ISSN: 2277-9655
Impact Factor: 4.116

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

UNDERSTANDING AND CONFIGURING HADOOP: TO HANDLE THE LARGE AMOUNT OF DATA

Rahul Rawat*
* Department of Computer Sc. & Engg ,University Institute of Engineering and Technology, KUK, Haryana (India)

DOI: 10.5281/zenodo.59632

ABSTRACT

Data is getting bigger and bigger in size that is called as Big Data. Big Data may be structured, unstructured and semi structured. Traditional systems are not good to manage this huge amount of data. So, it is required to use best sources to manage this Big Data. Hadoop is Highly Archived Distributed Object Oriented Programming tool which is an open source software platform. Hadoop is written Java. It is used to store and manage large amount of data. In this paper configuration of Hadoop single node cluster is explained. Hardware and software requirements are also described. Some running commands are also explained for Hadoop. Map Reduce job of Hadoop also presented.

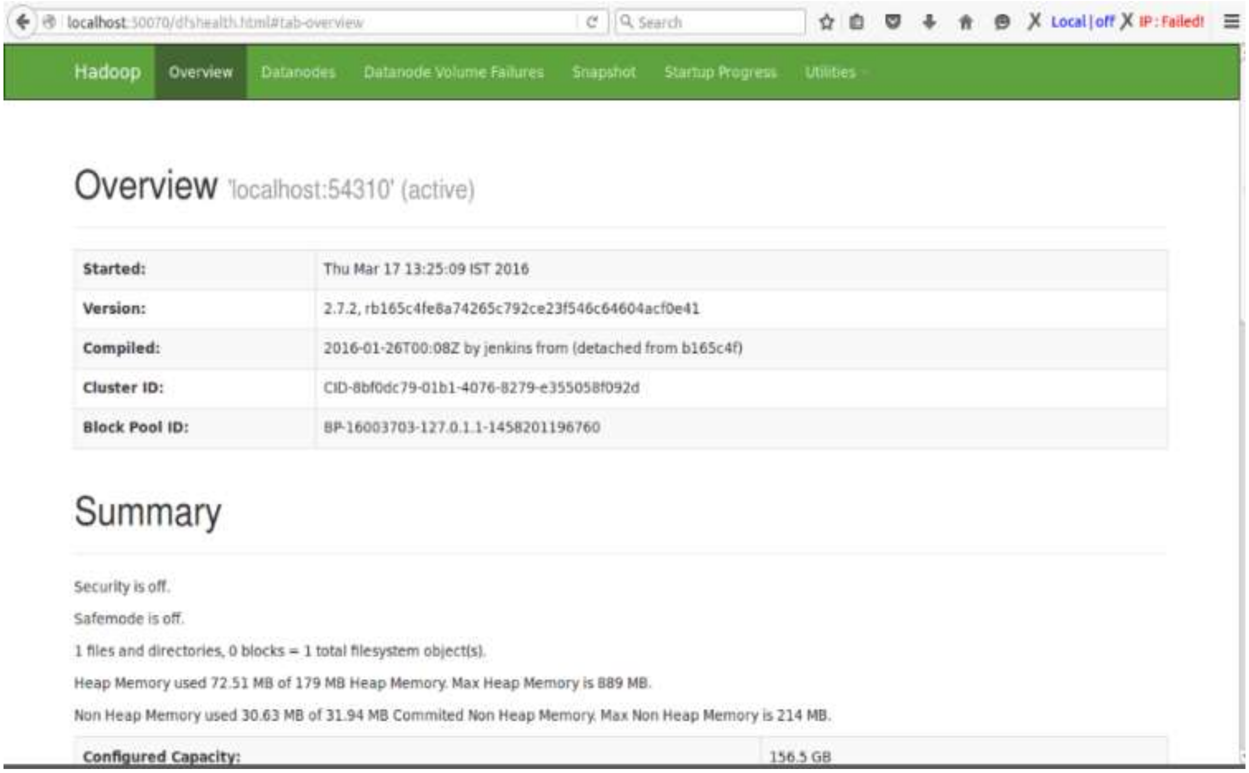
KEYWORDS: Hadoop, Big Data, Configuration, Single node cluster.

INTRODUCTION

Data is increasing day by day at higher rate and with huge volume. Bigdata is a data which is bigger in size. Data is increasing frequently with the increase of development of the organization and technologies. This kind of data may be structured, unstructured and semi-structured. Data may be present in terabytes, petabytes, exabytes etc. Bigdata has mainly three factors are volume, velocity and variety. According to IDC (International Data Cooperation) data is increasing at huge volume rate and 1.8 ZB had been produced in 2011. This cooperation estimated that data will be 9 times in coming years [1]. Due to rapid growth of data in large scale therefore, it becomes very difficult to handle such a huge amount of data using traditional methods. Traditional methods are good to handle structured and lower amount of data very efficiently. But when talk about the Bigdata these traditional methods are not good to handle such a huge amount of data. There are so many challenges regarding Bigdata like storage, computational efficiency, data loss and cost etc. Storage is always a big concern when Bigdata is there. Traditional systems are not well good enough for storing such huge amount of data. Data loss can be happen due to corruption of any part of data and may be due to hardware failure. Therefore, there is a requirement of system that can support huge volume of data very efficiently and also has good storage efficiency. System should be good for data recovery solutions and also cost effective [2] [3].

HADOOP

Hadoop is Highly Archived Distributed Object Oriented Programming. Hadoop is an open source platform to manage or handle large amount of data that is known as Big Data. Hadoop stores large amount of data in distributed fashion. It can store any kind of file format like images, videos, files, folders, xml, html files etc. Hadoop manage the large amount of data in distributed manner where it can store at least three copies of the data. One is the original copy of the data, second is the copy of the original data and the third one is another copy of the original data. These copies are stored in different nodes of cluster. If any one copy is destroyed or any one node in cluster is goes down then another copy of the data will be available for the users or clients. In Hadoop there are master node and other nodes are in the cluster is known as slave nodes [4]. Master node contains all the information about the files those are stored in the other nodes of the clusters. When anyone want to access the data from the slave nodes then it has to get permission from the master node first. When data is transfer to the Hadoop HDFS (Hadoop Distributed File System) then it automatically divides the data into blocks and then distribute to it slave nodes [5]. There may be more than one master node or namenode to avoid the damage from the loss. If one namenode is not working properly then secondary namenode can be used to manage the data in slave nodes. This gives fault tolerance feature for the Hadoop. Hadoop creates 64 Mb blocks in old versions of Hadoop but in new version it comes with default block size of 128 Mb. These block size can be modify using the Hadoop commands at the time of transferring data into HDFS. Hadoop Map Reduce is used to manage the large amount of data [6].



Overview 'localhost:54310' (active)

Started:	Thu Mar 17 13:25:09 IST 2016
Version:	2.7.2, rb165c4fe8a74265c792ce23f546c64604acf0e41
Compiled:	2016-01-26T00:08Z by jenkins from (detached from b165c4f)
Cluster ID:	CID-8bf0dc79-01b1-4076-8279-e355058f092d
Block Pool ID:	BP-16003703-127.0.1.1-1458201196760

Summary

Security is off.

Safemode is off.

1 files and directories, 0 blocks = 1 total filesystem object(s).

Heap Memory used 72.51 MB of 179 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 30.63 MB of 31.94 MB Commited Non Heap Memory. Max Non Heap Memory is 214 MB.

Configured Capacity:	156.5 GB

Fig. 1: Web Browser GUI of Hadoop

COFIGURATION OF HADOOP SINGLE NODE CLUSTER

Hadoop Requirements

RAM	**2 GB**
Operating System	**Ubuntu (Linux)**
Harddisk	**80 GB**
Java	**JDK**

To install Hadoop on the system it is required to install the JDK first. When the Java is installed on the system then Hadoop can be installed and can be run to manage the large amount of the data.

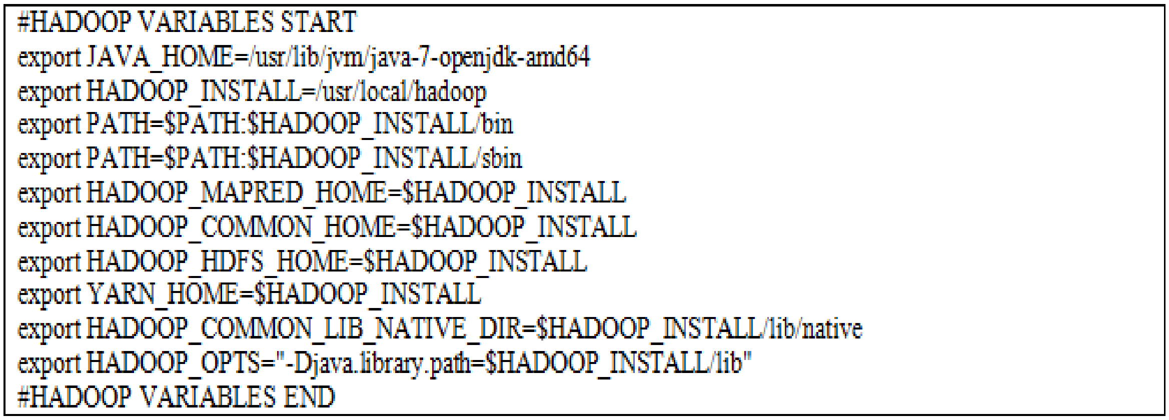
Hadoop Single Node Cluster

To install Hadoop single node cluster it is required to install Java and SSH on the system. Hadoop single node configuration commands are presented in [7].

Step 1: sudo apt-get install default-jdk / to install the java /
Step 2: sudo addgroup hadoop / create hadoop group /
Step 3: sudo adduser --ingroup hadoop hduser / create hadoop user for the installation /
Step 4: sudo apt-get install ssh / to install the ssh /

Step 5: wget http://mirrors.sonic.net/apache/hadoop/common/hadoop-2.6.0/hadoop-2.6.0.tar.gz /download the hadoop
Step 6: tar xvzf hadoop-2.6.0.tar.gz
Step 7: sudo mv * /usr/local/hadoop / move hadoop to the directory/
Step 8: sudo chown -R hduser:hadoop /usr/local/hadoop / gives permission to the user/

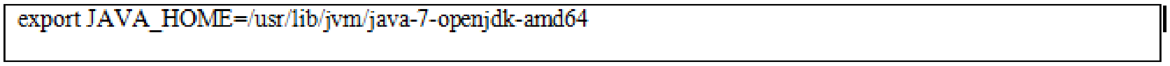
Configuration of Hadoop environment
Step 1: vi ~/.bashrc



```
#HADOOP VARIABLES START
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export HADOOP_INSTALL=/usr/local/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
#HADOOP VARIABLES END
```

Fig. 2: Append these files at the end of bashrc file

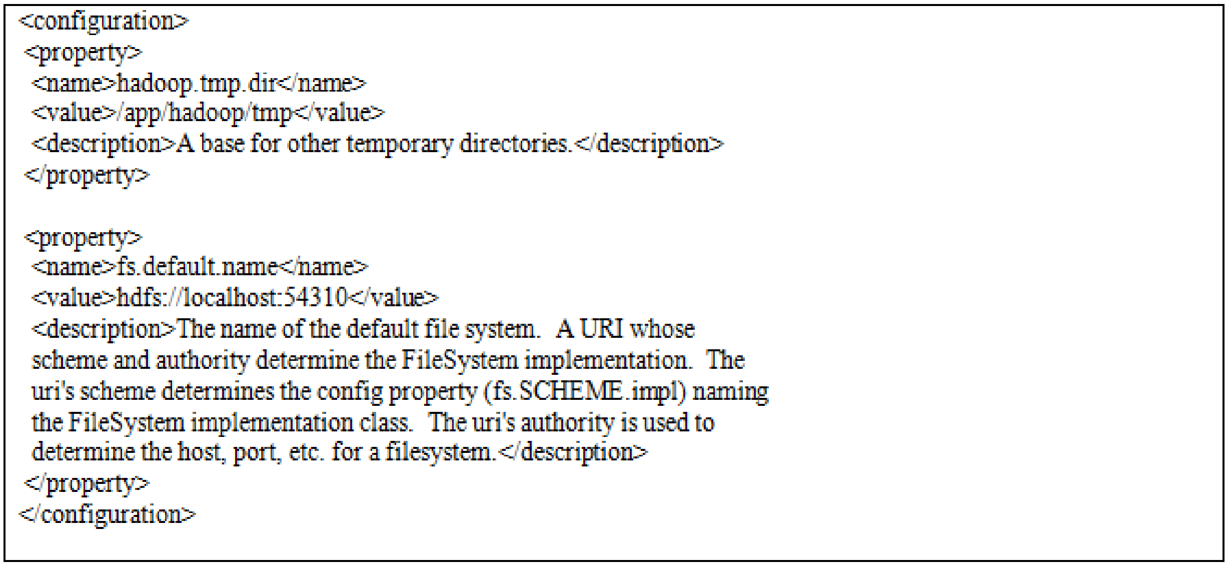
Step 2: vi /usr/local/hadoop/etc/hadoop/hadoop-env.sh



```
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
```

Fig. 3: Change the java path to run the Hadoop

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

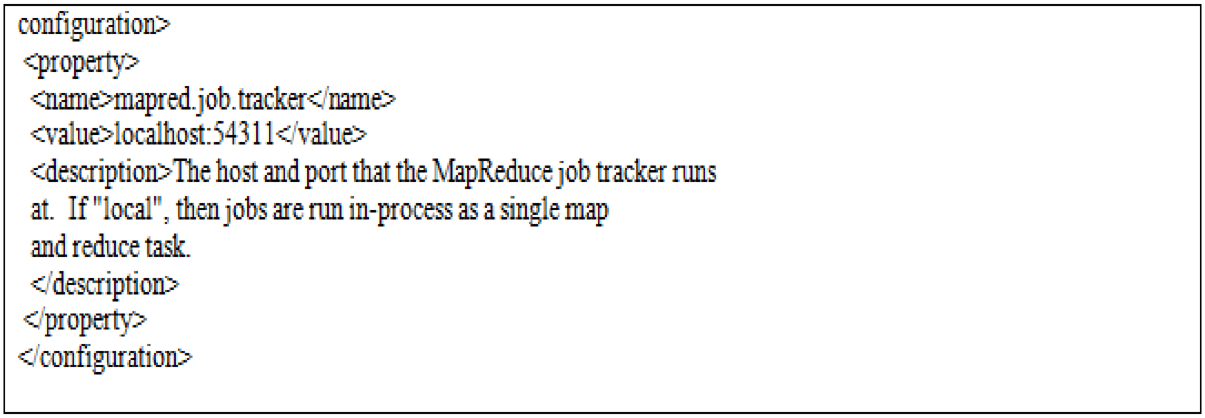


```
<configuration>
 <property>
  <name>hadoop.tmp.dir</name>
  <value>/app/hadoop/tmp</value>
  <description>A base for other temporary directories.</description>
 </property>

 <property>
  <name>fs.default.name</name>
  <value>hdfs://localhost:54310</value>
  <description>The name of the default file system.  A URI whose
  scheme and authority determine the FileSystem implementation.  The
  uri's scheme determines the config property (fs.SCHEME.impl) naming
  the FileSystem implementation class.  The uri's authority is used to
  determine the host, port, etc. for a filesystem.</description>
 </property>
</configuration>
```

Fig. 4: Edit the core-site.xml file

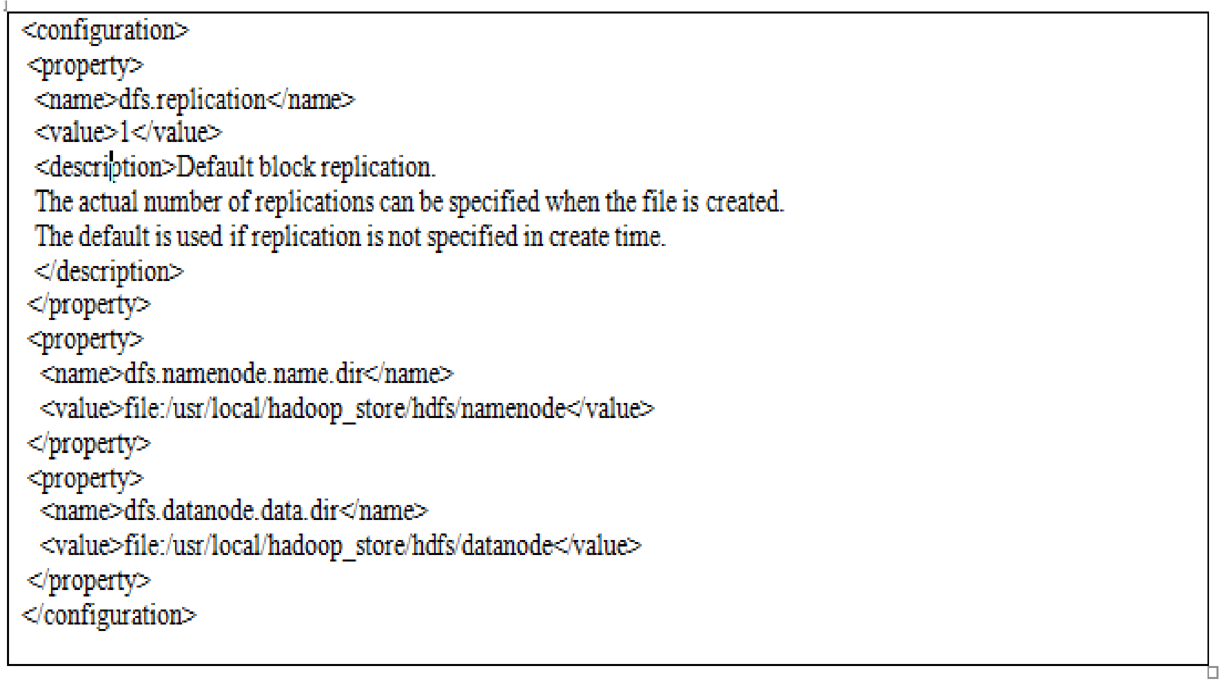
Step 4: vi /usr/local/hadoop/etc/hadoop/mapred-site.xml



```
configuration>
 <property>
  <name>mapred.job.tracker</name>
  <value>localhost:54311</value>
  <description>The host and port that the MapReduce job tracker runs
  at.  If "local", then jobs are run in-process as a single map
  and reduce task.
  </description>
 </property>
</configuration>
```

Fig. 5: Edit the mapred-site.xml file

Step 5: vi /usr/local/hadoop/etc/hadoop/hdfs-site.xml



```
<configuration>
 <property>
  <name>dfs.replication</name>
  <value>1</value>
  <description>Default block replication.
  The actual number of replications can be specified when the file is created.
  The default is used if replication is not specified in create time.
  </description>
 </property>
 <property>
   <name>dfs.namenode.name.dir</name>
   <value>file:/usr/local/hadoop_store/hdfs/namenode</value>
 </property>
 <property>
   <name>dfs.datanode.data.dir</name>
   <value>file:/usr/local/hadoop_store/hdfs/datanode</value>
 </property>
</configuration>
```

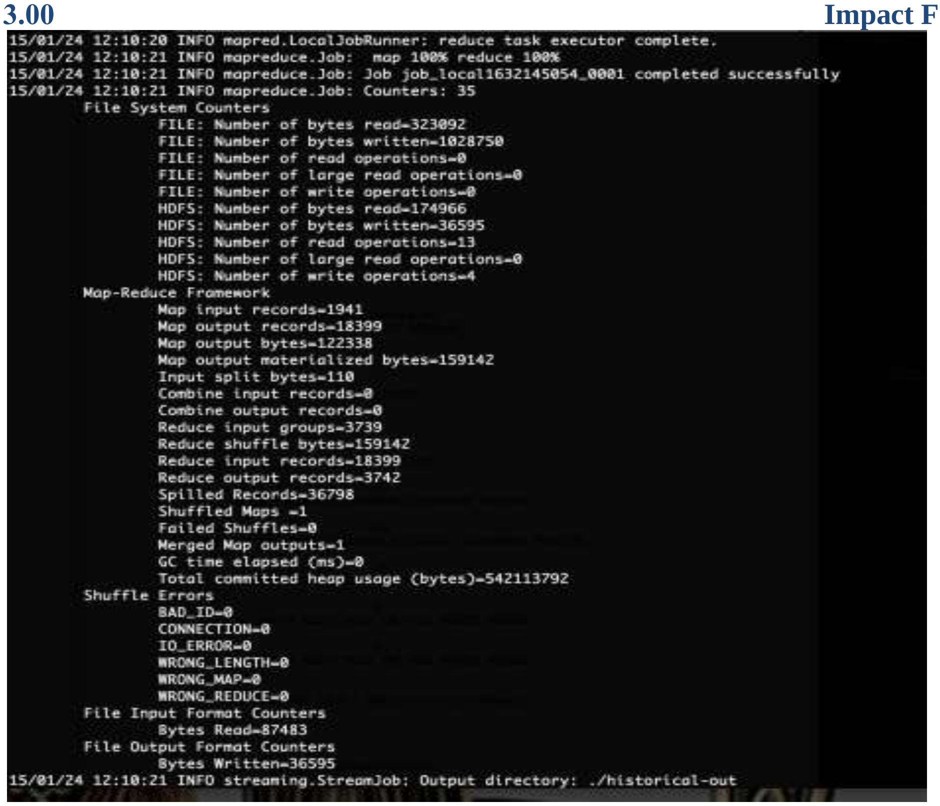
Fig. 6: Edit the hdfs-site.xml file

Running the Hadoop

To start the Hadoop following command can be used
Command: Start-all.sh

To check the state of the processes use this command
Command: jps

To stop the Hadoop following command can be executed
Command Stop-all.sh



```
15/01/24 12:10:20 INFO mapred.LocalJobRunner: reduce task executor complete.
15/01/24 12:10:21 INFO mapreduce.Job:  map 100% reduce 100%
15/01/24 12:10:21 INFO mapreduce.Job: Job job_local1632145054_0001 completed successfully
15/01/24 12:10:21 INFO mapreduce.Job: Counters: 35
        File System Counters
                FILE: Number of bytes read=323092
                FILE: Number of bytes written=1028750
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=174966
                HDFS: Number of bytes written=36595
                HDFS: Number of read operations=13
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=4
        Map-Reduce Framework
                Map input records=1941
                Map output records=18399
                Map output bytes=122338
                Map output materialized bytes=159142
                Input split bytes=110
                Combine input records=0
                Combine output records=0
                Reduce input groups=3739
                Reduce shuffle bytes=159142
                Reduce input records=18399
                Reduce output records=3742
                Spilled Records=36798
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=0
                Total committed heap usage (bytes)=542113792
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=87483
        File Output Format Counters
                Bytes Written=36595
15/01/24 12:10:21 INFO streaming.StreamJob: Output directory: ./historical-out
```

Fig. 7: Running Map Reduce Job

CONCLUSION & FUTURE WORK

This paper presents the knowledge about the Hadoop that is Highly Archived Distributed Object Oriented Programming. Hadoop can be used to manage the large amount of data and provide fault tolerance feature for the storage of the data. Map Reduce is used to handle the large amount of data in HDFS (Hadoop Distributed File System). In this paper configuration of Hadoop single node cluster is explained which can be used to set up single node cluster for testing purpose. There are also commands are described to run and stop the Hadoop. In future multimode cluster setup will be examined and that can be used to test the environment of Hadoop in distributed manner.

REFERENCES

[1] Min Chen, Shiwen Mao and Yunhao Liu, “Big Data: A Survey”, Business Media New York 2014, Springer, pp.171–209.

[2] F. Li, B. C. Ooi, M. T. Özsu and S. Wu, “Distributed Data Management Using MapReduce”, ACM Computing Surveys, 2014, pp. 1-42.

[3] Alaka, Oyedele, Bilal and Akinade, “Bankruptcy Prediction Of Construction Businesses: Towards A Big Data Analytics Approach”, 2015 IEEE First International Conference on Big Data Computing Service and Applications, IEEE 2015, pp.1-5.

[4] Apache Hadoop, http://hadoop.apache.org, Accessed on 12-July-2016.

[5] HDFS, http://searchbusinessanalytics.techtarget.com/definition/Hadoop-Distributed-File-System- HDFS, Accessed on 12-July-2016.

[6] Hadoop mapreduce, http://www.tutorialspoint.com/hadoop/hadoop_mapreduce.htm, Accessed on 22-July-2016.

[7] Big Data hadoop Install on ubuntu single node cluster, http://www.bogotobogo.com/Hadoop/BigData_hadoop_Install_on_ubuntu_single_node_ cluster.php, Accessed on 22-july-2016