American Journal of Engineering Research (AJER)
e-ISSN: 2320-0847 p-ISSN : 2320-0936
Volume-5, Issue-6, pp-146-151
www.ajer.org
Research Paper Open Access

Big Data Analysis Using Hadoop Mapreduce

Dr. Urmila R. Pol
Department Of Computer Science , Shivaji University, Kolhapur,India

ABSTRACT: *In recent years, "Big Data" has become a new pervasive term. Big Data is transforming science, engineering, medicine, healthcare, finance, business, and ultimately our society itself. The age of big data is now coming. But the traditional data analytics may not be able to handle such large quantities of data. In this paper, we present a study of Big Data and its analytics using Hadoop MapReduce, which is open-source software for reliable, scalable, distributed computing.*

Keywords: *Big Data, Hadoop, MapReduce,HDFS,zettabyte.*

I. INTRODUCTION TO BIG DATA

'Big Data' is a **data, but** with a **huge size**. 'Big Data' is a term used to describe a collection of data that is large in size and yet growing exponentially with time. In short,such a data is so massive and complex that none of the traditional data management tools are able to store it or process it efficiently. The **New York Stock Exchange** generates about *one terabyte* of new trade data per day. Statistic shows that *500+terabytes* of new data gets ingested into the databases of social media site **Facebook**, every day. This data is mainly generated in terms of photo and video uploads, message exchanges, putting comments etc.Single **Jet engine** can generate *10+terabytes* of data in *30 minutes* of a flight time. With many thousand flights per day, generation of data reaches up to many *Petabytes*. Much of the tech industry follows Gartner's '3Vs' model to define Big Data. Big data' could be found in three forms: **Structured ,Un-structured, Semi-structured.**
Any data that can be stored, accessed and processed in the form of fixed format is termed as a 'structured' data. Now days, we are foreseeing issues when size of such data grows to a enormous extent, typical sizes are being in the range of multiple Zettabyte. 10^{21} bytes equals to 1 zettabyte or one billion terabytes forms a zettabyte.Data stored in a relational database management system is one example of a **'structured'** data. A table in a database is an example of Structured Data.

Any data with unknown form or the structure is classified as unstructured data. In addition to the size being huge, un-structured data poses multiple challenges in terms of its processing for deriving value out of it. Typical example of **unstructured data** is, a heterogeneous data source containing a combination of simple text files, images, videos etc. Now a days organizations have wealth of data available with them but unfortunately they don't know how to derive value out of it since this data is in its raw form or unstructured format. Output returned by 'Google Search' is an example of **Unstructured Data**.**Semi-structured** data can contain both the forms of data. We can see **semi-structured data** as a strcutured in form but it is actually not defined with e.g. a table definition in relational DBMS. Example of semi structured data is a data represented in XML file.

II. HADOOP

Apache HADOOP is a framework used to develop data processing applications which are executed in a distributed computing environment. Similar to data residing in a local file system of personal computer system, in Hadoop, data resides in a distributed file system which is called as a Hadoop Distributed File system. Processing model is based on 'Data Locality' concept wherein computational logic is sent to cluster nodes(server) containing data.

This computational logic is nothing but a compiled version of a program written in a high level language such as Java. Such a program, processes data stored in Hadoop HDFS. HADOOP is an open source software framework. Applications built using HADOOP are run on large data sets distributed across clusters of commodity computers. Commodity computers are cheap and widely available. These are mainly useful for achieving greater computational power at low cost. Computer cluster consists of a set of multiple processing units (storage disk + processor) which are connected to each other and acts as a single system.
Apache Hadoop consists of two sub-projects –

1. Hadoop MapReduce: MapReduce is a computational model and software framework for writing applications which are run on Hadoop. These MapReduce programs are capable of processing massive data in parallel on large clusters of computation nodes.
2. HDFS (Hadoop Distributed File System): HDFS takes care of storage part of Hadoop applications. MapReduce applications consume data from HDFS. HDFS creates multiple replicas of data blocks and distributes them on compute nodes in cluster. This distribution enables reliable and enormously rapid computations.

Although Hadoop is best known for MapReduce and its distributed file system- HDFS, the term is also used for a family of related projects that fall under the umbrella of distributed computing and large-scale data processing. Other Hadoop-related projects at Apache include are Hive, Hbase, Mahout, Sqoop, Flume and ZooKeeper.

III. HADOOP INSTALLATION

HADOOP (2.6) SINGLE NODE INSTALLATION ON UBUNTU 14.04 LTS(32 bit)GUIDELINES

STEP 1:
open your terminal using(ctrl+alt+t) write down following command

```
urp@localhost$ sudo apt-get update
```

STEP 2:

```
urp@localhost$ sudo apt-get upgrade
```

STEP 3:

```
urp@localhost$ sudo apt-get install openssh-server
```

STEP 4:
First check whether jdk(java set up) is installed in your machine or not with following command
open your terminal using (ctrl+at+t) and write down following command

```
urp@localhost$java -version
```

(IF JAVA IS ALREADY INSTALLED THEN NO PROBLEM ,DONT EXECUTE STEP NUMBER 5)
(in my machine java is not installed,so I have to installed first java on my machine with following command)

STEP 5:
for installing java write down following command

```
urp@localhost$sudo apt-get  install openjdk-7-jdk
```

```
urp@localhost$ java  -version
```

NOTE: IT WILL SHOW US JAVA 1.7 VERSION IS INSTALLED SUCCESSFULLY
it is installed in file /usr/lib/jvm/java-7-openjdk-i386(this is the java path)

STEP 6:
Download hadoop from following command

```
urp@localhost$wget http://mirrors.sonic.net/apache/hadoop/common/hadoop-2.6.0/hadoop-2.6.0.tar.gz
```

(move the hadoop folder into /usr/local file(it means that our hadoop is installed on /usr/local directory)

NOTE: You can install hadoop where you wants for my cases I am installed on /usr/local directory

```
urp@localhost$sudo tar -zxvf ~/Downloads/hadoop-2.6.0.tar.gz -C /usr/local;
```

Note: To extract   hadoop-2.6.0.tar.gz in /usr/local(location);

```
urp@localhost$sudo mv /usr/local/hadoop-2.6.0 /usr/local/hadoop;
```

Note : To rename file

```
urp@localhost$sudo chown -R urp /usr/local/hadoop;
```

STEP 7
ADD HADOOP GROUP AND USER

```
urp@localhost$sudo addgroup hadoop;  (hadoop is my user)
urp@localhost$sudo adduser --ingroup hadoop urp;(urp is my hadoop user name)
urp@localhost$sudo adduser urp sudo;
```

STEP 8
Setup SSH Certificate key genration:

```
urp@localhost$ssh-keygen -t rsa -P ''
urp@localhost$cat .ssh/id_rsa.pub >> .ssh/authorized_keys
```

STEP 9 Get your IP Address

```
urp@localhost$ ifconfig
urp@localhost$ sudo gedit /etc/hosts;
urp@localhost$ sudo gedit /etc/hostname;
urp@localhost$ssh localhost;
```

Note: Restart the computer(urp@localhost$ sudo reboot)

STEP 10
EDIT bashrc file and write down following code for hadoop

```
urp@localhost$ sudo nano ~/.bashrc

urp@localhost$ source ~/.bashrc
```

write down the following code on above file

```
--------------------------------------------------------------------------------------------------------------------------------
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-i386
export HADOOP_INSTALL=/usr/local/hadoop
export PATH=urp@localhost$PATH:urp@localhost$HADOOP_INSTALL/bin
export PATH=urp@localhost$PATH:urp@localhost$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=urp@localhost$HADOOP_INSTALL
export HADOOP_COMMON_HOME=urp@localhost$HADOOP_INSTALL
export HADOOP_HDFS_HOME=urp@localhost$HADOOP_INSTALL
export YARN_HOME=urp@localhost$HADOOP_INSTALL
export HADOOP_COMMON_LIB_NATIVE_DIR=urp@localhost$HADOOP_HOME/lib/native
export JAVA_LIBRARY_PATH=urp@localhost$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path"=urp@localhost$HADOOP_INSTALL/lib
--------------------------------------------------------------------------------------------------------------------------------
```

(**NOTE:** Here JAVA_HOME is jdk path Here my jdk is installed in /usr/lib/jvm/java-7-openjdk-i386
HADOOP_INSTALL is the path where hadoop is installed in my computer hadoop is installed in /usr/loca/hadoop)

STEP 10 : configure following xml files

```
urp@localhost$cd /usr/local/hadoop/etc/hadoop
```

Edit core-site.xml:

```
urp@localhost$  sudo gedit core-site.xml
```

write down following lines

```
<property>
<name>fs.default.name</name>
```

```
<value>hdfs://localhost:9000</value>
</property>
```

STEP 10.1
configure hdfs-site.xml

```
urp@localhost$ sudo gedit hdfs-site.xml
```

write down following lines

```
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.permissions</name>
<value>false</value>
</property>
```

STEP 10.2
configure mapred-site.xml

NOTE: actualy in /usr/local/hadoop/etc/hadoop folder mapred-site.xml file is not present insted of that mapred-site.templete file is available,So that we can convert mapred-site.templet into mapred-site.xml file with following command

CONVERT mapred-site.templete into mapred-site.xml

```
urp@localhost$ mv mapred-site.xml.template mapred-site.xml
```

STEP 10.3 Edit mapred-site.xml file

```
urp@localhost$sudo nano  mapred-site.xml
```

write down following lines

```
<property>
<name>mapred.job.tracker</name>
<value>localhost:9001</value>
</property>
```

STEP 10.4 Configure hadoop-env.sh file

```
urp@localhost$sudo nano hadoop-env.sh
```

```
/usr/lib/jvm/java-7-openjdk-i386
```

(just write down JAVA_HOME path) my jdk is installed in /usr/lib/jvm/java-7-openjdk-i386)

STEP 11:
FORMAT NAMENODE

use following command on terminal :

```
cd /usr/local/hadoop/bin
```

write down following command terminal :

```
urp@localhost$hadoop namenode -format
```

STEP 12: START ALL THE DEMONS

```
urp@localhost$ start-all.sh
```

STEP 13: CHECK WHETHER ALL THE DEMONS ARE RUNNING

write down following command

```
urp@localhost$   jps
```

(if all the demons are present then your hadoop installation is sucess othrwise not)

IV. HADOOP MAPREDUCE

Hadoop MapReduce is a software framework for easy writing applications which process huge amounts of data (multi-terabyte datasets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. A MapReduce *job* usually splits the input dataset into independent chunks which are processed by the *map tasks* in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the *reduce tasks*. Typically, both the input and the output of the job are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks. Typically the compute nodes and the storage nodes are the same, that is, the MapReduce framework and the Hadoop Distributed File System are running on the same set of nodes[1]

This configuration allows the framework to effectively schedule tasks on the nodes where the data were already present, resulting in very high total bandwidth across the cluster. The MapReduce framework consistsof a single master ResourceManager, one slave Node Manager per cluster-node, and MRAppMaster per application.Minimally, applications specify the input/output locations and supply *map* and *reduce* functions via implementations of appropriate interfaces and/or abstract-classes. These, and other job parameters,include the *job configuration*.The Hadoop *job client* then submits the job (jar/executable etc.) and configuration to the ResourceManager which then assumes the responsibility of distributing the software/configuration to the slaves, scheduling tasks and monitoring them, providing status and diagnostic information to the job-client.
Although the Hadoop framework is implemented in Java™, MapReduce applications need not be written in Java.

- Hadoop Streaming is a utility which allows users to create and run jobs with any executables (e.g. shell utilities) as the mapper and/or the reducer.
- Hadoop Pipes is a SWIG-compatible C++ API to implement MapReduce applications (non JNI™ based).

Inputs and Outputs

The MapReduce framework operates exclusively on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types.

The key and value classes have to be serializable by the framework and hence need to implement the Writable interface. Additionally, the key classes have to implement the WritableComparable interface to facilitate sorting by the framework.
Input and Output types of a MapReduce job:
(input) <k1, v1> -> **map** -> <k2, v2> -> **combine** -> <k2, v2> -> **reduce** -> <k3, v3> (output)

MapReduce Example

WordCount is a simple application that counts the number of occurrences of each word in a given input set. WordCount programme is available on https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html#Source_Code.[1]
Copy code into java file. Execute following commands.

Set Environment Variables

```
export JAVA_HOME=/usr/java/default
export PATH=urp@localhost${JAVA_HOME}/bin:urp@localhost${PATH}
export HADOOP_CLASSPATH=urp@localhost${JAVA_HOME}/lib/tools.jar
```

Compile WordCount.java and create a jar:

```
urp@localhost$ bin/hadoop com.sun.tools.javac.Main WordCount.java
urp@localhost$ jar cf wc.jar WordCount*.class
```

Assuming that:

- /user/urp/wordcount/input - input directory in HDFS
- /user/urp/wordcount/output - output directory in HDFS

Sample text-files as input:

```
urp@localhost$ bin/hadoop fs -ls /user/urp/wordcount/input/ /user/urp/wordcount/input/file01
/user/urp/wordcount/input/file02

urp@localhost$ bin/hadoop fs -cat /user/joe/wordcount/input/file01

Hello Hadoop Welcome Hadoop

urp@localhost$ bin/hadoop fs -cat /user/joe/wordcount/input/file02
Hello Hadoop Goodbye Hadoop
```

Run the application:

```
urp@localhost$ bin/hadoop jar wc.jar WordCount /user/urp/wordcount/input /user/urp/wordcount/output
```

Output:

```
urp@localhost$ bin/hadoop fs -cat /user/urp/wordcount/output/part-r-00000`
Goodbye 1
Hadoop 4
Hello 2
Welcome 1
```

V. CONCLUSION

Hadoop is an open-source framework that allows to store and process big data in a distributed environment across clusters of computers using simple programming models. It is designed to scale up from single server to thousands of machines, each offering local computation and storage. The major advantage of Hadoop MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into mappers and reducers is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.

REFERENCES

[1]. https://hadoop.apache.org/docs/