Part-A
Assignment No.1

Title: To perform Single node/Multiple node Hadoop Installation.

Objective: To study,

1. Configure Hadoop on open source software

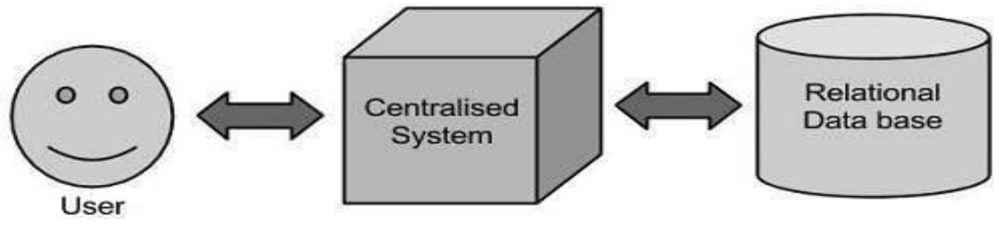
Theory:

Hadoop

Hadoop is an open sourcesoftware framework written in Java for distributed storage and distributed processing of very large data sets on computer clusters built from commodity hardware. All the modules in Hadoop are designed with a fundamental assumption that hardware failures (of individual machines or racks of machines) are common and thus should be automatically handled in software by the framework.

Traditional Approach

In this approach, an enterprise will have a computer to store and process big data. Here data will be stored in an RDBMS like Oracle Database, MS SQL Server or DB2 and sophisticated software can be written to interact with the database, process the required data and present it to the users for analysis purpose.

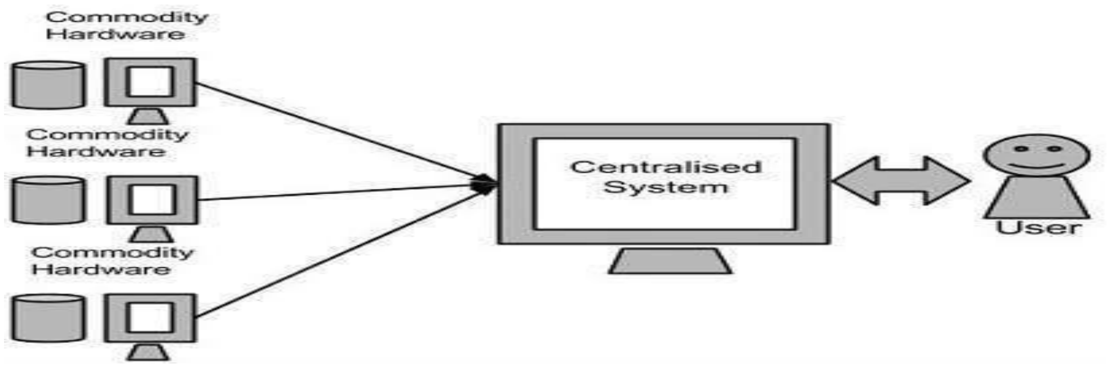


Limitation

This approach works well where we have less volume of data that can be accommodated by standard database servers, or up to the limit of the processor which is processing the data. But when it comes to dealing with huge amounts of data, it is really a tedious task to process such data through a traditional database server.

Google's Solution

Google solved this problem using an algorithm called MapReduce. This algorithm divides the task into small parts and assigns those parts to many computers connected over the network, and collectsthe results to form the final result dataset.

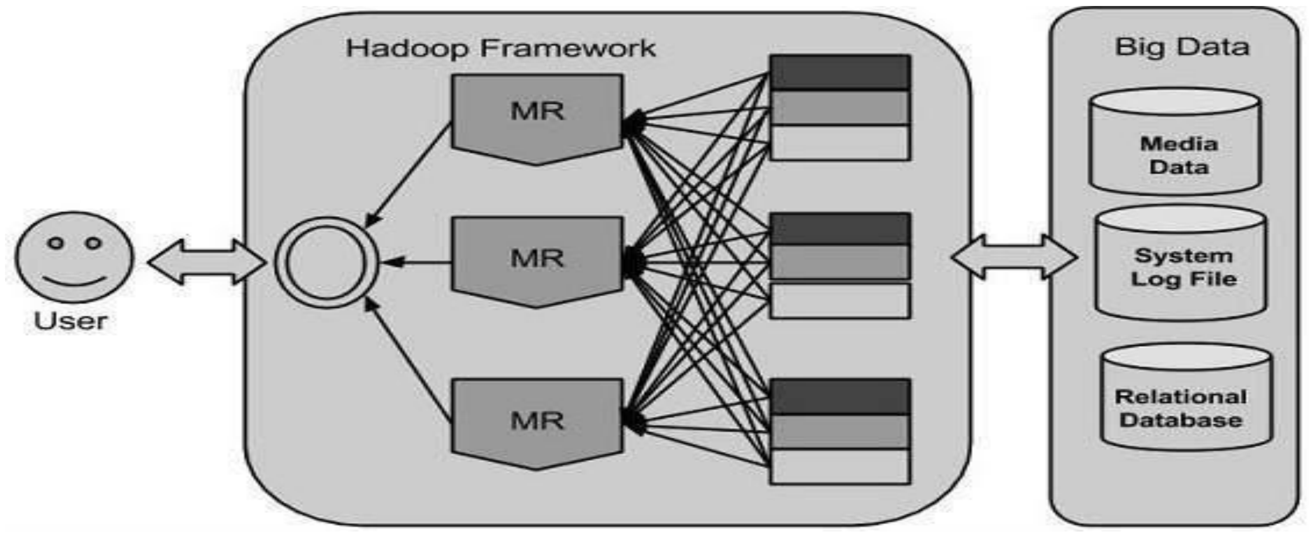


Above diagram shows various commodity hardware‟s which could be single CPU machines or servers with higher capacity.

Hadoop

Doug Cutting, Mike Cafarella and team took the solution provided by Google and started an Open Source Project called HADOOP in 2005 and Doug named it after his son's toy elephant. Now Apache Hadoop is a registered trademark of the Apache Software Foundation.

Hadoop runs applications using the MapReduce algorithm, where the data is processed in parallel on different CPU nodes. In short, Hadoop framework is capable enough to develop applications capable of running on clusters of computers and they could perform complete statistical analysis for huge amounts of data.



Hadoop Architecture

Hadoop framework includes following four modules:

Hadoop Common: These are Java libraries and utilities required by other Hadoop modules. These libraries provide file system and OS level abstractions and contains the necessary Java files and scripts required to start Hadoop.

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

Hadoop Distributed File System (HDFS™): A distributed file system that provides high- throughput access to application data.

HadoopMapReduce: This is YARN-based system for parallel processing of large data sets.

MapReduce

Hadoop**MapReduce** is a software framework for easily writing applications which process big amounts of data in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.

The term MapReduce actually refers to the following two different tasks that Hadoop programs perform:

- **The Map Task:** This is the first task, which takes input data and converts it into a set of data, where individual elements are broken down into tuples (key/value pairs).

- **The Reduce Task:** This task takes the output from a map task as input and combines those data tuples into a smaller set of tuples. The reduce task is always performed after the map task.

Typically both the input and the output are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks.

The MapReduce framework consists of a single master Job Tracker and one slave Task Tracker per cluster-node. The master is responsible for resource management, tracking resource consumption/availability and scheduling the jobs component tasks on the slaves, monitoring them and re-executing the failed tasks. The slaves Task Tracker execute the tasks as directed by the master and provide task-status information to the master periodically.

The Job Tracker is a single point of failure for the Hadoop MapReduce service which means if Job Tracker goes down, all running jobs are halted.

Hadoop Distributed File System (HDFS):-

Hadoop can work directly with any mountable distributed file system such as Local FS, HFTP FS, S3 FS, and others, but the most common file system used by Hadoop isthe Hadoop Distributed FileSystem (HDFS).

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 large clusters (thousands of computers) of small computer machines in a reliable, fault-tolerant manner.

HDFS uses a master/slave architecture where master consists of a single **Name Node** that manages the file system metadata and one or more slave **Data Nodes** that store the actual data.

Installation of Hadoop 2.9.0 on Ubuntu

1. Installing Java

Hadoop framework is written in Java!!

```
# Update the source list rashmi@laptop:~$sudo apt-get update

rashmi@laptop:~$ sudo apt-get install openjdk-7-jdk

rashmi@laptop:~$ java -version
```

2. Installing SSH

ssh has two main components:

1. **ssh** : The command we use to connect to remote machines - the client.
2. **sshd** : The daemon that is running on the server and allows clients to connect to the server.

The **ssh** is pre-enabled on Linux, but in order to start **sshd** daemon, we need to install **ssh** first. Use this command to do that :

```
rashmi@laptop:~$ sudo apt-get install ssh
```

This will install ssh on our machine. If we get something similar to the following, we can think it is setup properly:

```
rashmi@laptop:~$ which ssh
/usr/bin/ssh

rashmi@laptop:~$ which sshd
/usr/sbin/sshd
```

3. Create and Setup SSH Certificates

Hadoop requires SSH access to manage its nodes, i.e. remote machines plus our local machine. For our single-node setup of Hadoop, we therefore need to configure SSH access to localhost.

So, we need to have SSH up and running on our machine and configured it to allow SSH public key authentication.

Hadoop uses SSH (to access its nodes) which would normally require the user to enter a password. However, this requirement can be eliminated by creating and setting up SSH certificates using the following commands. If asked for a filename just leave it blank and press the enter key to continue.

```
rashmi@laptop:~$ ssh-keygen -t rsa -P ""

rashmi@laptop:~$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

The second command adds the newly created key to the list of authorized keys so that Hadoop can use ssh without prompting for a password.

We can check if ssh works:

```
rashmi@laptop:~$ ssh localhost
```

4. Install Hadoop

```
rashmi@laptop:~$ wget http://mirrors.sonic.net/apache/hadoop/common/hadoop- 2.9.0/hadoop-2.9.0.tar.gz
```

```
rashmi@laptop:~$ tar xvzf hadoop-2.9.0.tar.gz
```

We want to move the Hadoop installation to the **/usr/local/hadoop** directory using the following command:

```
rashmi@laptop:~$ sudo mv hadoop/ /usr/local/
```

```
rashmi@laptop:~$ sudo chown -R rashmi:rashmi /usr/local/hadoop
```

5. Setup Configuration Files

The following files will have to be modified to complete the Hadoop setup:

1. ~/.bashrc
2. /usr/local/hadoop/etc/hadoop/hadoop-env.sh
3. /usr/local/hadoop/etc/hadoop/core-site.xml
4. /usr/local/hadoop/etc/hadoop/mapred-site.xml.template
5. /usr/local/hadoop/etc/hadoop/hdfs-site.xml

1. ~/.bashrc:

Before editing the **.bashrc** file in our home directory, we need to find the path where Java has been installed to set the **JAVA_HOME** environment variable using the following command:

```
rashmi@laptop:~$ update-alternatives --config java
```

Now we can append the following to the end of **~/.bashrc**:

```
rashmi@laptop:~$ gedit .bashrc
```

```
rashmi@laptop:~$ source .bashrc
```

This command applies the changes made in the .bashrc file.

2. /usr/local/hadoop/etc/hadoop/hadoop-env.sh

We need to set **JAVA_HOME** by modifying **hadoop-env.sh** file.

```
rashmi@laptop:~$ gedit /usr/local/hadoop/etc/hadoop/hadoop-env.sh
```

Adding the above statement in the **hadoop-env.sh** file ensures that the value of JAVA_HOME variable will be available to Hadoop whenever it is started up.

3. /usr/local/hadoop/etc/hadoop/core-site.xml:

The **/usr/local/hadoop/etc/hadoop/core-site.xml** file contains configuration properties that Hadoop uses when starting up. This file can be used to override the default settings that Hadoop starts with.

```
rashmi@laptop:~$ sudo mkdir -p /app/hadoop/tmp
rashmi@laptop:~$ sudo chown rashmi:rashmi /app/hadoop/tmp
```

Open the file and enter the following in between the <configuration>   </configuration> tag:

```
rashmi@laptop:~$ gedit /usr/local/hadoop/etc/hadoop/core-site.xml
```

4. /usr/local/hadoop/etc/hadoop/mapred-site.xml

By default, the **/usr/local/hadoop/etc/hadoop/** folder contains
/usr/local/hadoop/etc/hadoop/mapred-site.xml.template
file which has to be renamed/copied with the name **mapred-site.xml**:

The **mapred-site.xml** file is used to specify which framework is being used for MapReduce.
We need to enter the following content in between the <configuration>   </configuration>
tag:

5. /usr/local/hadoop/etc/hadoop/hdfs-site.xml

The **/usr/local/hadoop/etc/hadoop/hdfs-site.xml** file needs to be configured for each host in the cluster that is being used. It is used to specify the directories which will be used as the **namenode** and the **datanode** on that host.

Before editing this file, we need to create two directories which will contain the namenode and the datanode for this Hadoop installation. This can be done using the following commands:

```
rashmi@laptop:~$    sudo    mkdir    -p    /usr/local/hadoop_store/hdfs/namenode
rashmi@laptop:~$    sudo    mkdir    -p     /usr/local/hadoop_store/hdfs/datanode
rashmi@laptop:~$ sudo chown -R rashmi:rashmi /usr/local/hadoop_store
```

Open the file and enter the following content in between the <configuration> </configuration> tag:

```
rashmi@laptop:~$ gedit /usr/local/hadoop/etc/hadoop/hdfs-site.xml
```

6. Format the New Hadoop Filesystem

Now, the Hadoop file system needs to be formatted so that we can start to use it. The format command should be issued with write permission since it creates **current** directory under **/usr/local/hadoop_store/hdfs/namenode** folder:

```
rashmi@laptop:~$ hadoop namenode -format
```

Note that **hadoop namenode -format** command should be executed once before we start using Hadoop. If this command is executed again after Hadoop has been used, it'll destroy all the data on the Hadoop file system.

7. <u>Starting Hadoop</u>

Now it's time to start the newly installed single node cluster. We can use **start-all.sh** or (**start-dfs.sh** and **start-yarn.sh**)

```
rashmi@laptop:~$ start-all.sh
```

We can check if it's really up and running:

```
rashmi@laptop:~$ jps
9026 NodeManager
7348 NameNode
9766 Jps
8887 ResourceManager
7507 DataNode
```

The output means that we now have a functional instance of Hadoop running on our VPS (Virtual private server).

8. Hadoop Web Interfaces

Let's start the Hadoop again and see its Web UI:

Accessing HADOOP through browser

http://localhost:50070/

Verify all applications for cluster

http://localhost:8088/

Conclusion: In this way, the Hadoop was installed & configured on Ubuntu.