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

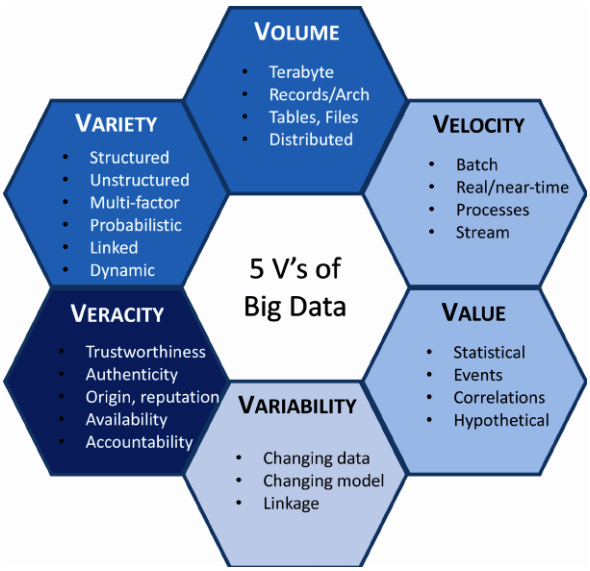
These data come from many sources like

- **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.

5V's of Big Data

- **Volume:** the size and amounts of big data that companies manage and analyze
- **Value:** the most important “V” from the perspective of the business, the value of big data usually comes from insight discovery and pattern recognition that lead to more effective operations, stronger customer relationships and other clear and quantifiable business benefits
- **Variety:** the diversity and range of different data types, including unstructured data, semi-structured data and raw data
- **Velocity:** the speed at which companies receive, store and manage data – e.g., the specific number of social media posts or search queries received within a day, hour or other unit of time

- **Veracity:** the "truth" or accuracy of data and information assets, which often determines executive-level confidence



What is 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.

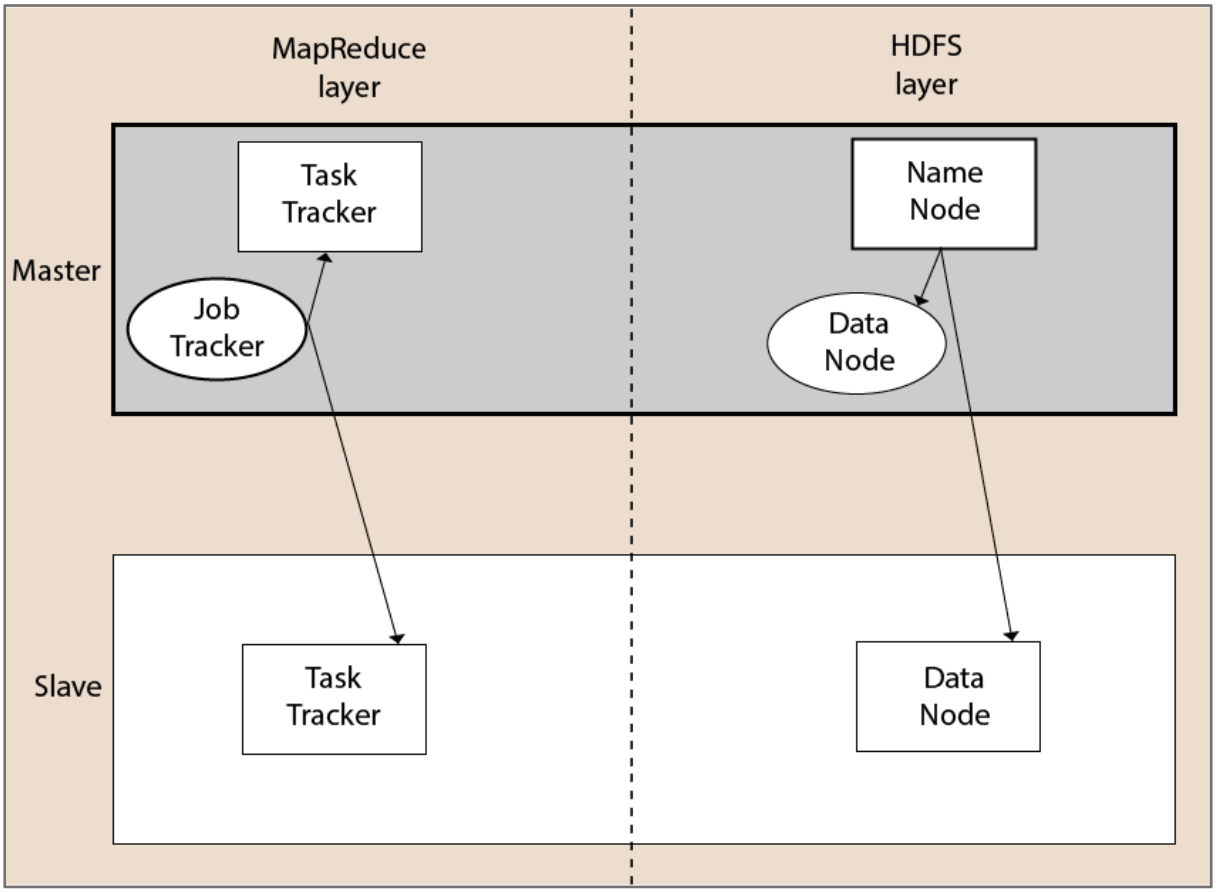
Modules of Hadoop

1. **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.
2. **Yarn:** Yet another Resource Negotiator is used for job scheduling and manage the cluster.
3. **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.
4. **Hadoop Common:** These Java libraries are used to start Hadoop and are used by other Hadoop modules.

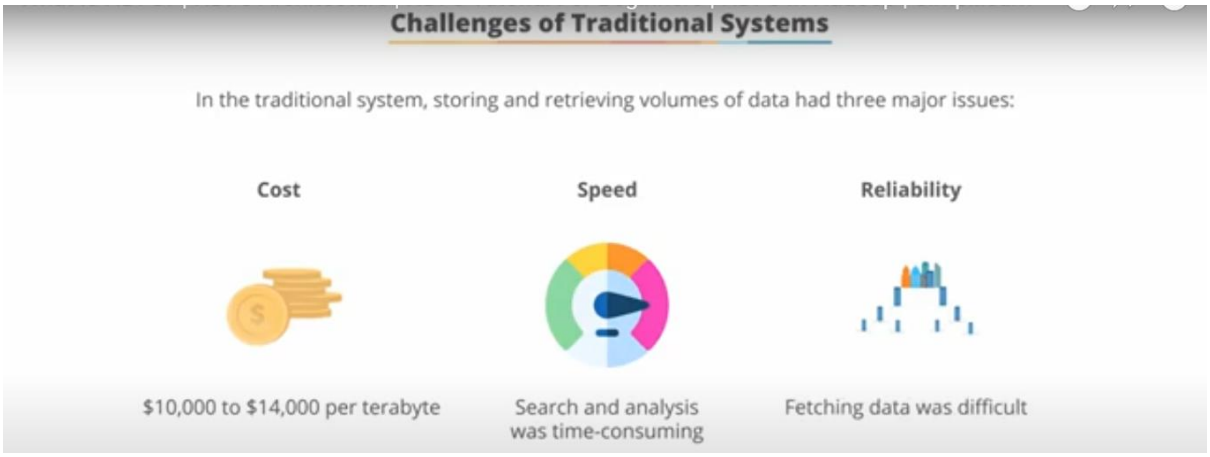
Hadoop Architecture

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.

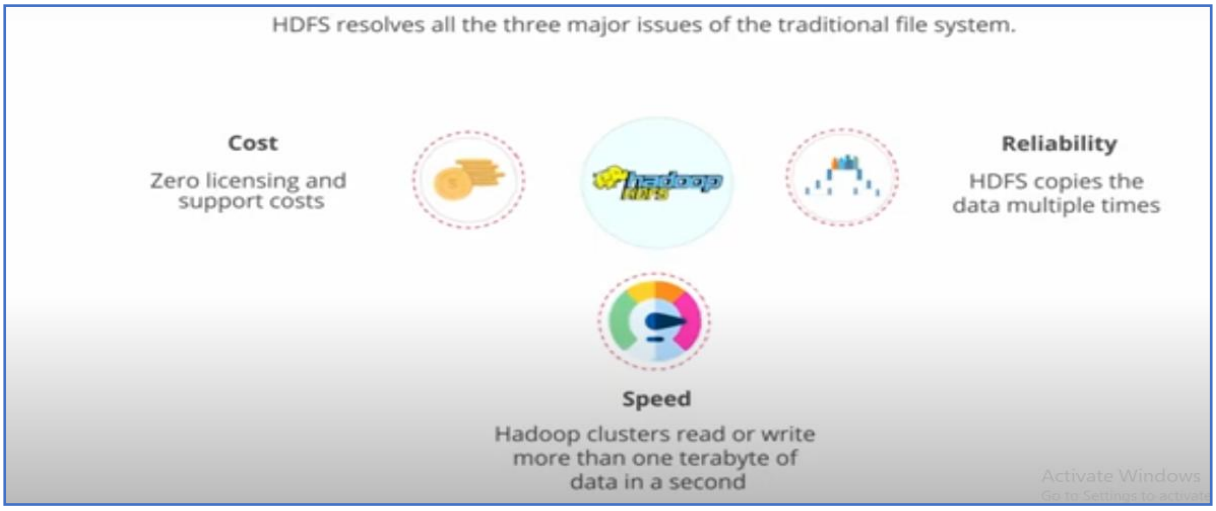
A Hadoop cluster consists of a single master and multiple slave nodes. The master node includes Job Tracker, Task Tracker, NameNode, and DataNode whereas the slave node includes DataNode and TaskTracker.



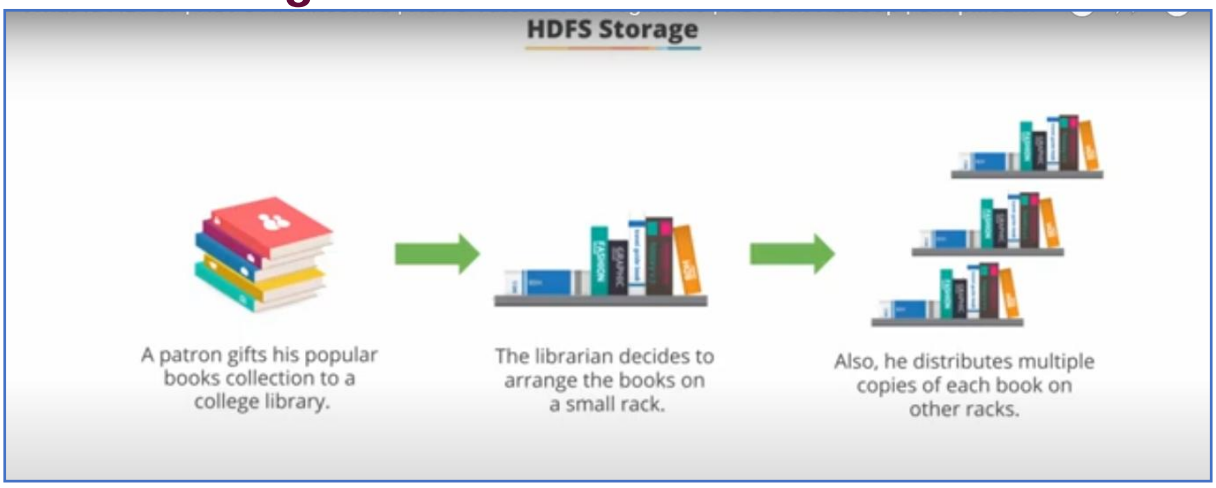
Challenges of Traditional File System



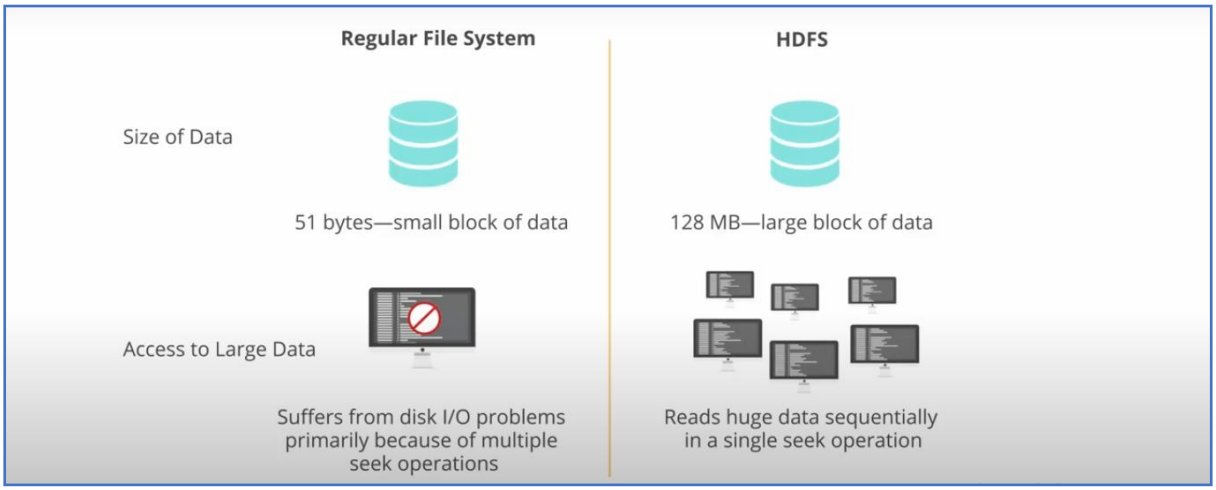
Need of HDFS



HDFS Storage



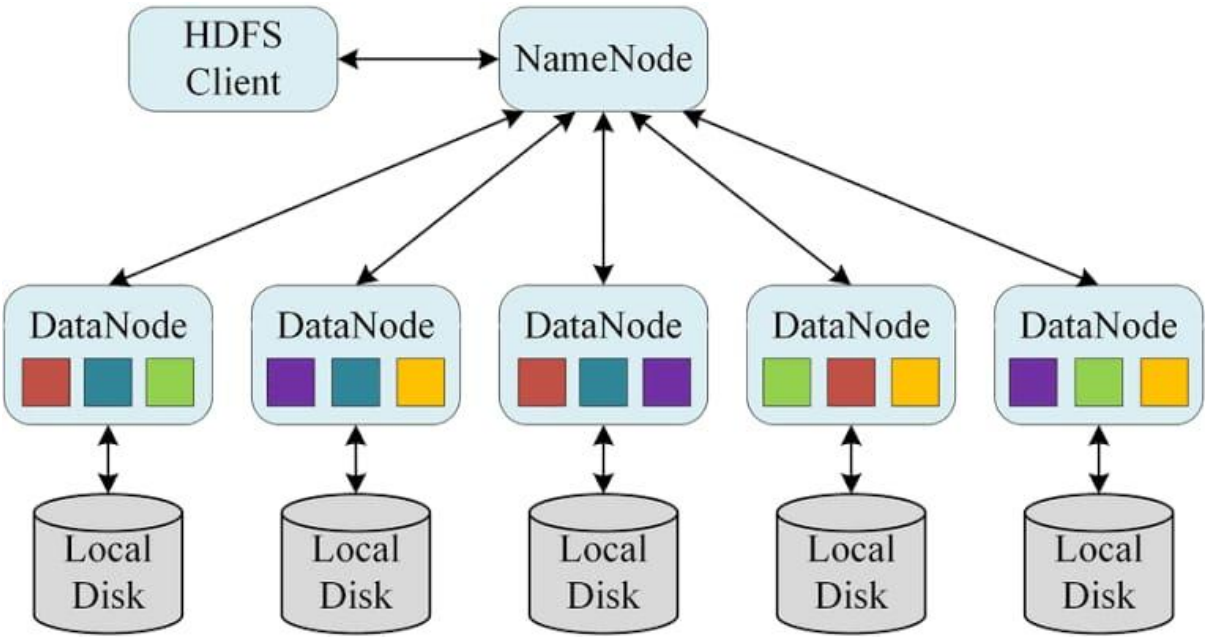
Regular File System vs. HDFS



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 Data Node 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 cmdcmdblock 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

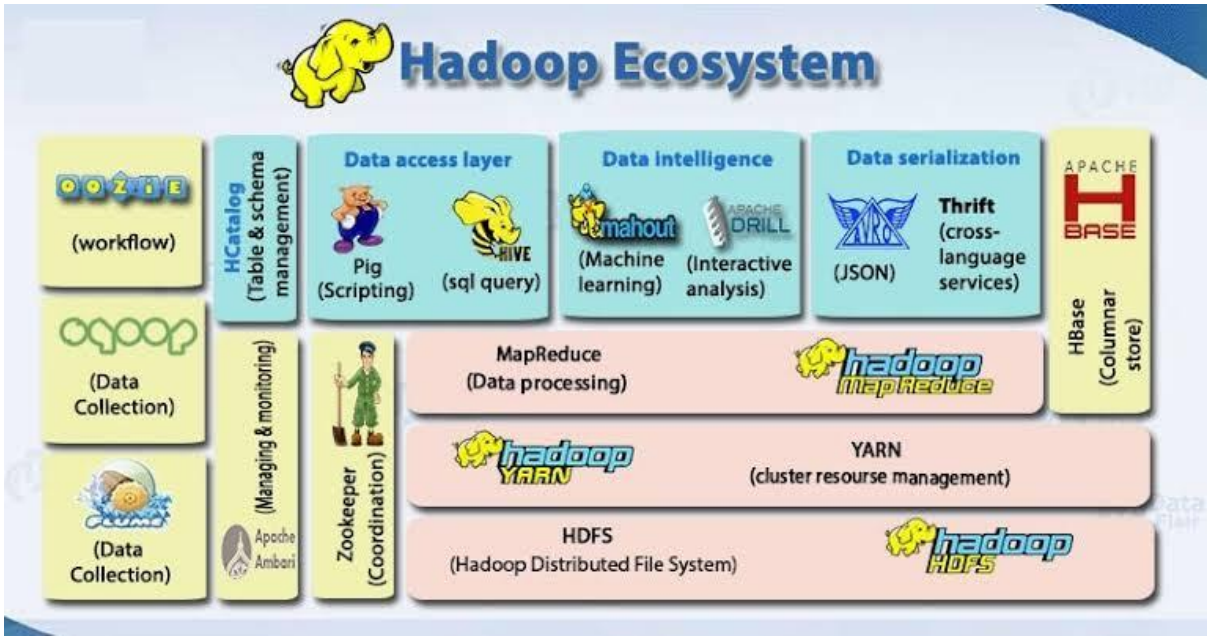
- It works as a slave node for Job Tracker.
- It receives task and code from Job Tracker and applies that code on the file. This process can also be called as a Mapper.

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.

Hadoop Ecosystem Components

Hadoop Ecosystem is a platform or a suite which provides various services to solve the big data problems. It includes Apache projects and various commercial tools and solutions. There are *four major elements of Hadoop* i.e. **HDFS, MapReduce, YARN, and Hadoop Common Utilities**. Most of the tools or solutions are used to supplement or support these major elements. All these tools work collectively to provide services such as absorption, analysis, storage and maintenance of data etc.



Following are the components that collectively form a Hadoop ecosystem:

- **HDFS:** Hadoop Distributed File System

- **YARN:** Yet Another Resource Negotiator
- **MapReduce:** Programming based Data Processing
- **Spark:** In-Memory data processing
- **PIG, HIVE:** Query based processing of data services
- **HBase:** NoSQL Database
- **Mahout, Spark MLLib:** Machine Learning algorithm libraries
- **Solar, Lucene:** Searching and Indexing
- **Zookeeper:** Managing cluster
- **Oozie:** Job Scheduling

All these toolkits or components revolve around one term i.e. *Data*. That's the beauty of Hadoop that it revolves around data and hence making its synthesis easier.

HDFS:

- HDFS is the primary or major component of Hadoop ecosystem and is responsible for storing large data sets of structured or unstructured data across various nodes and thereby maintaining the metadata in the form of log files.
- HDFS consists of two core components i.e.
    1. Name node
    2. Data Node
- Name Node is the prime node which contains metadata (data about data) requiring comparatively fewer resources than the data nodes that stores the actual data. These data nodes are commodity hardware in the distributed environment. Undoubtedly, making Hadoop cost effective.
- HDFS maintains all the coordination between the clusters and hardware, thus working at the heart of the system.

YARN:

- Yet Another Resource Negotiator, as the name implies, YARN is the one who helps to manage the resources across the clusters. In short, it performs scheduling and resource allocation for the Hadoop System.
- Consists of three major components i.e.
    1. Resource Manager
    2. Nodes Manager
    3. Application Manager
- Resource manager has the privilege of allocating resources for the applications in a system whereas Node managers work on the allocation of resources such as CPU, memory, bandwidth per machine and later on acknowledges the resource manager. Application manager works as an

interface between the resource manager and node manager and performs negotiations as per the requirement of the two.

MapReduce:

- By making the use of distributed and parallel algorithms, MapReduce makes it possible to carry over the processing's logic and helps to write applications which transform big data sets into a manageable one.
- MapReduce makes the use of two functions i.e. Map() and Reduce() whose task is:
    1. *Map()* performs sorting and filtering of data and thereby organizing them in the form of group. Map generates a key-value pair based result which is later on processed by the Reduce() method.
    2. *Reduce()*, as the name suggests does the summarization by aggregating the mapped data. In simple, Reduce() takes the output generated by Map() as input and combines those tuples into smaller set of tuples.

PIG:

Pig was basically developed by Yahoo which works on a pig Latin language, which is Query based language similar to SQL.

- It is a platform for structuring the data flow, processing and analyzing huge data sets.
- Pig does the work of executing commands and in the background, all the activities of MapReduce are taken care of. After the processing, pig stores the result in HDFS.
- Pig Latin language is specially designed for this framework which runs on Pig Runtime. Just the way Java runs on the JVM.
- Pig helps to achieve ease of programming and optimization and hence is a major segment of the Hadoop Ecosystem.

HIVE:

- With the help of SQL methodology and interface, HIVE performs reading and writing of large data sets. However, its query language is called as HQL (Hive Query Language).
- It is highly scalable as it allows real-time processing and batch processing both. Also, all the SQL datatypes are supported by Hive thus, making the query processing easier.
- Similar to the Query Processing frameworks, HIVE too comes with two components: *JDBC Drivers* and *HIVE Command Line*.
- JDBC, along with ODBC drivers work on establishing the data storage permissions and connection whereas HIVE Command line helps in the processing of queries.

Mahout:

- Mahout, allows Machine Learnability to a system or application. Machine Learning, as the name suggests helps the system to develop itself based on some patterns, user/environmental interaction or on the basis of algorithms.
- It provides various libraries or functionalities such as collaborative filtering, clustering, and classification which are nothing but concepts of Machine learning. It allows invoking algorithms as per our need with the help of its own libraries.

Apache Spark:

- It's a platform that handles all the process consumptive tasks like batch processing, interactive or iterative real-time processing, graph conversions, and visualization, etc.
- It consumes in memory resources hence, thus being faster than the prior in terms of optimization.
- Spark is best suited for real-time data whereas Hadoop is best suited for structured data or batch processing, hence both are used in most of the companies interchangeably.

Apache HBase:

- It's a NoSQL database which supports all kinds of data and thus capable of handling anything of Hadoop Database. It provides capabilities of Google's BigTable, thus able to work on Big Data sets effectively.
- At times where we need to search or retrieve the occurrences of something small in a huge database, the request must be processed within a short quick span of time. At such times, HBase comes handy as it gives us a tolerant way of storing limited data

Other Components: Apart from all of these, there are some other components too that carry out a huge task in order to make Hadoop capable of processing large datasets. They are as follows:

- **Solr, Lucene:** These are the two services that perform the task of searching and indexing with the help of some java libraries, especially Lucene is based on Java which allows spell check mechanism, as well. However, Lucene is driven by Solr.
- **Zookeeper:** There was a huge issue of management of coordination and synchronization among the resources or the components of Hadoop which resulted in inconsistency, often. Zookeeper overcame all the problems by performing synchronization, inter-component based communication, grouping, and maintenance.
- **Oozie:** Oozie simply performs the task of a scheduler, thus scheduling jobs and binding them together as a single unit. There is two kinds of jobs .i.e Oozie workflow and Oozie coordinator jobs. Oozie workflow

is the jobs that need to be executed in a sequentially ordered manner whereas Oozie Coordinator jobs are those that are triggered when some data or external stimulus is given to it.

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.

Hadoop Installation

Step 1: Download and Install Java

1. **Download JDK**: Download java version "1.8.0_411"official Oracle website

2. **Install JDK**: Follow the installation instructions provided by Oracle.

3. **Set JAVA_HOME Environment Variable**:

    - Right-click on This PC or Computer on the desktop or in File Explorer.
    - Click Properties.
    - Click Advanced system settings.
    - Click Environment Variables.
    - Click New under System variables.
    - Enter JAVA_HOME as the Variable name.
    - Enter the path to your JDK installation as the Variable value (C:\Java\jdk-1.8).
    - Click OK.

4. **Add Java to PATH:**

    - In the Environment Variables window, find the Path variable under System variables and click Edit.
    - Click New and add the path to the JDK bin directory (C:\Java\jdk-1.8\bin).
    - Click OK.

Step 2: Download and Install Hadoop

1. **Download Hadoop**: Download Hadoop 3.2.4 from the Apache Hadoop Releases page.
2. **Extract Hadoop**: Extract the downloaded Hadoop tar.gz file to a suitable directory, e.g., C:\hadoop.

3. **Set HADOOP_HOME Environment Variable**:
    - Follow the same steps as setting JAVA_HOME.

- Enter HADOOP_HOME as the Variable name.
- Enter the path to your Hadoop installation as the Variable value (C:\hadoop).
- Click OK.

4. **Add Hadoop to PATH**:
   - In the Environment Variables window, find the Path variable under System variables and click Edit.
   - Click New and add the path to the Hadoop bin directory (C:\hadoop).
   - Click OK.

Step 3: Configure Hadoop

1. **Edit hadoop-env.cmd**:
   - Navigate to C:\hadoop\etc\hadoop\hadoop-env.cmd.
   - Set the JAVA_HOME variable to your JDK installation path:

     ```
     set JAVA_HOME=C:\Java\jdk-1.8
     ```

2. **Edit core-site.xml**:
   - Navigate to C:\hadoop\etc\hadoop\core-site.xml.
   - Add the following configuration between <configuration> tags:

     ```
     <property>
     <name>fs.defaultFS</name>
     <value>hdfs://localhost:9000</value>
     </property>
     <property>
     <name>hadoop.tmp.dir</name>
     <value>file:/C:/hadoop/tmp</value>
     </property>
     ```

3. **Edit hdfs-site.xml**:
   - Navigate to C:\hadoop\etc\hadoop\hdfs-site.xml.
   - Add the following configuration:

     ```
     <property>
     <name>dfs.replication</name>
     <value>1</value>
     </property>
     <property>
     <name>dfs.namenode.name.dir</name>
     ```

```
<value>file:/C:/hadoop/data/namenode</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>file:/C:/hadoop/data/datanode</value>
</property>
```

4. **Edit mapred-site.xml**:
   - Rename mapred-site.xml.template to mapred-site.xml in C:\hadoop\etc\hadoop\.
   - Add the following configuration:

```
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
```

5. **Edit yarn-site.xml**:
   - Navigate to C:\hadoop\etc\hadoop\yarn-site.xml.
   - Add the following configuration:

```
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
```

Step 4: Format the Namenode

1. Open a command prompt and run:

   hdfsnamenode -format

   The command hdfsnamenode -format is used to format the Hadoop Distributed File System (HDFS) Namenode. Formatting the Namenode initializes the directory structure and sets up the initial filesystem metadata, including the namespace and version. This command should be run only once when setting up Hadoop for the first time or when explicitly needed (such as during recovery or maintenance), as it wipes out all metadata and files in the filesystem. It prepares the Namenode to start fresh with a clean slate.

   Check by typing command
   Jps

Output will be:

Step 5: Start Hadoop Daemons

1. Open a command prompt and navigate to C:\hadoop\sbin.
2. Start the HDFS daemons:

   start-dfs.cmd

   Check by typing command
   Jps

   Output will be:
   7824 Jps
   12696 DataNode
   6200 NameNode

The `start-dfs.cmd` command is used to start the Hadoop Distributed File System (HDFS) daemons on a Hadoop cluster. Specifically, it launches the following components:

1. **NameNode**: Manages the file system namespace and controls access to files by clients.
2. **DataNode(s)**: Stores the actual data blocks in HDFS and serves read and write requests from the file system's clients.
3. **Secondary NameNode**: Performs periodic checkpoints of the HDFS metadata stored on the NameNode.

When you run `start-dfs.cmd`, it:

1. Starts the NameNode on the designated master node.
2. Starts the DataNodes on all the designated slave nodes.
3. Starts the Secondary NameNode if configured.

This command is typically run after setting up and configuring the Hadoop environment to get the HDFS components up and running, enabling the cluster to store and manage data.

3. Start the YARN daemons:

   start-yarn.cmd

Check by typing command
Jps

Output will be:
7824 Jps
12696 DataNode
6200 NameNode
23852 ResourceManager
380 NodeManager

The `start-yarn.cmd` command is used to start the Yet Another Resource Negotiator (YARN) daemons on a Hadoop cluster. YARN is the resource management layer of Hadoop, responsible for managing and scheduling resources in the cluster. Running this command starts the following key YARN components:

1. **ResourceManager**: Manages resources and schedules applications (such as MapReduce jobs) running on the cluster.
2. **NodeManager(s)**: Runs on each node in the cluster and is responsible for managing the resources available on that node, such as memory and CPU, and reporting their status to the ResourceManager.

When you run `start-yarn.cmd`, it:

1. Starts the ResourceManager on the designated master node.
2. Starts the NodeManagers on all the designated slave nodes.

This command is typically executed after `start-dfs.cmd` to ensure that the HDFS is running before the resource management components. Running both commands sets up the entire Hadoop environment, allowing you to store data in HDFS and run distributed processing jobs using YARN.

Step 6: Verify Hadoop Installation

1. Access the Hadoop web interfaces to ensure everything is running correctly:
   - HDFS: http://localhost:9870/
   - YARN: http://localhost:8088/

The URL `http://localhost:9870/` is used to access the HDFS NameNode web user interface (UI). This web UI provides an overview of the Hadoop Distributed File System (HDFS) and allows you to:

1. **Browse the filesystem**: View the directory structure and files stored in HDFS.
2. **Monitor HDFS**: Check the status of the NameNode, DataNodes, and overall health of the filesystem.

3. **View metrics and logs**: Access various metrics related to the performance and status of the HDFS, as well as logs for troubleshooting and monitoring purposes.
4. **Manage filesystem operations**: Perform administrative tasks like creating directories, uploading, downloading, or deleting files.

To access the HDFS web UI:

1. Ensure that the HDFS daemons are running by executing `start-dfs.cmd`.
2. Open a web browser and navigate to `http://localhost:9870/`.

This interface is very helpful for managing and monitoring the HDFS cluster visually.

The URL `http://localhost:8088/` is used to access the YARN ResourceManager web user interface (UI). This web UI provides an overview and management capabilities for the Yet Another Resource Negotiator (YARN) resource management framework in Hadoop. The ResourceManager UI allows you to:

1. **Monitor cluster metrics**: View resource usage, including memory and CPU usage across the cluster.
2. **Track application status**: See the status of applications (such as MapReduce jobs) running on the cluster, including their progress, resource consumption, and logs.
3. **View and manage queues**: Check the status of resource queues, which control how resources are allocated to different types of applications.
4. **Access logs**: Retrieve logs for debugging and performance analysis of applications running on the cluster.
5. **Monitor NodeManagers**: Get information about the status and health of NodeManagers in the cluster.

To access the YARN ResourceManager web UI:

1. Ensure that the YARN daemons are running by executing `start-yarn.cmd`.
2. Open a web browser and navigate to `http://localhost:8088/`.

This interface is essential for managing and monitoring the resource allocation and application execution within a Hadoop cluster.

Step 7: Run a Sample Hadoop Program (Server Side)

1. Open command prompt to c:\hadoop\sbin.

2. Create a folder named input

   hadoop fs -mkdir /input

3. Copy the data.txt to input directory oh hdfs

   hdfsdfs -put C:/Users/Akansha/data.txt /input

   To check:

4. Display the content of data.txt

   hdfsdfs -cat /input/data.txt

5. lists the files and directories in the specified HDFS directory
   c:\hadoop\sbin>hdfs dfs -ls /input
   Found 1 items
   -rw-r--r-- 1 Akansha supergroup 57 2024-08-02 10:01 /input/data.txt

6. Run a sample Hadoop Program i.e word count example:

   c:\hadoop\sbin>hadoop jar %HADOOP_HOME%\share\hadoop\mapreduce\hadoop-mapreduce-examples-3.2.4.jar wordcount /input /output

7. Display the list of files created in output folder
   c:\hadoop\sbin>hdfs dfs -ls /output

   Found 2 items
   -rw-r--r-- 1 Akansha supergroup 0 2024-07-25 21:59 /output/_SUCCESS
   -rw-r--r-- 1 Akansha supergroup 23 2024-07-25 21:59 /output/part-r-00000

8. Display the content of part-00000 file

   hdfsdfs -cat /output/part-r-00000

   Output:
   hello 1
   hii 2
   monday 2

Example: RandomWriter

1. **Run RandomWriter to generate random data:**

hadoop jar C:/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.4.jar randomwriter /randomwriter/output

/user/hadoop/randomwriter/output is the HDFS output directory where the generated random data will be stored.

2. **Verify the generated random data:**

   hdfsdfs -ls /randomwriter/output

3. **To check the output:**

   hdfsdfs -cat /randomwriter/output-part-r-00001

These examples demonstrate how to use different Hadoop example applications. Make sure the output directories specified in HDFS do not already exist, or Hadoop will throw an error. If they do exist, you can delete them using:

hdfsdfs -rm -r /output

hdfsdfs -rm -r /input