UNIT-V Module-II

Hadoop

Hadoop is an open source framework from Apache and is used to store process and analyze data which are very huge in volume. Hadoop is written in Java and is not OLAP (online analytical processing). It is used for batch/offline processing.It is being used by Facebook, Yahoo, Google, Twitter, LinkedIn and many more. Moreover it can be scaled up just by adding nodes in the cluster.

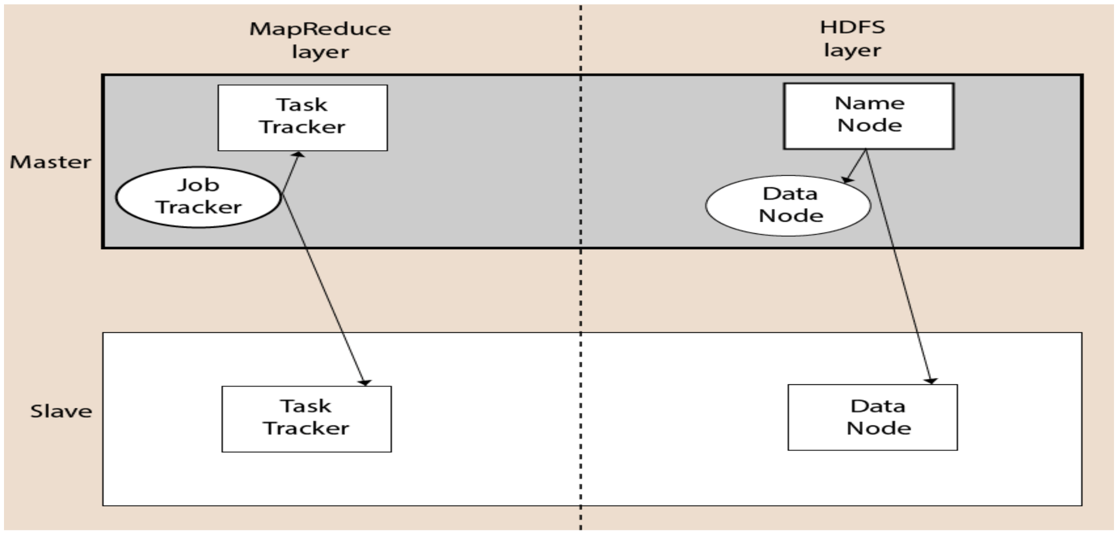
Modules of Hadoop

HDFS: Hadoop Distributed File System. Google published its paper GFS and on the basis of that HDFS was developed. It states that the files will be broken into blocks and stored in nodes over the distributed architecture.
Yarn: Yet another Resource Negotiator is used for job scheduling and manage the cluster.
Map Reduce: This is a framework which helps Java programs to do the parallel computation on data using key value pair. The Map task takes input data and converts it into a data set which can be computed in Key value pair. The output of Map task is consumed by reduce task and then the out of reducer gives the desired result.
Hadoop Common: These Java libraries are used to start Hadoop and are used by other Hadoop modules.

Hadoop Architecture

The Hadoop architecture is a package of the file system, MapReduce engine and the HDFS (Hadoop Distributed File System). The MapReduce engine can be MapReduce/MR1 or YARN/MR2.

A Hadoop cluster consists of a single master and multiple slave nodes. The master node includes Job Tracker, Task Tracker, NameNode, and DataNode whereas the slave node includes DataNode and TaskTracker.



Hadoop Distributed File System

The Hadoop Distributed File System (HDFS) is a distributed file system for Hadoop. It contains a master/slave architecture. This architecture consist of a single NameNode performs the role of master, and multiple DataNodes performs the role of a slave.

Both NameNode and DataNode are capable enough to run on commodity machines. The Java language is used to develop HDFS. So any machine that supports Java language can easily run the NameNode and DataNode software.

NameNode

It is a single master server exist in the HDFS cluster.
As it is a single node, it may become the reason of single point failure.
It manages the file system namespace by executing an operation like the opening, renaming and closing the files.
It simplifies the architecture of the system.

DataNode

The HDFS cluster contains multiple DataNodes.
Each DataNode contains multiple data blocks.
These data blocks are used to store data.
It is the responsibility of DataNode to read and write requests from the file system's clients.
It performs block creation, deletion, and replication upon instruction from the NameNode.

Job Tracker

The role of Job Tracker is to accept the MapReduce jobs from client and process the data by using NameNode.
In response, NameNode provides metadata to Job Tracker.

Task Tracker

It works as a slave node for Job Tracker.
It receives task and code from Job Tracker and applies that code on the file. This process can also be called as a Mapper.

MapReduce Layer

The MapReduce comes into existence when the client application submits the MapReduce job to Job Tracker. In response, the Job Tracker sends the request to the appropriate Task Trackers. Sometimes, the TaskTracker fails or time out. In such a case, that part of the job is rescheduled.

Advantages of Hadoop

Fast: In HDFS the data distributed over the cluster and are mapped which helps in faster retrieval. Even the tools to process the data are often on the same servers, thus reducing the processing time. It is able to process terabytes of data in minutes and Peta bytes in hours.

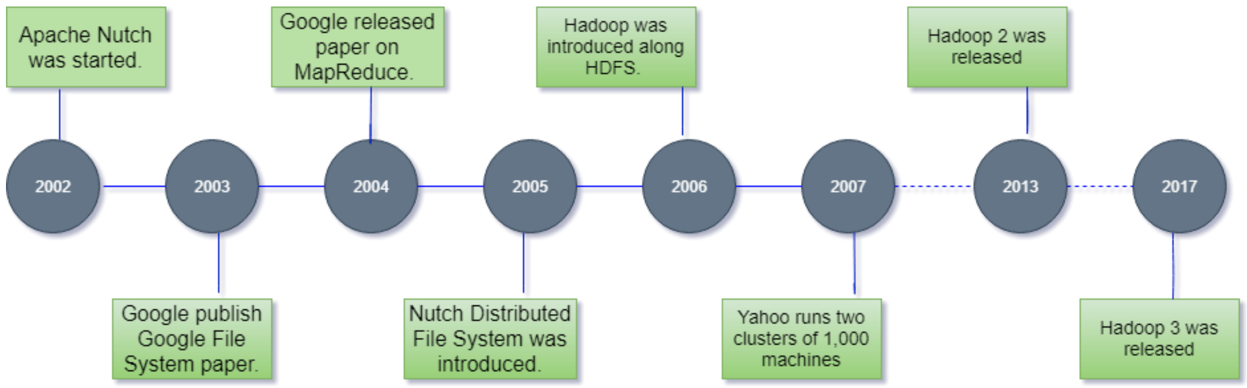
Scalable: Hadoop cluster can be extended by just adding nodes in the cluster.

Cost Effective: Hadoop is open source and uses commodity hardware to store data so it really cost effective as compared to traditional relational database management system.

Resilient to failure: HDFS has the property with which it can replicate data over the network, so if one node is down or some other network failure happens, then Hadoop takes the other copy of data and use it. Normally, data are replicated thrice but the replication factor is configurable.

History of Hadoop

The Hadoop was started by Doug Cutting and Mike Cafarella in 2002. Its origin was the Google File System paper, published by Google.


Apache Nutch was started.
Google released paper on MapReduce.
Hadoop was introduced along HDFS.
Hadoop 2 was released
2002
2003
2004
2005
2006
2007
2013
2017
Google publish Google File System paper.
Nutch Distributed File System was introduced.
Yahoo runs two clusters of 1,000 machines
Hadoop 3 was released

Let's focus on the history of Hadoop in the following steps: -

In 2002, Doug Cutting and Mike Cafarella started to work on a project, Apache Nutch. It is an open source web crawler software project.

While working on Apache Nutch, they were dealing with big data. To store that data they have to spend a lot of costs which becomes the consequence of that project. This problem becomes one of the important reason for the emergence of Hadoop.

In 2003, Google introduced a file system known as GFS (Google file system). It is a proprietary distributed file system developed to provide efficient access to data.

In 2004, Google released a white paper on Map Reduce. This technique simplifies the data processing on large clusters.

In 2005, Doug Cutting and Mike Cafarella introduced a new file system known as NDFS (Nutch Distributed File System). This file system also includes Map reduce.

In 2006, Doug Cutting quit Google and joined Yahoo. On the basis of the Nutch project, Dough Cutting introduces a new project Hadoop with a file system known as HDFS (Hadoop Distributed File System). Hadoop first version 0.1.0 released in this year.
Doug Cutting gave named his project Hadoop after his son's toy elephant.
In 2007, Yahoo runs two clusters of 1000 machines.
In 2008, Hadoop became the fastest system to sort 1 terabyte of data on a 900 node cluster within 209 seconds.
In 2013, Hadoop 2.2 was released.
In 2017, Hadoop 3.0 was released.

Year	Event
2003	Google released the paper, Google File System (GFS).
2004	Google released a white paper on Map Reduce.
2006	Hadoop introduced. Hadoop 0.1.0 released. Yahoo deploys 300 machines and within this year reaches 600 machines.
2007	Yahoo runs 2 clusters of 1000 machines. Hadoop includes HBase.
2008	YARN JIRA opened Hadoop becomes the fastest system to sort 1 terabyte of data on a 900 node cluster within 209 seconds. Yahoo clusters loaded with 10 terabytes per day. Cloudera was founded as a Hadoop distributor.
2009	Yahoo runs 17 clusters of 24,000 machines. Hadoop becomes capable enough to sort a petabyte. MapReduce and HDFS become separate subproject.
2010	Hadoop added the support for Kerberos. Hadoop operates 4,000 nodes with 40 petabytes. Apache Hive and Pig released.
2011	Apache Zookeeper released. Yahoo has 42,000 Hadoop nodes and hundreds of petabytes of storage.
2012	Apache Hadoop 1.0 version released.
2013	Apache Hadoop 2.2 version released.
2014	Apache Hadoop 2.6 version released.
2015	Apache Hadoop 2.7 version released.
2017	Apache Hadoop 3.0 version released.
2018	Apache Hadoop 3.1 version released.

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 Concepts

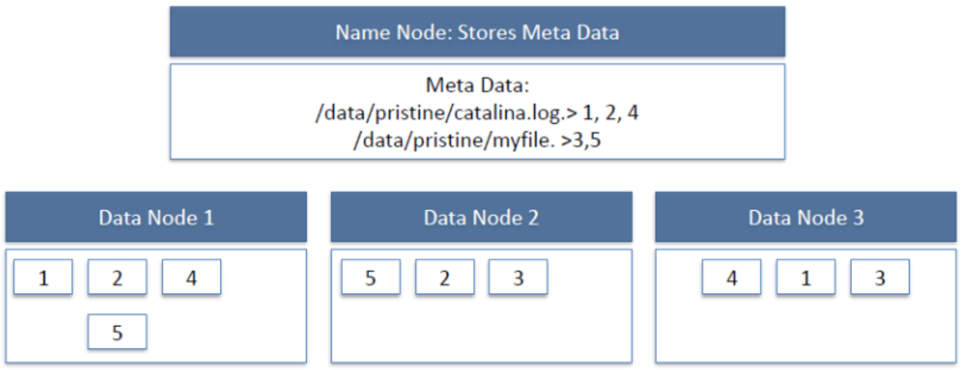
Blocks: A Block is the minimum amount of data that it can read or write. HDFS blocks are 128 MB by default and this is configurable. Files n HDFS are broken into block-sized chunks, which are stored as independent units. Unlike a file system, if the file is in HDFS is smaller than block size, then it does not occupy full blocks 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.

Name Node: 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

.

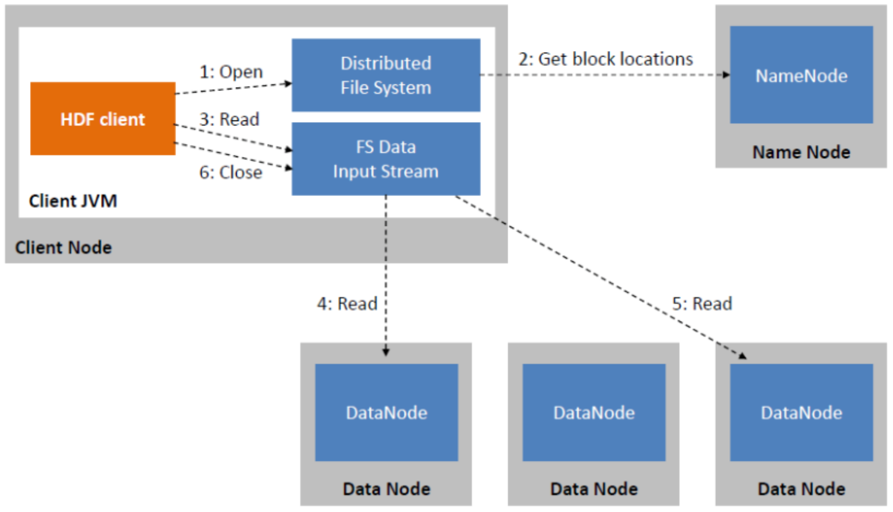
Data Node: They store and retrieve blocks when they are told to; by client or name node. They report back to name node periodically, with list of blocks that they are storing. The data node being a commodity hardware also does the work of block creation, deletion and replication as stated by the name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

HDFS DataNode and NameNode Image:


Name Node: Stores Meta Data
Meta Data:
/data/pristine/catalina.log.> 1, 2, 4
/data/pristine/myfile. >3,5
Data Node 1
1
2
4
5
Data Node 2
5
2
3
Data Node 3
4
1
3

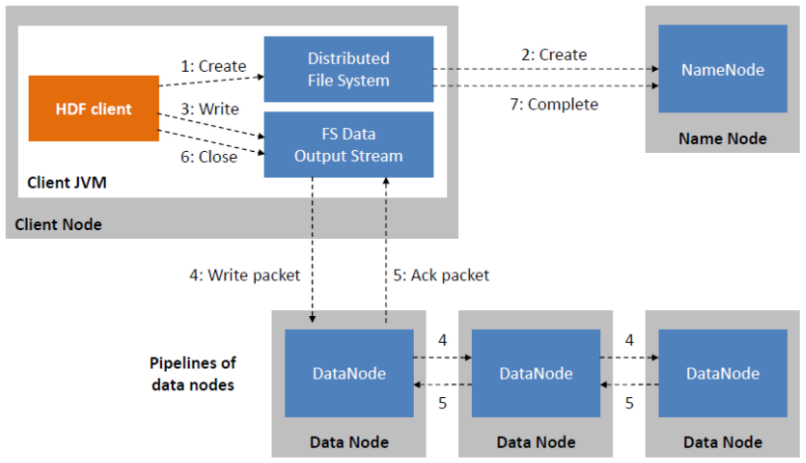
HDFS Read Image

HDFS Read Image:



HDFS Write Image

HDFS Write Image:



Since all the metadata is stored in name node, it is very important. If it fails the file system can not be used as there would be no way of knowing how to reconstruct the files from blocks present in data node. To overcome this, the concept of secondary name node arises.

Secondary Name Node: It is a separate physical machine which acts as a helper of name node. It performs periodic check points. It communicates with the name node and take snapshot of meta data which helps minimize downtime and loss of data.

Starting HDFS

The HDFS should be formatted initially and then started in the distributed mode. Commands are given below.

To Format **$ hadoop namenode -format**

To Start **$ start-dfs.sh**

HDFS Basic File Operations

1. Putting data to HDFS from local file system
    - First create a folder in HDFS where data can be put form local file system.
      $ hadoop fs -mkdir /user/test
    - Copy the file "data.txt" from a file kept in local folder /usr/home/Desktop to HDFS folder /user/test
      $ hadoop fs -copyFromLocal /usr/home/Desktop/data.txt /user/test
    - Display the content of HDFS folder
      $ Hadoop fs -ls /user/test
2. Copying data from HDFS to local file system
    - $ hadoop fs -copyToLocal /user/test/data.txt /usr/bin/data_copy.txt
3. Compare the files and see that both are same
    - $ md5 /usr/bin/data_copy.txt /usr/home/Desktop/data.txt

   Recursive deleting
    - hadoop fs -rmr <arg>

   Example:
    - hadoop fs -rmr /user/sonoo/
    -

HDFS Other commands

The below is used in the commands
"<path>" means any file or directory name.
"<path>..." means one or more file or directory names.
"<file>" means any filename.
"<src>" and "<dest>" are path names in a directed operation.
"<localSrc>" and "<localDest>" are paths as above, but on the local file system

- put <localSrc><dest>
  Copies the file or directory from the local file system identified by localSrc to dest within the DFS.
- copyFromLocal <localSrc><dest>
  Identical to -put
- copyFromLocal <localSrc><dest>

Identical to -put

- moveFromLocal <localSrc><dest>
  Copies the file or directory from the local file system identified by localSrc to dest within HDFS, and then deletes the local copy on success.
- get [-crc] <src><localDest>
  Copies the file or directory in HDFS identified by src to the local file system path identified by localDest.
- cat <filen-ame>
  Displays the contents of filename on stdout.
- moveToLocal <src><localDest>
  Works like -get, but deletes the HDFS copy on success.
- setrep [-R] [-w] rep <path>
  Sets the target replication factor for files identified by path to rep. (The actual replication factor will move toward the target over time)
- touchz <path>
  Creates a file at path containing the current time as a timestamp. Fails if a file already exists at path, unless the file is already size 0.
- test -[ezd] <path>
  Returns 1 if path exists; has zero length; or is a directory or 0 otherwise.
- stat [format] <path>
  Prints information about path. Format is a string which accepts file size in blocks (%b), filename (%n), block size (%o), replication (%r), and modification date (%y, %Y).

HDFS Features and Goals

The Hadoop Distributed File System (HDFS) is a distributed file system. It is a core part of Hadoop which is used for data storage. It is designed to run on commodity hardware.

Unlike other distributed file system, HDFS is highly fault-tolerant and can be deployed on low-cost hardware. It can easily handle the application that contains large data sets.

Features of HDFS

Highly Scalable - HDFS is highly scalable as it can scale hundreds of nodes in a single cluster.
Replication - Due to some unfavorable conditions, the node containing the data may be loss. So, to overcome such problems, HDFS always maintains the copy of data on a different machine.
Fault tolerance - In HDFS, the fault tolerance signifies the robustness of the system in the event of failure. The HDFS is highly fault-tolerant that if any machine fails, the other machine containing the copy of that data automatically become active.
Distributed data storage - This is one of the most important features of HDFS that makes Hadoop very powerful. Here, data is divided into multiple blocks and stored into nodes.
Portable - HDFS is designed in such a way that it can easily portable from platform to another.

Goals of HDFS

Handling the hardware failure - The HDFS contains multiple server machines. Anyhow, if any machine fails, the HDFS goal is to recover it quickly.

Streaming data access - The HDFS applications usually run on the general-purpose file system. This application requires streaming access to their data sets.
Coherence Model - The application that runs on HDFS require to follow the write-once-ready-many approach. So, a file once created need not to be changed. However, it can be appended and truncate.

YARN

Yet Another Resource Manager takes programming to the next level beyond Java , and makes it interactive to let another application Hbase, Spark etc. to work on it.Different Yarn applications can co-exist on the same cluster so MapReduce, Hbase, Spark all can run at the same time bringing great benefits for manageability and cluster utilization.

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.

Jobtracker & Tasktrackerwere were used in previous version of Hadoop, which were responsible for handling resources and checking progress management. However, Hadoop 2.0 has Resource manager and NodeManager to overcome the shortfall of Jobtracker & Tasktracker.

Benefits of YARN

Scalability: Map Reduce 1 hits ascalability bottleneck at 4000 nodes and 40000 task, but Yarn is designed for 10,000 nodes and 1 lakh tasks.

Utiliazation: 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.

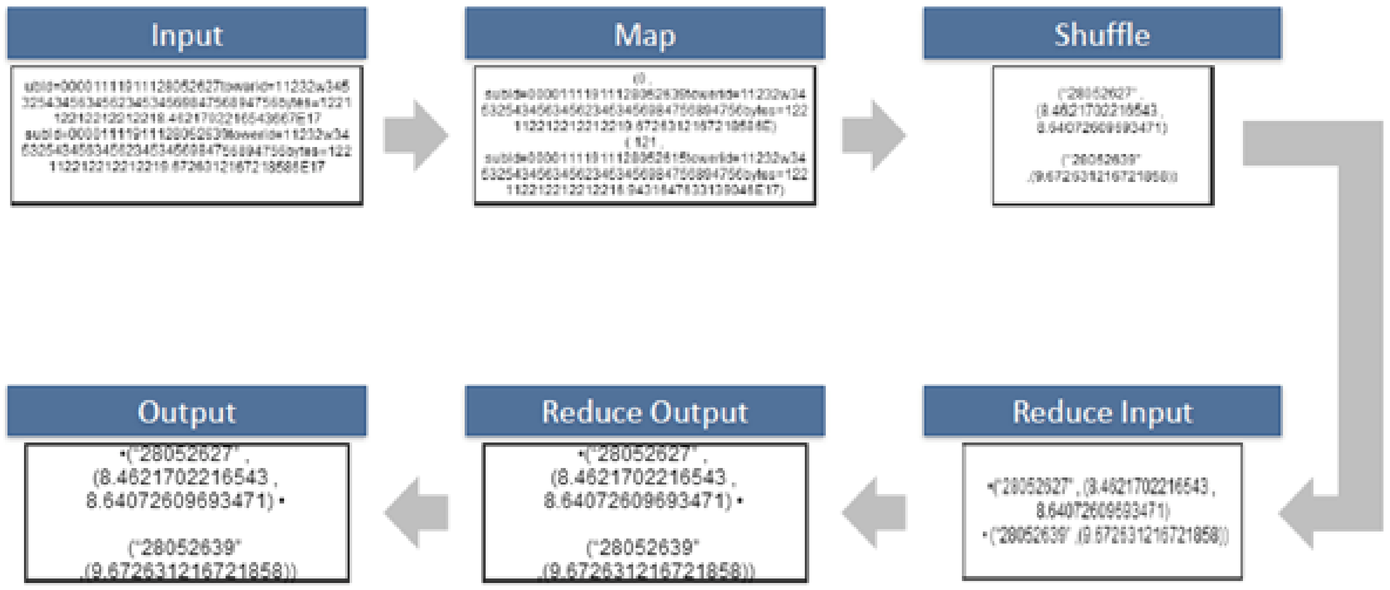
MapReduce

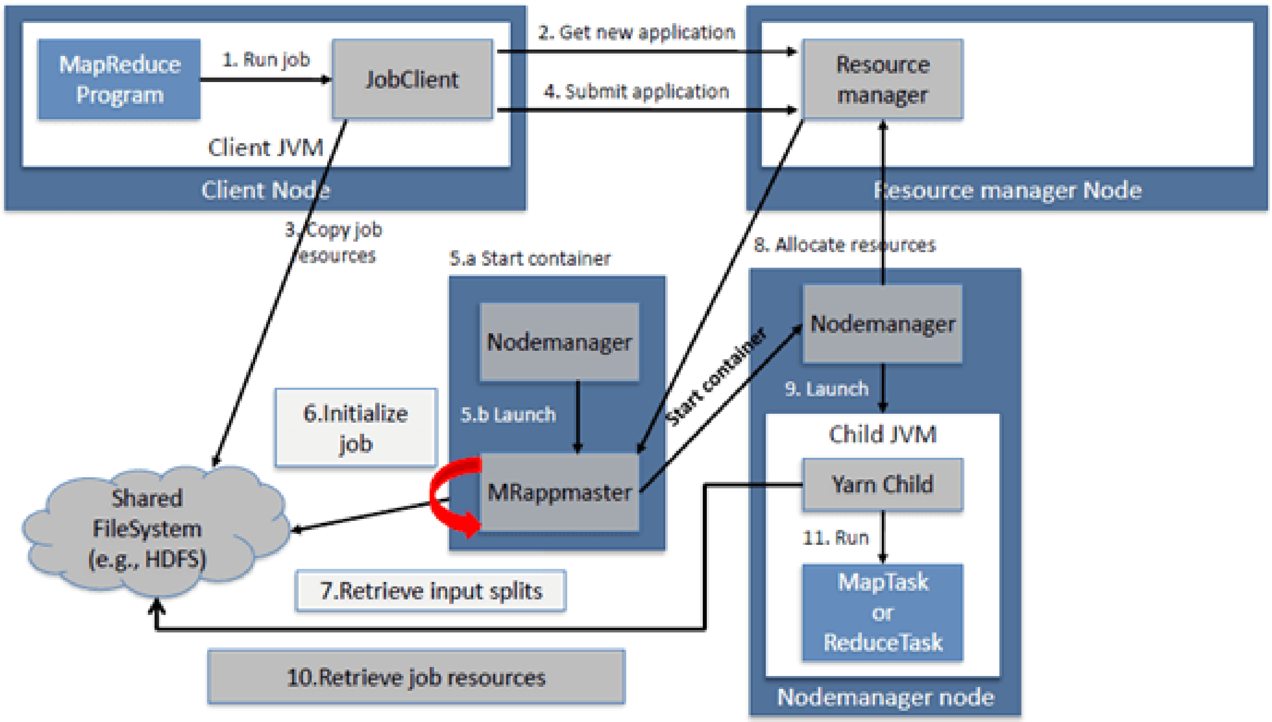
A MapReduce is a data processing tool which is used to process the data parallelly in a distributed form. It was developed in 2004, on the basis of paper titled as "MapReduce: Simplified Data Processing on Large Clusters," published by Google.

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.

Steps in Map Reduce

1. 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.
2. 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)>.
3.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.





<u>Sort and Shuffle</u>

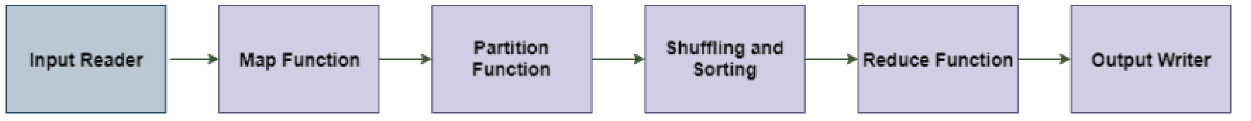
The sort and shuffle occur on the output of Mapper and before the reducer. When the Mapper task is complete, the results are sorted by key, partitioned if there are multiple reducers, and then written to disk. Using the input from each Mapper <k2,v2>, we collect all the values for each unique key k2. This output from the shuffle phase in the form of <k2, list(v2)> is sent as input to reducer phase.

<u>Usage of MapReduce</u>

- 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.

Data Flow In MapReduce

MapReduce is used to compute the huge amount of data . To handle the upcoming data in a parallel and distributed form, the data has to flow from various phases.



Input reader

The input reader reads the upcoming data and splits it into the data blocks of the appropriate size (64 MB to 128 MB). Each data block is associated with a Map function.
Once input reads the data, it generates the corresponding key-value pairs. The input files reside in HDFS. Input data can be in any form.

Map function

The map function process the upcoming key-value pairs and generated the corresponding output key-value pairs. The map input and output type may be different from each other.

Partition function

The partition function assigns the output of each Map function to the appropriate reducer. The available key and value provide this function. It returns the index of reducers.

Shuffling and Sorting

The data are shuffled between/within nodes so that it moves out from the map and get ready to process for reduce function. Sometimes, the shuffling of data can take much computation time.

The sorting operation is performed on input data for Reduce function. Here, the data is compared using comparison function and arranged in a sorted form.

Reduce function

The Reduce function is assigned to each unique key. These keys are already arranged in sorted order. The values associated with the keys can iterate the Reduce and generates the corresponding output.

Output writer

Once the data flow from all the above phases, Output writer executes. The role of Output writer is to write the Reduce output to the stable storage.

MapReduce API

MapReduce Mapper Class

In MapReduce, the role of the Mapper class is to map the input key-value pairs to a set of intermediate key-value pairs. It transforms the input records into intermediate records.
These intermediate records associated with a given output key and passed to Reducer for the final output.

Methods of Mapper Class

void cleanup(Context context)	This method called only once at the end of the task.
void map(KEYIN key, VALUEIN value, Context context)	This method can be called only once for each key-value in the input split.
void run(Context context)	This method can be override to control the execution of the Mapper.
void setup(Context context)	This method called only once at the beginning of the task.

MapReduce Reducer Class

In MapReduce, the role of the Reducer class is to reduce the set of intermediate values. Its implementations can access the Configuration for the job via the JobContext.getConfiguration() method.

Methods of Reducer Class

void cleanup(Context context)	This method called only once at the end of the task.
void map(KEYIN key, Iterable<VALUEIN> values, Context context)	This method called only once for each key.
void run(Context context)	This method can be used to control the tasks of the Reducer.
void setup(Context context)	This method called only once at the beginning of the task.

MapReduce Job Class

The Job class is used to configure the job and submits it. It also controls the execution and query the state. Once the job is submitted, the set method throws IllegalStateException.

Methods of Job Class

Methods	Description
Counters getCounters()	This method is used to get the counters for the job.
long getFinishTime()	This method is used to get the finish time for the job.
Job getInstance()	This method is used to generate a new Job without any cluster.
Job getInstance(Configuration conf)	This method is used to generate a new Job without any cluster and provided configuration.
Job getInstance(Configuration conf, String jobName)	This method is used to generate a new Job without any cluster and provided configuration and job name.
String getJobFile()	This method is used to get the path of the submitted job configuration.
String getJobName()	This method is used to get the user-specified job name.
JobPriority getPriority()	This method is used to get the scheduling function of the job.
void setJarByClass(Class<?> c)	This method is used to set the jar by providing the class name with .class extension.
void setJobName(String name)	This method is used to set the user-specified job name.
void setMapOutputKeyClass(Class<?> class)	This method is used to set the key class for the map output data.
void setMapOutputValueClass(Class<?> class)	This method is used to set the value class for the map output data.
void setMapperClass(Class<? extends Mapper> class)	This method is used to set the Mapper for the job.
void setNumReduceTasks(int tasks)	This method is used to set the number of reduce tasks for the job
void setReducerClass(Class<? extends	This method is used to set the Reducer for the job.

Reducer> class)	

HIVE

Hive is a data warehouse system which is used to analyze structured data. It is built on the top of Hadoop. It was developed by Facebook.

Hive provides the functionality of reading, writing, and managing large datasets residing in distributed storage. It runs SQL like queries called HQL (Hive query language) which gets internally converted to MapReduce jobs.

Using Hive, we can skip the requirement of the traditional approach of writing complex MapReduce programs. Hive supports Data Definition Language (DDL), Data Manipulation Language (DML), and User Defined Functions (UDF).

Features of Hive

- Hive is fast and scalable.
- It provides SQL-like queries (i.e., HQL) that are implicitly transformed to MapReduce or Spark jobs.
- It is capable of analyzing large datasets stored in HDFS.
- It allows different storage types such as plain text, RCFile, and HBase.
- It uses indexing to accelerate queries.
- It can operate on compressed data stored in the Hadoop ecosystem.
- It supports user-defined functions (UDFs) where user can provide its functionality.

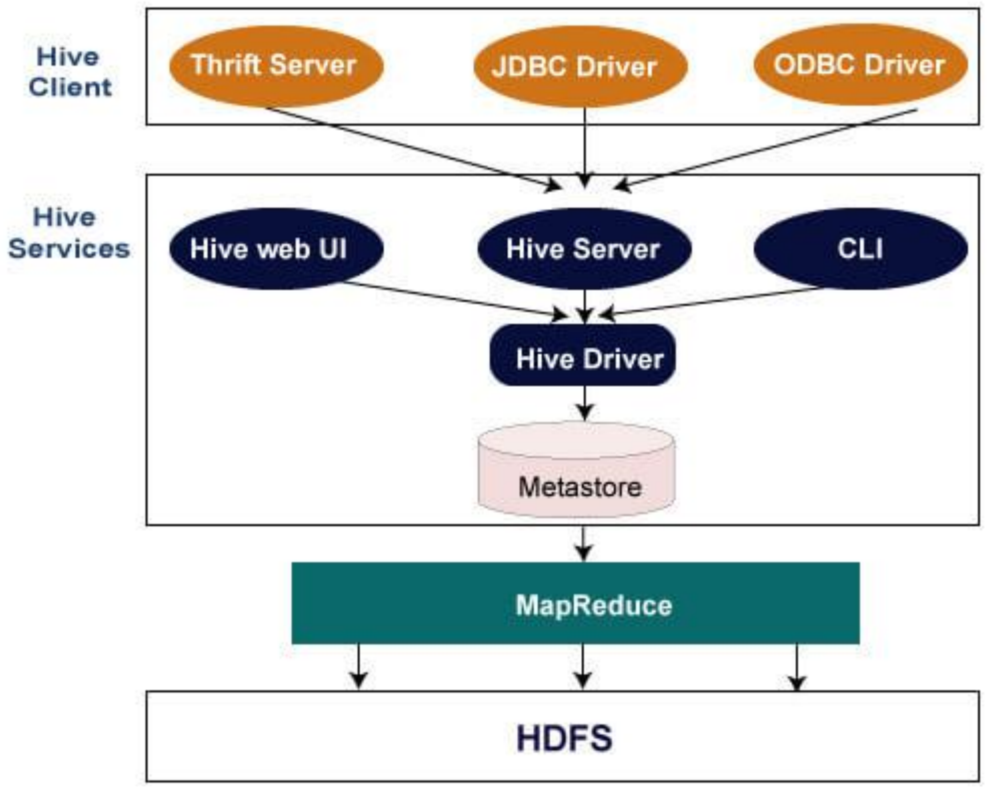
Limitations of Hive

- Hive is not capable of handling real-time data.
- It is not designed for online transaction processing.
- Hive queries contain high latency.

Hive	**Pig**
Hive is commonly used by Data Analysts.	Pig is commonly used by programmers.
It follows SQL-like queries.	It follows the data-flow language.
It can handle structured data.	It can handle semi-structured data.
It works on server-side of HDFS cluster.	It works on client-side of HDFS cluster.
Hive is slower than Pig.	Pig is comparatively faster than Hive.

Hive Architecture

The following architecture explains the flow of submission of query into Hive.



Hive Client

Hive allows writing applications in various languages, including Java, Python, and C++. It supports different types of clients such as:-

Thrift Server - It is a cross-language service provider platform that serves the request from all those programming languages that supports Thrift.
JDBC Driver - It is used to establish a connection between hive and Java applications. The JDBC Driver is present in the class org.apache.hadoop.hive.jdbc.HiveDriver.
ODBC Driver - It allows the applications that support the ODBC protocol to connect to Hive.

Hive Services

Hive CLI - The Hive CLI (Command Line Interface) is a shell where we can execute Hive queries and commands.

Hive Web User Interface - The Hive Web UI is just an alternative of Hive CLI. It provides a web-based GUI for executing Hive queries and commands.
Hive MetaStore - It is a central repository that stores all the structure information of various tables and partitions in the warehouse. It also includes metadata of column and its type information, the serializers and deserializers which is used to read and write data and the corresponding HDFS files where the data is stored.
Hive Server - It is referred to as Apache Thrift Server. It accepts the request from different clients and provides it to Hive Driver.
Hive Driver - It receives queries from different sources like web UI, CLI, Thrift, and JDBC/ODBC driver. It transfers the queries to the compiler.
Hive Compiler - The purpose of the compiler is to parse the query and perform semantic analysis on the different query blocks and expressions. It converts HiveQL statements into MapReduce jobs.
Hive Execution Engine - Optimizer generates the logical plan in the form of DAG of map-reduce tasks and HDFS tasks. In the end, the execution engine executes the incoming tasks in the order of their dependencies.

Apache Pig

Apache Pig is a high-level data flow platform for executing MapReduce programs of Hadoop. The language used for Pig is Pig Latin.

The Pig scripts get internally converted to Map Reduce jobs and get executed on data stored in HDFS. Apart from that, Pig can also execute its job in Apache Tez or Apache Spark.

Pig can handle any type of data, i.e., structured, semi-structured or unstructured and stores the corresponding results into Hadoop Data File System. Every task which can be achieved using Pig can also be achieved using java used in MapReduce.

Features of Apache Pig

Ease of programming : Writing complex java programs for map reduce is quite tough for non-programmers. Pig makes this process easy. In the Pig, the queries are converted to MapReduce internally.

Optimization opportunities: It is how tasks are encoded permits the system to optimize their execution automatically, allowing the user to focus on semantics rather than efficiency.

Extensibility:A user-defined function is written in which the user can write their logic to execute over the data set.

Flexible: It can easily handle structured as well as unstructured data.

In-built operators: It contains various type of operators such as sort, filter and joins.

Differences between Apache MapReduce and PIG

Apache MapReduce	Apache PIG
It is a low-level data processing tool.	It is a high-level data flow tool.
Here, it is required to develop complex programs using Java or Python.	It is not required to develop complex programs.
It is difficult to perform data operations in MapReduce.	It provides built-in operators to perform data operations like union, sorting and ordering.
It doesn't allow nested data types.	It provides nested data types like tuple, bag, and map.

Advantages of Apache Pig

Less code - The Pig consumes less line of code to perform any operation.

Reusability - The Pig code is flexible enough to reuse again.
Nested data types - The Pig provides a useful concept of nested data types like tuple, bag, and map.