UNIT-2

Hadoop and HDFS

UNIT-II

Hadoop and HDFS

Syllabus

Introduction to Hadoop, Brief history of Hadoop, Apache Hadoop eco system, The design of Hadoop Distributed File System (HDFS), Architecture, Building blocks of Hadoop: Namenode, Datanode, Secondary Namenode, Job Tracker, Task Tracker, Basic File System Operations.

2.1 Introduction to Hadoop

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.

Characteristics:

- ✓ Open source
- ✓ Distributed processing
- ✓ Distributed storage
- ✓ Scalable
- ✓ Reliable
- ✓ Fault-tolerant
- ✓ Economical
- ✓ Flexible

- ➢ Originally built as a Infrastructure for the “Nutch” project.
- ➢ Based on Google’s map reduce and Google File System.
- ➢ Created by Doug Cutting in 2005 at Yahoo Named after his son’s toy yellow elephant.
- ➢ Written in Java

2.2 Brief History of Hadoop

Apache Hadoop is an open-source software framework for storage and large-scale processing of data-sets on clusters of commodity hardware.

It gives companies the capability to gather, store and analyze huge sets of data.

2002 – Nutch an open source web search engine started. This architecture wouldn't scale to the billions of pages on the Web.

2003 – Google published a paper that describes the architecture of Google's distributed filesystem, called GFS, which was being used in production at Google, would solve their storage needs for the very large files generated as a part of the web crawl and indexing process.

2004 – writing an open source implementation, the Nutch Distributed File system(NDFS).

In the same year Google published the paper that introduced MapReduce to the world.

Early 2005 - the Nutch developers had a working MapReduce implementation in Nutch, and by the middle of that year all the major Nutch algorithms had been ported to run using MapReduce and NDFS.

2006 - Nutch to form an independent subproject of Lucene called Hadoop. At around the same time, Doug Cutting joined Yahoo!, which provided a dedicated team and the resources to turn Hadoop into a system that ran at web scale

2008 - Yahoo! announced that its production search index was being generated by a 10,000-core Hadoop cluster. January 2008, Hadoop was made its own top-level project at Apache.

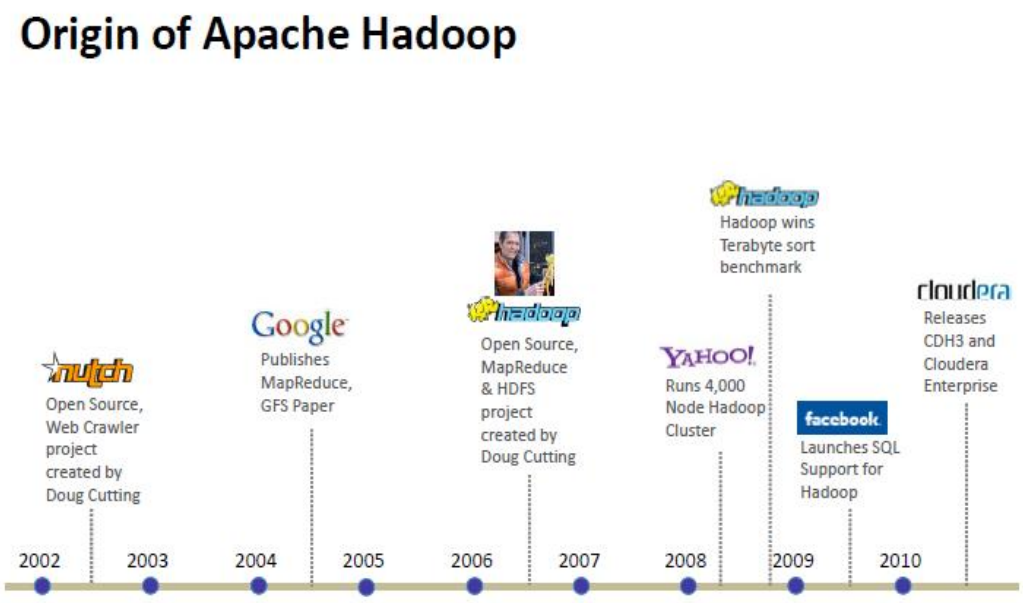
April 2008 - Hadoop broke a world record to become the fastest system to sort a terabyte of data. Running on a 910-node cluster, Hadoop sorted one terabyte in 209 seconds (just under 3½ minutes), beating the previous year's winner of 297 seconds.

November 2008 - Google reported that its MapReduce implementation sorted one terabyte in 68 seconds.

May 2009 - A team at Yahoo! used Hadoop to sort one terabyte in 62 seconds. Hadoop's is a general-purpose storage and analysis platform for big data .

Hadoop used by many companies Last.fm, FaceBook and New York Times etc.,

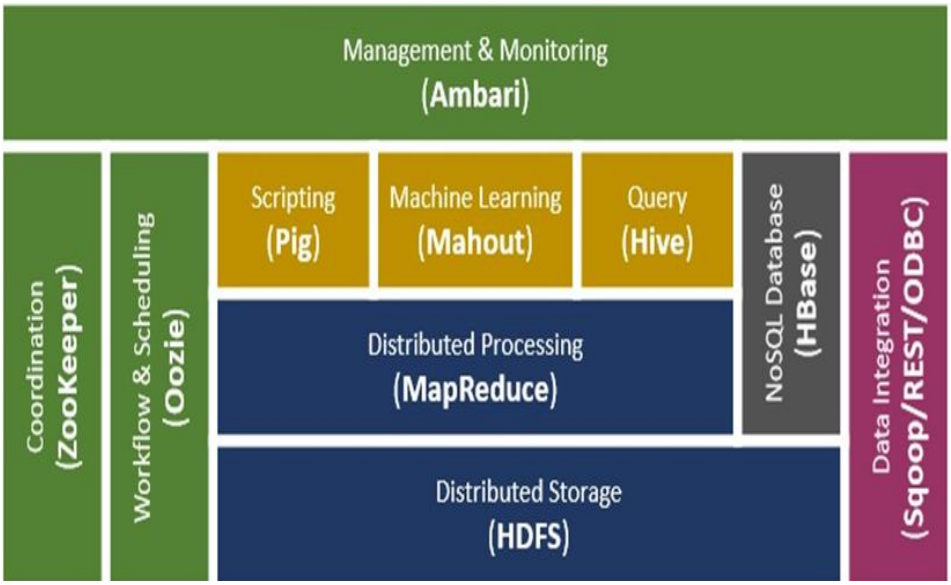
Hadoop distributios from the large, established enterprise vendors EMC, IBM, Microsoft, and Oralce. Specialist Hadoop companies are Cloudera, Hortonworks and MapR.



2.3 Apache Hadoop Eco System

The term eco system is also used for a family of related projects that fall under the umbrella of infrastructure for distributed computing and largescale data processing.

All of the core projects are hosted by the Apache Software Foundation, which provides support for a community of open source software projects, including the original HTTP Server from which it gets its name.



Hadoop ecosystem grows, more projects are appearing, not necessarily hosted at Apache, which provide complementary services to Hadoop, or build on the core to add higher-level abstractions.

The Hadoop projects

Common

A set of components and interfaces for distributed file systems and general I/O (serialization,Java RPC, persistent data structures).

Avro

A serialization system for efficient, cross-language RPC, and persistent Data storage.

MapReduce

A distributed data processing model and execution environment that runs on large clusters ofcommodity machines.

HDFS

A distributed file system that runs on large clusters of commodity machines.

Pig

A data flow language and execution environment for exploring very large datasets. Pig runs on HDFS and MapReduce clusters.

Hive

A distributed data warehouse. Hive manages data stored in HDFS and provides a query language based on SQL (and which is translated by the runtime engine to MapReduce jobs) for querying the data.

HBase

A distributed, column-oriented database. HBase uses HDFS for its underlying storage, andsupports both batch-style computations using MapReduce and point queries (random reads).

ZooKeeper

A distributed, highly available coordination service. ZooKeeper provides primitives such asdistributed locks that can be used for building distributed applications.

Sqoop

A tool for efficiently moving data between Relational Databases and HDFS.

Oozie

A service for running and scheduling workflows of Hadoop jobs(including Map-Reduce,Pig, Hive and Sqoop jobs.

Hadoop Architecture

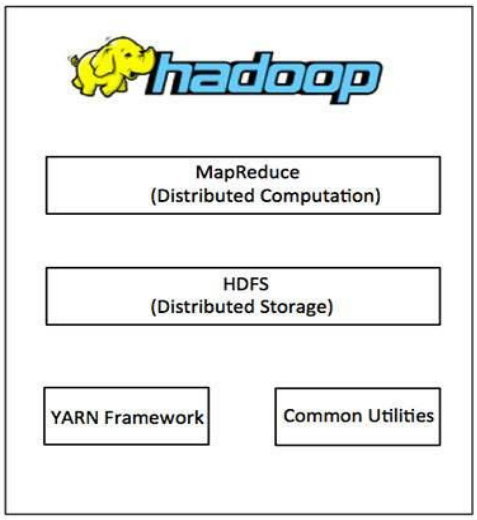
Hadoop framework includes following four modules:

1) **Hadoop Common:** A set of components (libraries,utilities) and interfaces for DFS and general I/O. These libraries provides filesystem and OS level abstractions and contains the necessary Java files and scripts required to start Hadoop.

2) **Hadoop YARN**: This is a framework for job scheduling and cluster resource management.

3) **Hadoop Distributed File System (HDFS):** A distributed file system.

4) **Hadoop MapReduce:** This is YARN-based system for parallel processing of large data sets.



2.4 The Design of Hadoop Distributed File System (HDFS)

HDFS is a filesystem designed for storing very large files with streaming data access patterns, running on clusters of commodity hardware.

✓ **Very large files**

"Very large" in this context means files that are hundreds of megabytes gigabytes, or terabytes in size. There are Hadoop clusters running today that store petabytes of data.

✓ **Streaming data access**

HDFS is built around the idea that the most efficient data processing pattern is a write-once, read-many-times pattern. A dataset is typically generated or copied from source, and then various analyses are performed on that dataset over time.

✓ **Commodity hardware**

Hadoop doesn't require expensive, highly reliable hardware. It's designed to run on clusters of commodity hardware (commonly available hardware that can be obtained from multiple vendors) for which the chance of node failure across the cluster is high, at least for large clusters. HDFS is designed to carry on working without a noticeable interruption to the user in the face of such failure.

HDFS does not work well for some areas

✓ **Low-latency data access**

Applications that require low-latency access to data, in the tens of milliseconds range, will not work well with HDFS. Remember, HDFS is optimized for delivering a high throughput of data, and this may be at the expense of latency.

✓ **Lots of small files**

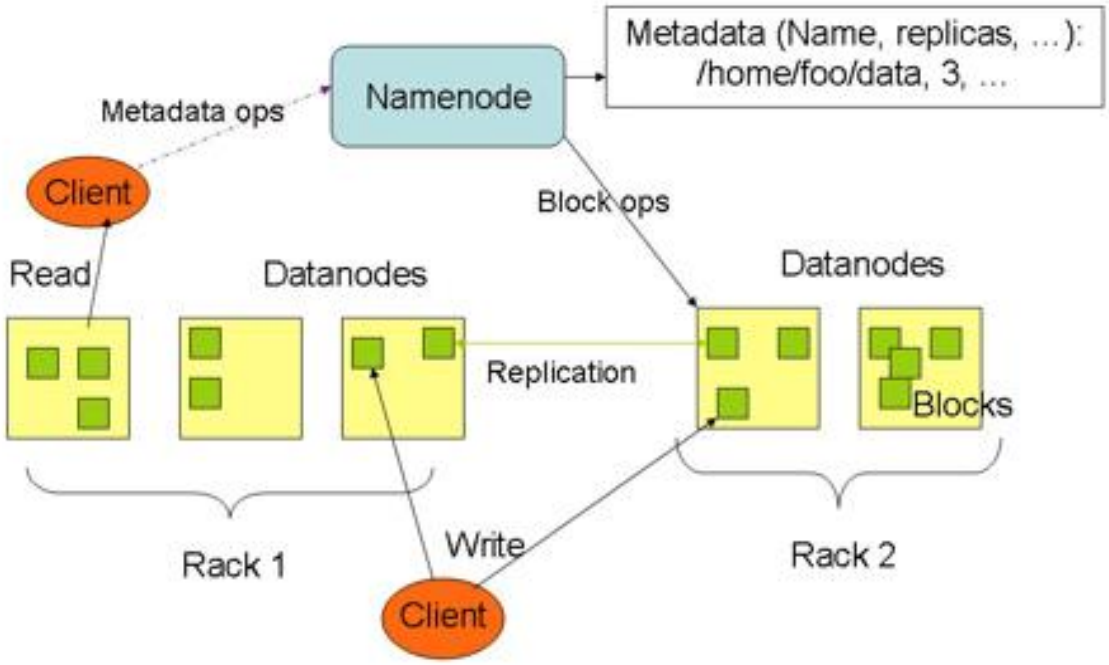
The namenode holds file system metadata in memory, the limit to the number of files in a file system is governed by the amount of memory on the namenode. As a rule of thumb, each file, directory, and block takes about 150 bytes.

✓ **Multiple writers, arbitrary file modifications**

Files in HDFS may be written to by a single writer. Writes are always made at the end of the file, in append-only fashion. There is no support for multiple writers, or for modifications at arbitrary offsets in the file.

2.5 HDFS Architecture

- The Hadoop Distributed File System (HDFS) architecture is designed to store large datasets across multiple machines in a distributed environment.
- **HDFS is Master/slave architecture.** This architecture provides high fault tolerance and scalability, which is essential for big data processing.
- An HDFS cluster has two types of nodes,a *namenode* (the master) and a number of *datanodes* (slaves). There is a single NameNode for each cluster. The cluster can have thousands of DataNodes and tens of thousands of HDFS clients per cluster.



Key Components of HDFS Architecture:

1. **NameNode (Master):**
   - **Role:** The NameNode is the centerpiece of HDFS. It manages the **metadata** of the file system, such as the directory structure, file locations, and permissions. It does not store the actual data but maintains a mapping of files to the DataNodes.
   - **Responsibilities:**
     - Manages the namespace of HDFS.
     - Keeps track of file blocks and their locations across the cluster.
     - Handles client requests for file operations (create, delete, read, etc.).
     - Stores metadata in two files: **fsimage** (file system image) and **edit logs** (log of file system changes).

The Negative aspect of Name node is single point of failure. If the Name node is failed we can't access the files stored in HDFS.

2. **DataNodes (Slaves):**
   - **Role:** DataNodes store the actual data in the form of blocks. Each file is divided into blocks (typically 128 MB), which are replicated across multiple DataNodes for fault tolerance.
   - **Responsibilities:**
     - Store and manage the data blocks.
     - Periodically send block reports to the NameNode with information about which blocks are stored.
     - Serve read and write requests from clients.
     - Perform replication, deletion, and block creation as instructed by the NameNode.
3. **Secondary NameNode:**
   - **Role:** Secondary NameNode assists the NameNode by periodically merging the **fsimage** and **edit logs**. It is not a failover node but helps in reducing the size of the edit logs for efficient NameNode recovery in case of failure.
4. **Clients:**
   - **Role:** Clients interact with HDFS to read and write files. They communicate with the NameNode for metadata operations (e.g., finding which DataNodes hold the data) and with DataNodes to read/write the actual data.
5. **HDFS Blocks:**
   - Files are split into fixed-size blocks (usually 128 MB by default), the minimum amount of data that can read or write
   - Blocks are replicated across multiple DataNodes for fault tolerance (default replication factor is 3).

HDFS Architecture Workflow:

1. **File Write Operation:**
   - When a client wants to store a file, it sends a request to the **NameNode**.
   - The **NameNode** splits the file into blocks and assigns DataNodes to store these blocks.
   - The file is then written to the designated DataNodes in parallel.
   - Each block is replicated on multiple DataNodes for fault tolerance based on the replication factor.

2. **File Read Operation:**
    - When a client wants to read a file, it sends a request to the **NameNode**.
    - The **NameNode** provides the addresses of the DataNodes that store the requested blocks.
    - The client retrieves the blocks directly from the DataNodes, which are reconstructed into the original file.
3. **Fault Tolerance:**
    - If a DataNode fails, the NameNode can use the replicated blocks from other DataNodes to serve client requests.
    - The NameNode monitors the health of DataNodes and initiates block replication if a failure is detected.
4. **Block Replication:**
    - HDFS ensures fault tolerance by replicating blocks across multiple DataNodes.
    - The default replication factor is 3, meaning each block is stored on three different DataNodes. This ensures that if one or even two DataNodes fail, the data remains accessible.

2.6 The Building Blocks of Hadoop

Hadoop employs a master/slave architecture for both distributed storage and distributed computation. The distributed storage system is called the Hadoop Distributed File System (HDFS).

On a fully configured cluster, "running Hadoop" means running a set of daemons, or resident programs, on the different servers in you network. These daemons have specific roles; some exists only on one server, some exist across multiple servers.

The daemons include:

1. NameNode
2. DataNode
3. Secondary NameNode
4. JobTracker
5. TaskTracker

1) NameNode:

- An HDFS cluster has two types of nodes operating in a master-slave/worker pattern:

1) N*amenode* (the master)

2) A number of *datanodes (slaves/workers).*

Namenode tasks

- Manages the filesystem namespace.
- Maintains the filesystem tree and the metadata for all the files and directories in the tree.
- This information is stored persistently on the local disk in the form of two files: the namespace image and the edit log.
- The namenode also knows the datanodes on which all the blocks for a given file are located, however, it does not store block locations persistently, since this information is reconstructed from datanodes when the system starts.
- The user code does not need to know about the namenode and datanode to function.
- The NameNode is the bookkeeper of HDFS, it keeps track of
    1) How the files are broken down into file blocks.
    2) No. of blocks for each file, filepath
    3) Which Data Nodes store those blocks.
    4) Address of Data Node where exactly each block is present
- The overall health of the distributed filesystem.
- The function of the NameNode is memory and I/O intensive.
- The server hosting the NameNode typically doesn't store any user data or perform any computations for a MapReduce program to lower the workload on the machine.
- A negative aspect to the importance of the NameNode— it's a single point of failure of the Hadoop cluster.
- For any of the other daemons, if their host nodes fail for software or hardware reasons, the Hadoop cluster will likely continue to function smoothly or you can quickly restart it. Not so for the NameNode.

2) Data Node:

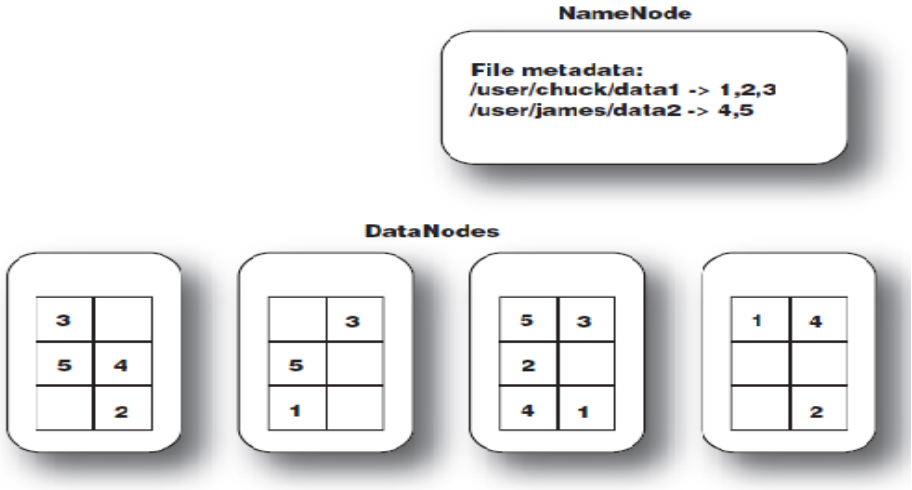
- Datanodes are the workhorses of the filesystem.
- DNs store and retrieve blocks when they are told to (by Clients or the Namenode)
- Periodically report to the Name node with lists of blocks that they are storing.

- Reading and writing HDFS blocks to actual files on the local file system.

To read or write a HDFS file

- The file is broken into blocks and the NameNode will tell your client which DataNode each block resides in.
- client communicates directly with the DataNode daemons to process the local files corresponding to the blocks.
- DataNode may communicate with other DataNodes to replicate its data blocks for redundancy.

The building blocks of Hadoop



NameNode/DataNode interaction in HDFS. The NameNode keeps track of the file metadata—which files are in the system and how each file is broken down into blocks. The DataNodes provide backup store of the blocks and constantly report to the NameNode to keep the metadata current.

- Two data files, one at /user/chuck/data1 and another at /user/james/data2.
- The data1 file takes up three blocks, which we denote 1, 2, and 3, and the data2 file consists of blocks 4 and 5.
- The content of the files are distributed among the DataNodes, each block has three replicas.
- If any one DataNode crashes or becomes inaccessible over the network, still be able to read the files.
- DataNodes are constantly reporting to the NameNode.
- At the time of initialization, each DataNodes informs the NameNode of the blocks it's currently storing.
- After the mapping is complete, the DataNodes continually poll the NameNode to

provide information regarding local changes as well as receive instructions to create, move, or delete blocks from the local disk.

3) Secondary NameNode (SNN)

- The Secondary NameNode (SNN) is an assistant daemon for monitoring the state of the cluster HDFS.
- Like the NameNode, each cluster has one SNN, and it typically resides on its own machine as well. No other DataNode or TaskTracker daemons run on the same server.
- Differences between NameNode and SNN is it doesn't receive or record any real-time changes to HDFS.
- It communicates with the NameNode to take snapshots of the HDFS metadata at intervals defined by the cluster configuration.
- When NN failed the SNN snapshots help to minimize the downtime and loss of data.
- NameNode failure requires human intervention to reconfigure the cluster to use the SNN as the primary NameNode

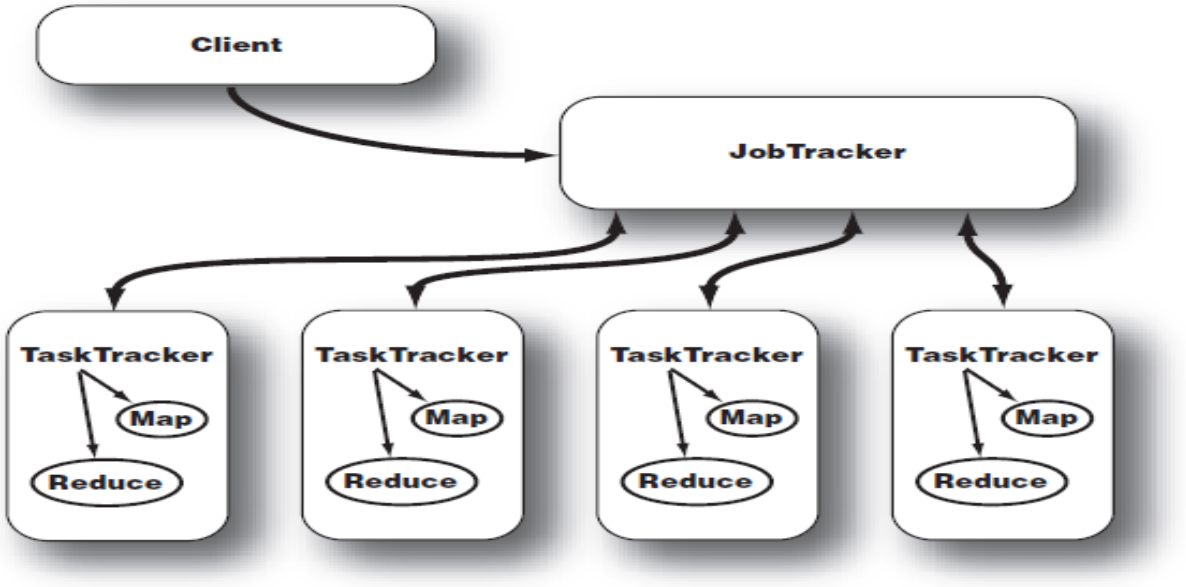
4) Job Tracker

- The JobTracker daemon is the liaison between your application and Hadoop.
- When the code is submitted to the cluster, the JobTracker determines the execution plan by
  - Determining which files to process,
  - Assigns nodes to different tasks
  - Monitors all tasks as they're running.
- When a task fail, the JobTracker will automatically relaunch the task on a different node.
- There is only one JobTracker daemon per Hadoop cluster. It is run on a server as a master node of the cluster.

5) TaskTracker:

- Similar to storage daemons, the computing daemons also follow a master/slave architecture:
- The JobTracker is the master overseeing the overall execution of a MapReduce Job .

- The TaskTrackers manage the execution of individual tasks on each slave node.



JobTracker and TaskTracker interaction. After a client calls the JobTracker to begin a data processing job, the JobTracker partitions the work and assigns different map and reduce tasks to each TaskTracker in the cluster.

- Each TaskTracker is responsible for executing the individual tasks that the JobTracker assigns.
- Although there is a single TaskTracker per slave node, each TaskTracker can spawn multiple JVMs to handle many map or reduce tasks in parallel.
- The responsibility of the TaskTracker is to constantly communicate with the JobTracker.
- If the JobTracker fails to receive a heartbeat from a TaskTracker within a specified amount of time, it will assume the TaskTracker has crashed and will resubmit the corresponding tasks to other nodes in the cluster.

2.7 Basic File System Operations

The filesystem is ready to be used, and we can do all of the usual filesystem operations, such as reading files, creating directories, moving files, deleting data, and listing directories.

Type hadoop fs -help to get detailed help on every command.

a) Creating directories:

mkdir : Create a directory in HDFS at given path(s).

Syntax : hadoop fs -mkdir <paths>

Example:

hadoop fs -mkdir /user/cloudera/myfolder1 (absolute path)

Or

hadoop fs –mkdir myfolder1 (relative path)

b) copying a file from the local filesystem to HDFS:

copyFromLocal or put copying a file from the local filesystem to HDFS

Syntax : hadoop fs – copyFromLocal <source file > <destination path>

Example:

hadoop fs -copyFromLocal file1,txt /user/cloudera

hadoop fs -put file2 . (put the file in the default directory)

c) Retriving files from HDFS to local filesystem

copyToLocal or get: copy files from HDFS to local filesystem or downloads files to the localfilesystem.

Syntax : hadoop fs – **copyToLocal** <source file > <destination path>

Example:

hadoop fs -copyToLocal file2 . (Copied to default directory of local filesystem)

hadoop fs –get out,txt

d) Deleting files :

Hadoop command for removing files is **rm**

Syntax : hadoop fs -rm <file name>

Example:

hadoop fs -rm file1

hadoop fs -rmr myfolder1 (remove directory recursively)

e) Moving files :

mv : Move file from source to destination.

Note:- Moving files across filesystem is not permitted

Syntax : hadoop fs –mv <src> <dest>

Example:

hadoop fs -mv /user/cloudera/file1 user/cloudera/myfolder1

f) listing of files :

ls : List the contents of a directory.

Syntax :

hadoop fs -ls <args>

Example:

hadoop fs – ls hadoop fs –ls / (list the contents of root directory)

hadoop fs –lsr / (recursively displays entries in all subdirectories of path)

hadoop fs –ls –R hadoop fs –lsr /user/cloudera/myfolder

Assignment cum Tutorial Questions

A. Objective Questions

1) HDFS is designed for_____ [ ]

A) Storing very large files C) Commodity Hardware

B) Streaming data access D) All

2) HDFS is _____Architecture.

3) Data node is _____daemon. [ ]

A) Storage B) Computing C) Server D) None

4) What mode that a Hadoop can run? [ ]

A) Standalone C) Fully Distributed Mode

B) Pseudo-Distributed mode D) All

5) In which language is Hadoop written?

A) C++ B) Java C) Ruby D) Python

6) HDFS always needs to work with large data sets. (True/ False)

7) HDFS follows the write-once, read-many. (True/ False)

8) _____ node serves as the Slave and is responsible for carrying out the Tasks that have been assigned to it by the JobTracker.

A) TaskReduce B) Mapreduce C) TaskTracker D)JobTracker

9) Which of the following is a distributed data warehouse ? [ ]

A) Hive B) Oozie C) Zookeeper D) Sqoop

10) What is the minimum amount of data that a disk can read or write in HDFS? [ ]

A) Byte size B) Block size C) Heap D) None

11) For reading/Writing data to/from HDFS.Clients first connect to [ ]

A) Name Node C) Secondary Name Node

B) Data Node D) none

12) The main goal of HDFS high availability is_____. [ ]

A) Faster creation of the replicas of primary namenode.

B) To reduce the cycle time required to bring back a new primary namenode after existing primary fails.

C) Prevent data loss due to failure of primary namenode.

D) Prevent the primary namenode form becoming single point of failure.

13) Negative aspect to the importance of the Name Node___. [ ]

A) Single point of failure C) No failure

B) Double point of failure D) None.

14) If we use Cloudera distributation of hadoop which is the default directory for HDFS [ ]

A) /home/cloudera C) /cloudera

B) /user/cloudera D) None

15) The information mapping data blocks with their corresponding files is stored in [ ]

A) Data Node C) Job Tracker

B) Name Node D) Task Tracker

16) What happen if number of reducer is 0 in Hadoop? [ ]

A) Map-only job take place

B) Reduce-only job take place.

C) Reducer output will be the final output

D) None

17) The HDFS command to create the copy of a file from a local system to HDFS is which of the following? [ ]

A) copyFromLocal C) copyfromlocal

B) CopyFromLocal D) copylocal

18) The daemons associated with the Map Reduce phase are_____ and Task-Trackers. [ ]

A) Job Tracker C) Reduce Trackers

B) Map Tracker D) All

19) In order to read any file in HDFS, instance of [ ]

A) fileSystem B) datastream C) outstream D) inputstream

20) _____ is a tool for efficiently moving data between relational databases and HDFS.

A) HBase B) Sqoop C) Avro D) Hive

B. Descriptive Questions

1) Distinguish distributed file system and HDFS? Summarize the design principles of Hadoop and explain where it does not works well.
2) Elaborate the history of Hadoop and explain its architecture.
3) Illustrate apache Hadoop and Hadoop ecosystem.
4) Write the methods for creating directory and display its status
5) Discuss Building blocks of Hadoop (or) Identify various Hadoop daemons and explain their roles in a Hadoop Cluster
6) What is Hadoop HDFS? Draw and explain its architecture?
7) How a secondary name node differs from the name node in HDFS?
8) Distinguish the uses of Name node and Data node in HDFS..
9) How to copy file from the local file system to HDFS and vice versa.