IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-3, Issue-9 E-ISSN: 2347-2693

Performance evaluation of Hadoop Distributed File System

In pseudo distributed mode and fully distributed mode

D.Sudheer[1*] and A.Ramana Lakshmi[2]
[1*,2] Department of Computer Science and Engineering,
Prasad V Potluri Siddhartha Institute of Technology, Kanuru, India

www.ijcseonline.org

Received: Aug/22/2015 Revised: Aug/29/2015 Accepted: Sep/24/2015 Published: Sep/30/2015

***Abstract*—** Huge amounts of data are required to build internet search engines and therefore large number of machines to process this entire data. The Apache Hadoop is a framework that allows for the distributed processing of large data sets across clusters of machines. The Hadoop having two modules: 1. Hadoop distributed file system and 2. MapReduce. The Hadoop distributed file system is different from the local normal file system. The hdfs can be implemented as single node cluster and multi node cluster. The large datasets are processed more efficiently by the multi node clusters. By increasing number of nodes the data will be processed faster than the fewer nodes.

***Keywords*—** BigData, HDFS, MapReduce, Namenode, Datanode, Jobtracker, Tasktracker

I. INTRODUCTION

The data uses through internet is rapidly increasing day by day because of the importance of internet of things (IOT) is more than earlier days. As 2013, the World Wide Web has 4 zettabytes of data. Research report from university southern California reports in 2007, humankind successfully sent 1.9 zettabytes of information to broadcast technology. The data is increasing as multiple of bytes as much as users. The most contribution of data in the internet is because of social media. As of January 2014, 74% of online adults are using the social networks. The present technologies are facing problems to process this huge amount of data in effective manner. So the distributed file systems are introduced to the industry. The distributed file systems are storing the data on several machines instead of single machine. The data will achieve parallel processing, so the fastness of data accessing is more than the other file system [1].

II. BIGDATA

The BigData is a keyword to the present data trends. The data is increasing rapidly, but the hardware is not feasible to process the huge amount of data with sufficient velocity. So the user has to wait more time to perform operations on the data.

The main challenges facing BigData are:

- **Velocity:-**User requires immediate result to their actions, so the data should be processed and transferred faster through the network.
- **Volume:-**Volume is the storage issue. Transactional based data contains from years. This data may have huge volume. Though the storage cost is negligible when compare to data processing.
- **Variety:-**There are many verities of data like transactional, textual and multimedia data. The data should be processed any kind of information.
- **Veracity:-**Trust worthiness of data.

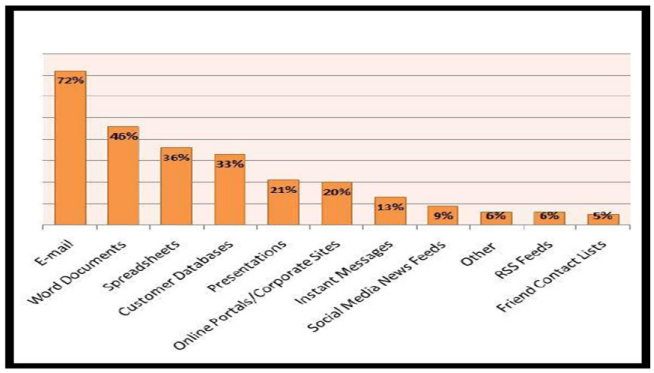


Fig.1. where does the data comes from?

Figure 1 shows in the global marketplace, business suppliers and customers are creating and consuming vast amounts of information. Today the petabytes of data is considered as BigData. The commodity of hardware is failure to process the large datasets. Maintaining high performance computers to process this datasets is costly service. The large datasets are processed with low cost clusters of machines [5]. Here the dataset will distributed to several machines has small chunks. To maintain relations between chunks of data is highly complex task. For that purpose schema less data is used for BigData concept. To achieve high availability, reliability and parallelism to process the data the present trend using Hadoop technology.

III. TECHNOLOGY OVERVIEW

Hadoop is a framework introduced by the apache organization which is used to process the large datasets using commodity of hardware. The Hadoop having two modules: a. HDFS (Hadoop Distributed File System) b. MapReduce.

a. HDFS

HDFS is a block-structured file system: individual files are broken into blocks of a fixed size. These blocks are stored across a cluster of one or more machines with data storage capacity. Individual machines in the cluster are referred to as DataNodes. A file can be made of several blocks, and they are not necessarily stored on the same machine; the target machines which hold each block are chosen randomly on a block-by-block basis. Thus access to a file may require the cooperation of multiple machines, but supports file sizes far larger than a single-machine DFS; individual files can require more space than a single hard drive could hold. The default block size of HDFS is 64MB. The default replication factor for HDFS is 3. For Example, to store 512MB of data we need 8 blocks (as block size is 64MB) and every block has 3 replications [8]. Finally, we should have 24 blocks to store 512MB of data in HDFS.

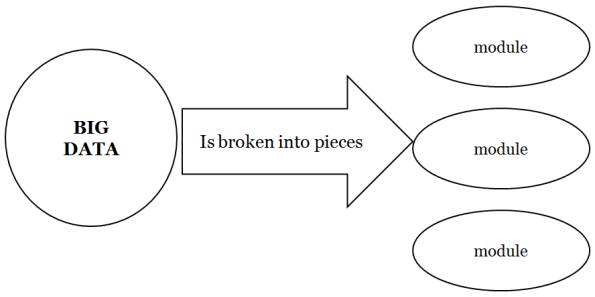


Fig.2. Dividing large datasets into small chunks

Hadoop is designed as master-slave architecture. The namenode acts as the master and all datanodes acts as the slaves. The jobs assigned to slaves through the jobtracker. The datanodes are responsible to store the data in blocks. All slaves are frequently communicated with the master and sending reports periodically. The above divided small-small chunks (Fig.2) are going to store in a blocks of datanodes.

Figure 3 shows the communication between namenode and datanodes in both master and slaves. The slave machines are having tasktracker and datanodes. The master machine having jobtracker and namenode, sometimes master can act as slave also.

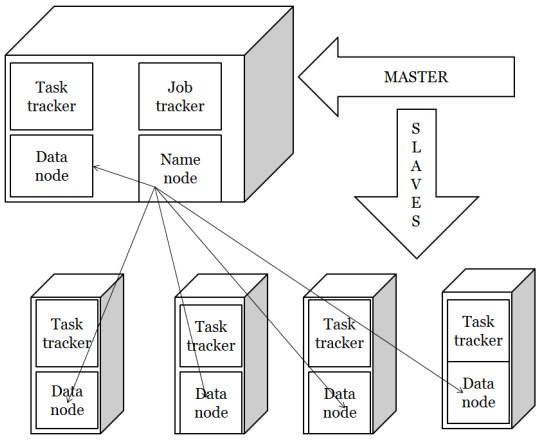


Fig.3. Communication between namenode and datanode

The NameNode and DataNode are designed to run on commodity machines. Namenode will store the metadata of the HDFS. These machines typically run on a Linux operating system (OS). HDFS is built using the Java language; any machine that supports Java can run the NameNode or the DataNode. Usage of the highly portable Java language means that HDFS can be deployed on a wide range of machines. A typical deployment has a dedicated machine that runs only the NameNode. Each of the other machines in the cluster runs one instance of the DataNode.

The jobtracker is another part along with namenode in the master. Jobtracker will assign jobs to the slaves through the tasktrackers. Jobtracker uses the metadata stored in the namenode to assign the jobs. All the demons are continuously communicate each other by the TCP/IP protocol. There is another master to store logs of a namenode periodically that is called as secondary namenode. The secondary namenode act as a mirror to the namenode.

b. MapReduce

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

A MapReduce (Fig.4) job usually splits the input data-set 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 [4].

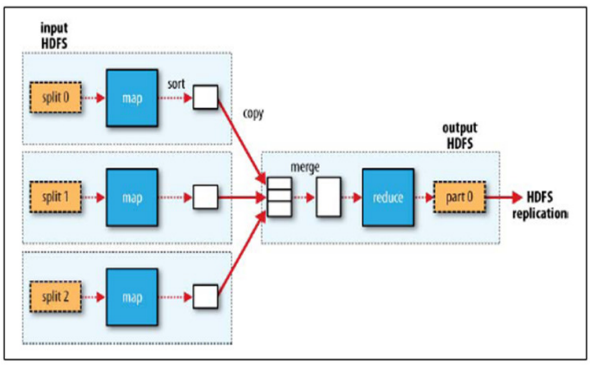


Fig.4. MapReduce

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. The MapReduce is a simple programming paradigm to process large datasets parallel on several systems. The MapReduce program creates as multiple instances depend on the blocks of data. All the instances are running parallel at each block. The combiner will join the all intermediate results and reduce as single output. It is easiest process because of there is no need to travel the data through the network every time. The mapreduce program is having small size. So distribution of mapreduce program to all nodes is easiest task than other techniques. The data never travels through namenode [2]. When user requests to store the data according to the metadata, namenode will give the available block information to the user, then the user directly stores into the blocks.

IV. HADOOP RUNS ON A SINGLE NODE CLUSTER

Hadoop single node cluster runs on single machine. The namenodes and datanodes are performing on the one machine. The installation and configuration steps as given below:

- Openjdk 7 Installation
  Command: sudo apt-get install openjdk-7-jdk.
  Command: java –version
- Install openssh-server
  Command: sudo apt-get install openssh-server
- Create a ssh key
  Command: ssh-keygen -t rsa –P ""
- Moving the key to authorized key
  Command: cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
- Add JAVA_HOME in hadoop-env.sh file
  Command: sudo gedit hadoop-1.2.0/conf/hadoop-env.sh
  Type: export JAVA_HOME=/usr/lib/jvm/java-6-openjdk-i386
- Download and extract Hadoop:
  Command: wget http://archive.apache.org/dist/hadoop/core/hadoop-1.2.0/hadoop-1.2.0.tar.gz
  Command: tar -xvf hadoop-1.2.0.tar.gz
- Change the directory to hadoop-1.2.0
  Command: cd hadoop-1.2.0/
- To check hadoop version give the below command
  Command: bin/hadoop version
- Configure core-site.xml
  Command: sudo gedit conf/core-site.xml

```
<property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:8020</value>
</property>
```

- Configure hdfs-site.xml
  Command: sudo gedit conf/hdfs-site.xml

```
<property>
    <name>dfs.replication</name>
    <value>1</value>
</property>
<property>
    <name>dfs.permissions</name>
    <value>false</value>
</property>
<property>
    <name>dfs.name.dir</name>
    <value>${Hadoop.tmp.dir}/dfs/name</value>
</property>
```

- Configure mapred-site.xml
  Command: sudo gedit conf/mapred-site.xml

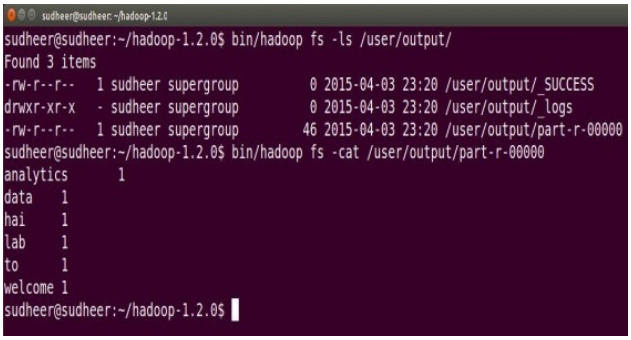
```
<property>
    <name>mapred.job.tracker</name>
    <value>localhost:8021</value>
</property>
```

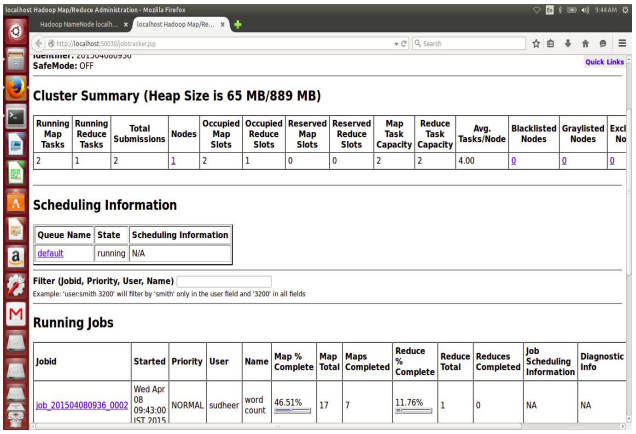
- Format the name node
  Command: bin/hadoop namenode -format
- Start the namenode, datanode
  Command: bin/start-dfs.sh
- Start the task tracker and job tracker
  Command: bin/start-mapred.sh
- To check if Hadoop started correctly
  Command: jps

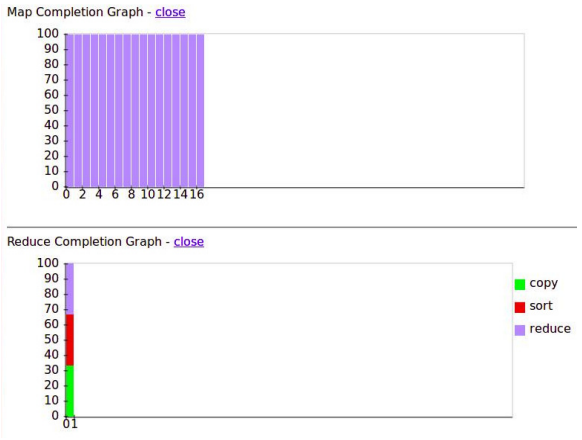
```
sudheer@sudheer: ~
sudheer@sudheer:~$ jps
2912 TaskTracker
2474 DataNode
2657 SecondaryNameNode
3084 Jps
2236 NameNode
2742 JobTracker
sudheer@sudheer:~$
```

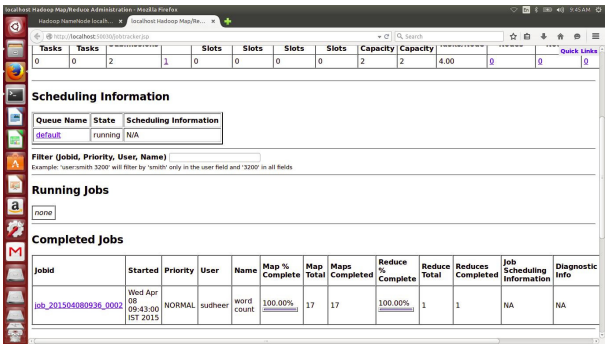
- Create input directory on hdfs
  Command: bin/hadoop fs -mkdir /user/input
- Put the file from local file system to hdfs
  Command: bin/hadoop fs -put input/file.txt /user/input
- Apply the WordCount program on input directory
  Command: bin/hadoop jar wc.jar WordCount /user/input /user/output

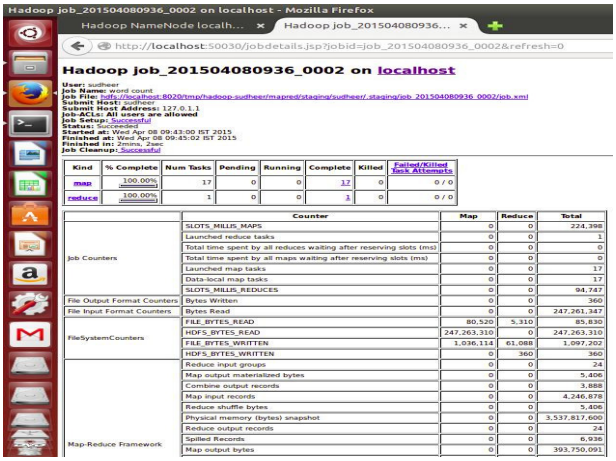
- To see the output
  Command: bin/hadoop fs -ls /user/output/
  Command: bin/hadoop fs -cat /user/output/part-r-00000











V. HADOOP RUNS ON A MULTI NODE CLUSTER

All the demons like namenodes and datanodes are runs on different machines. The data will replicate according to the replication factor in client machines. The secondary namenode will store the mirror images of namenode periodically. The namenode having the metadata where the blocks are stored and number of replicas in the client machines [9]. The slaves and master communicate each other periodically. The configurations of multinode cluster are given below:

Openssh Server Installation

Hadoop requires openssh-server for creating passwordless ssh environment to manage hadoop services from namenode to datanodes. The namenode seldom communicates with the datanodes. Instead datanodes communicate with namenode to send heartbeat and build block locations with the help of block reports sent by datanodes to namenodes. The datanode to namenode communication is based on simple socket communications.

sudheer@nn:~$ sudo apt-get install openssh-server

Passwordless Ssh Configuration

sudheer@nn:~$ssh-keygen -t rsa
sudheer@nn:~$ssh-copy-id -i ~/.ssh/id_rsa.pub sudheer@nn2
sudheer@nn:~$ssh-copy-id -i ~/.ssh/id_rsa.pub sudheer@dn1
sudheer@nn:~$ssh-copy-id -i ~/.ssh/id_rsa.pub sudheer@dn2
sudheer@nn:~$ssh-copy-id -i ~/.ssh/id_rsa.pub sudheer@dn3

NOTE: Verify the passwordless ssh environment from namenode to all datanodes as "sudheer" user.

Openjdk 7 Installation

sudheer@nn:~$ sudo apt-get install openjdk-7-jdk

NOTE: Openjdk installation has to be done on all nodes. After the installation verify the java installation by running the below command.
sudheer@nn:~$ java -version

Hadoop Installation

- Download and extract Hadoop:
  Command: wget http://archive.apache.org/dist/hadoop/core/hadoop-1.2.0/hadoop-1.2.0.tar.gz
  Command: tar -xvf hadoop-1.2.0.tar.gz

Now move it to /usr/local location.
sudheer@nn:~$ sudo mv hadoop-1.2.1 /usr/local/hadoop
Note: Installation of hadoop has to be done on all nodes.

User Environment Configuration

Now we will configure the environment for user "sudheer". As we are already logged into the sudheer home directory, we need to append the .bashrc file with the below mentioned variables to set the user environment.
sudheer@nn:~$ sudo gedit .bashrc
###HADOOP ENVIRONMENT###
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64/
export PATH=$PATH:$JAVA_HOME/bin
export HADOOP_PREFIX=/usr/local/hadoop/
export PATH=$PATH:$HADOOP_PREFIX/bin
export PATH=$PATH:$HADOOP_PREFIX/sbin

Execute the .bashrc file.

sudheer@nn:~$ exec bash

To verify the defined hadoop variables execute the below command to check the hadoop version that is installed.
sudheer@nn:~$ hadoop version

NOTE: User environment has to be configured on all nodes. It is very important to define the hosts in /etc/hosts file on all nodes.
sudheer@nn:~$sudo gedit /etc/hosts
192.168.8.30 nn
192.168.8.31 nn2
192.168.8.32 dn1
192.168.8.33 dn2
192.168.8.34 dn3

Hadoop Configuration

sudheer@nn:~$ sudo gedit /usr/local/hadoop/conf/hadoop-env.sh
Type: export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64

After defining the variables we need to create the hadoop-log directory in /var/log location on all nodes with the required "sudheer" user ownership and permissions.
sudheer@nn:~$ sudo mkdir /var/log/hadoop-log
sudheer@nn:~$ sudo chown -R sudheer:hadoop /var/log/hadoop-log
NOTE: It is necessary to modify hadoop-env.sh with same parameters across all nodes in the cluster and /var/log/hadoop-log directory has to be created on all nodes with required "sudheer" user ownership and permissions

- Configure core-site.xml
  Command: sudo gedit /usr/local/Hadoop/conf/core-site.xml
  <property>
  <name>fs.default.name</name>
  <value>hdfs://nn:10001</value>
  </property>
  <property>
  <name>Hadoop.tmp.dir</name>
  <value>/usr/local/Hadoop/tmp</value>
  </property>
  Command:sudo mkdir /usr/local/Hadoop/tmp
- Configure hdfs-site.xml
  Command: sudo gedit /usr/local/Hadoop/conf/hdfs-site.xml
  <property>
  <name>dfs.replication</name>
  <value>3</value>
  </property>
  <property>
  <name>dfs.name.dir</name>
  <value>${Hadoop.tmp.dir}/dfs/name</value>
  </property>
- Configure mapred-site.xml
  Command: sudo gedit /usr/local/Hadoop/conf/mapred-site.xml
  <property>
  <name>mapred.job.tracker</name>
  <value>localhost:10002</value>
  </property>

Similary in all datanodes we need to create hdfs and data directories.
sudheer@dn1:~$ sudo mkdir -p /hdfs/data
sudheer@dn1:~$ sudo chown -R sudheer:hadoop /hdfs

conf/masters

The masters file contains the location where the secondary namenode daemon would start.

```
sudheer@nn:~$ sudo gedit /usr/local/hadoop/conf/masters
nn2
```

NOTE: We have configured separate machine for the secondary namenode hence we are defining secondary namenode explicitly here. There is no need to duplicate this file to all nodes.

conf/slaves

The slaves file contains the list of datanodes where the datanode and task-tracker daemons will run.

```
sudheer@nn:~$ vi /usr/local/hadoop/conf/slaves
dn1
dn2
dn3
```

NOTE: The file should contain one entry per line. We can also mention "namenode" and "secondary namenode" hostname in this file if we want to run datanode and task-tracker daemons on "namenode" and "secondary namenode" too.

Starting Hadoop Cluster

Formatting HDFS via namenode

Before we start our cluster we will format the HDFS via namenode. Formatting the namenode means to initialize the directory specified in "dfs.name.dir" and "dfs.data.dir" parameter in hdfs-site.xml file.

```
sudheer@nn1:~$ hadoop namenode -format
```

NOTE: We need to format the namenode only the first time we setup hadoop cluster. Formatting a running cluster will destroy all the existing data. After executing the format command, it will prompt for confirmation where we need to type "Y" as it is case-sensitive.

Starting the Multi-Node Hadoop Cluster

Starting a hadoop cluster can be done by a single command mentioned below.

```
sudheer@nn:~$ start-all.sh
```

This command will first start the HDFS daemons. The namenode daemon is started on namenode and datanode daemon is started on all datanodes. The secondary namenode daemon is also started on secondary namenode. In the second phase it will start theMapReduce daemons, job-tracker on namenode and task-trackers on datanodes.

JPS (Java Virtual Machine Process Status Tool) is used to verify that all the daemons are running on their respective nodes, we will execute "jps" command to see the java processes running.

VI. CONCLUSION

In single node cluster all the demons are run on a single machine that means the burden java processes is heavy. The large datasets cannot efficiently process by the HDFS in a single node cluster because resource utilization is more than supply. Whereas in multi node cluster all demons are run on different machines. So the processing of data will be faster and efficient than the single node cluster. So that we conclude that Hadoop distributed file system will achieve high availability, reliability, fault tolerant and more features in multi node cluster than the single node cluster.

REFERENCES

[1] Linthala Srinithya, Dr. G. Venkata Rami Reddy, "Performance Evaluation of Hadoop Distributed file System and Local File System" in IJSR ISSN: 2319-7064, Volume 3 Issue 9, September 2014.

[2] Liu Liu, Jiangtao Yin, Lixin Gao, "Efficient Social Network Data Query Processing on MapReduce" ACM August 16, 2013.

[3] E. Dede, M. Govindaraju, D. Gunter, R. Canon, L. Ramakrishnan,"Performance Evaluation of a MongoDB and Hadoop Platform for Scientific Data Analysis".

[4] Christos Doulkeridis, Kjetil Norvag, "A Survey of Large-Scale Analytical Query Processing in MapReduce".

[5] Stephen Kaisler, Frank Armour, J. Alberto Espinosa, William Money, "Big Data: Issues and Challenges Moving Forward".

[6] Hadoop The Definitive Guide,©2012, Tom White.

[7] K.Udhaya Malar,D.Ragupathi and G.M.Prabhu, "The Hadoop Dispersed File system: Balancing Movability And Performance", IJCSE, Volume-2, Issue-9, september-2014.

[8] Apache Hadoop. http://hadoop.apache.org/ Tuesday, June 23, 2015.

[9] Hadoop multinode cluster configuration, http://hashprompt.blogspot.in/2014/06/multi-node-hadoop-cluster-on-ubuntu-1404.html, Wednesday, August 19, 2015.

AUTHORS PROFILE

D.Sudheer is presently M.Tech Student, Dept. of Computer Science & Engineering, Prasad V Potluri Siddhartha Institute of technology (Autonomous), kanuru, India.

A.Ramana Lakshmi is presently Associate Professor, Dept. of Computer Science & Engineering, Prasad V Potluri Siddhartha Institute of technology (Autonomous), kanuru, India.