these list of <key, value> pairs and sends out unique keys and a list of values associated with this unique key <key, list(values)>.
- An output of sort and shuffle sent to the reducer phase. The reducer performs a defined function on a list of values for unique keys, and Final output <key, value> will be stored/displayed.

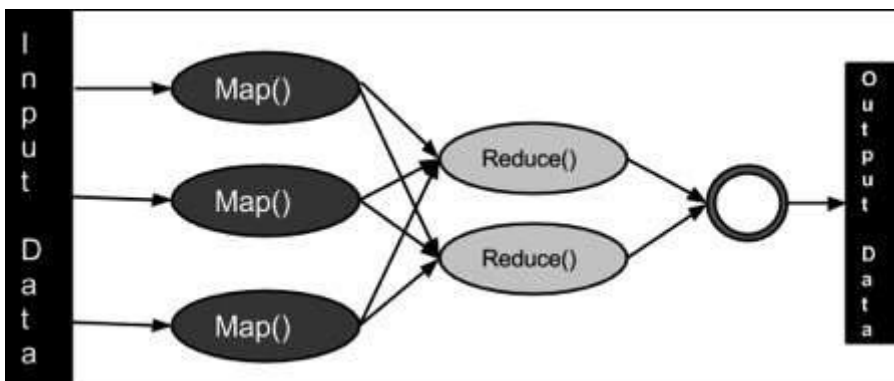


Sort and Shuffle

The sort and shuffle occur on the output of Mapper and before the reducer. When the Mapper task is complete, the results are sorted by key, partitioned if there are multiple reducers, and then written to disk. Using the input from each Mapper $\langle k_2, v_2 \rangle$, we collect all the values for each unique key k_2 . This output from the shuffle phase in the form of $\langle k_2, \text{list}(v_2) \rangle$ is sent as input to reducer phase.

Usage of MapReduce

- It can be used in various application like document clustering, distributed sorting, and web link-graph reversal.
- It can be used for distributed pattern-based searching.
- We can also use MapReduce in machine learning.
- It was used by Google to regenerate Google's index of the World Wide Web.
- It can be used in multiple computing environments such as multi-cluster, multi-core, and mobile environment.



Inputs and Outputs (Java Perspective)

The MapReduce framework operates on $\langle \text{key}, \text{value} \rangle$ pairs, that is, the framework views the input to the job as a set of $\langle \text{key}, \text{value} \rangle$ pairs and produces a set of $\langle \text{key}, \text{value} \rangle$ pairs as the output of the job, conceivably of different types.

The key and the value classes should be in serialized manner by the framework and hence, need to implement the Writable interface. Additionally, the key classes have to implement the Writable-Comparable interface to facilitate sorting by the framework. Input and Output types of a **MapReduce job** – (Input) $\langle k1, v1 \rangle \rightarrow \text{map} \rightarrow \langle k2, v2 \rangle \rightarrow \text{reduce} \rightarrow \langle k3, v3 \rangle$ (Output).

	Input	Output
Map	$\langle k1, v1 \rangle$	list ($\langle k2, v2 \rangle$)
Reduce	$\langle k2, \text{list}(v2) \rangle$	list ($\langle k3, v3 \rangle$)

Terminology

PayLoad – Applications implement the Map and the Reduce functions, and form the core of the job.

Mapper – Mapper maps the input key/value pairs to a set of intermediate key/value pair.

NamedNode – Node that manages the Hadoop Distributed File System (HDFS).

DataNode – Node where data is presented in advance before any processing takes place.

MasterNode – Node where JobTracker runs and which accepts job requests from clients.

SlaveNode – Node where Map and Reduce program runs.

JobTracker – Schedules jobs and tracks the assign jobs to Task tracker.

Task Tracker – Tracks the task and reports status to JobTracker.

Job – A program is an execution of a Mapper and Reducer across a dataset.

Task – An execution of a Mapper or a Reducer on a slice of data.

Task Attempt – A particular instance of an attempt to execute a task on a SlaveNode.