Getting Hadoop Up and Running in a cluster

Experiment – 1

AIM: To set up hadoop on standalone machine.

Installing Java

Hadoop framework is written in Java!!

# Update the source list

```
$ cd ~
```

# The OpenJDK project is the default version of Java that is provided from a supported Ubuntu repository.

```
$ sudo apt-get update
```

# Install oracle-java8 in your system

```
$ sudo apt-get install oracle-java8-installer
```

# Check the java version

```
$ java -version
```

```
java version "1.8.0_91"
Java(TM) SE Runtime Environment (build 1.8.0_91-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.91-b14, mixed mode)
```

Adding a dedicated Hadoop user

# Add a group named hadoop

```
$ sudo addgroup hadoop
```

```
Adding group `hadoop' (GID 1002) ...
Done.
```

Add a user named hduser in the group hadoop

```
$ sudo adduser --ingroup hadoop hduser
```

```
Adding user `hduser' ...
Adding new user `hduser' (1001) with group `hadoop' ...
Creating home directory `/home/hduser' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for hduser
```

```
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] Y
```

Add hduser to sudo

```
$ sudo adduser hduser sudo
```

```
[sudo] password for lahari:
Adding user `hduser' to group `sudo' ...
Adding user hduser to group sudo
Done.
```

Installing SSH

ssh has two main components:

ssh : The command we use to connect to remote machines - the client.

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 :

```
# Install ssh on our machine.
$ sudo apt-get install ssh
# If we get something similar to the following, we can think it is setup properly:
$ which ssh
/usr/bin/ssh
$ which sshd
/usr/sbin/sshd
```

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.

```
$ sudo su hduser
Password:
$ ssh-keygen -t rsa -P ""
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hduser/.ssh/id_rsa):
Created directory '/home/hduser/.ssh'.
Your identification has been saved in /home/hduser/.ssh/id_rsa.
Your public key has been saved in /home/hduser/.ssh/id_rsa.pub.
The key fingerprint is:
50:6b:f3:fc:0f:32:bf:30:79:c2:41:71:26:cc:7d:e3 hduser@laptop
The key's randomart image is:
+--[ RSA 2048]----+
|       .oo.o    |
|      . .o=. o  |
|     . + .  o . |
|      o =    E  |
|       S +     |
|        . +    |
|         O +   |
|          O o  |
|           o.. |
+-----------------+
# Add the newly created key to the list of authorized keys so that Hadoop can use ssh without prompting for a password.
$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
# check if ssh works
$ ssh localhost
Welcome to Ubuntu 16.04 LTS (GNU/Linux 4.4.0-28-generic x86_64)
 * Documentation:  https://help.ubuntu.com/
205 packages can be updated.
7 updates are security updates.
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
```

```
applicable law.
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
Last login: Wed Jul 13 15:43:23 2016 from 127.0.0.1
```

Install Hadoop

# Download latest version of hadoop

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

# Extract hadoop files from the .tar zip file

```
$ tar xvzf hadoop-2.6.0.tar.gz
```

# Move the Hadoop installation to the **/usr/local/** directory

```
$ sudo mv hadoop-2.6.0 /usr/local/hadoop-2.6.0
[sudo] password for hduser:
```

# Change ownership to hadoop directory

```
$ sudo chown -R hduser:hadoop /usr/local/hadoop-2.6.0
```

Setup Configuration Files

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

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:

```
$ update-alternatives --config java
```

# Open the .bashrc file in editor

```
$ sudo gedit ~/.bashrc
```

# Append the following to the end of **~/.bashrc**

```
#HADOOP VARIABLES START
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
export HADOOP_HOME=/usr/local/hadoop-2.6.0
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
export PATH=$JAVA_HOME/bin:$PATH
export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar
#HADOOP VARIABLES END
```

# Update the .bashrc file

```
$ source ~/.bashrc
```

2. hadoop-env.sh

# Open the hadoop-env.sh file in editor

```
$ sudo gedit /usr/local/hadoop-2.6.0/etc/hadoop/hadoop-env.sh
```

# Set JAVA_HOME by modifying hadoop-env.sh file.

```
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
```

3. 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.

Create a hadoop temporary directory

```
$ sudo mkdir -p /app/hadoop/tmp
```

# Change ownership to the hadoop temp folder

```
$ sudo chown hduser:hadoop /app/hadoop/tmp
```

# Open the core-site.xml file in editor

```
$ sudo gedit /usr/local/hadoop-2.6.0/etc/hadoop/core-site.xml
```

# Add the following <property> tags in the <configuration> tags.

```
<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>
</property>
</configuration>
```

4. mapred-site.xml

By default, the **/usr/local/hadoop-2.6.0/etc/hadoop/** folder contains **/usr/local/hadoop/etc/hadoop-2.6.0/mapred-site.xml.template** file.

Rename / Copy the file with the name **mapred-site.xml**

```
$ sudo cp /usr/local/hadoop-2.6.0/etc/hadoop/mapred-site.xml.template /usr/local/hadoop-2.6.0/etc/hadoop/mapred-site.xml
```

The **mapred-site.xml** file is used to specify which framework is being used for MapReduce.

Open the mapred-site.xml file in editor

```
$ sudo gedit /usr/local/hadoop-2.6.0/etc/hadoop/mapred-site.xml
```

# Add the following <property> tags in the <configuration> tags.

```
<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>
```

5. 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.

Create namenode directory

```
$ sudo mkdir -p /usr/local/hadoop_store/hdfs/namenode
```

# Create datanode diectory

```
$ sudo mkdir -p /usr/local/hadoop_store/hdfs/datanode
```

# Change ownership to the hadoop_store folder

```
$ sudo chown -R hduser:hadoop /usr/local/hadoop_store/hdfs
```

# Open the hdfs-site.xml file in editor

```
$ sudo gedit /usr/local/hadoop-2.6.0/etc/hadoop/hdfs-site.xml
```

# Add the following <property> tags in the <configuration> tags.

```
<configuration>
 <property>
  <name>dfs.replication</name>
  <value>1</value>
 </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>
```

6. yarn-site.xml

# Open the yarn-site.xml file in editor

```
$ sudo gedit /usr/local/hadoop-2.6.0/etc/hadoop/yarn-site.xml
```

# Add the following <property> tags in the <configuration> tags.

```
<configuration>
 <property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
 </property>
 <property>
  <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
  <value>org.apache.hadoop.mapred.ShuffleHandler</value>
 </property>
</configuration>
```

```
$ hadoop version
$ which hadoop
```

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.

Format the hadoop FileSystem

```
$ hadoop namenode -format
```

```
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
15/04/18 14:43:03 INFO namenode.NameNode: STARTUP_MSG:
/**********************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = laptop/192.168.1.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.6.0
STARTUP_MSG:   classpath = /usr/local/hadoop/etc/hadoop
...
STARTUP_MSG:   java = 1.7.0_65
**********************************************************/
15/04/18 14:43:03 INFO namenode.NameNode: registered UNIX signal handlers for [TERM, HUP, INT]
15/04/18 14:43:03 INFO namenode.NameNode: createNameNode [-format]
15/04/18 14:43:07 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Formatting using clusterid: CID-e2f515ac-33da-45bc-8466-5b1100a2bf7f
15/04/18 14:43:09 INFO namenode.FSNamesystem: No KeyProvider found.
15/04/18 14:43:09 INFO namenode.FSNamesystem: fsLock is fair:true
15/04/18 14:43:10 INFO blockmanagement.DatanodeManager: dfs.block.invalidate.limit=1000
15/04/18 14:43:10 INFO blockmanagement.DatanodeManager: dfs.namenode.datanode.registration.ip-hostname-check=true
15/04/18 14:43:10 INFO blockmanagement.BlockManager: dfs.namenode.startup.delay.block.deletion.sec is set to 000:00:00:00.000
15/04/18 14:43:10 INFO blockmanagement.BlockManager: The block deletion will start around 2015 Apr 18 14:43:10
15/04/18 14:43:10 INFO util.GSet: Computing capacity for map BlocksMap
15/04/18 14:43:10 INFO util.GSet: VM type       = 64-bit
```

```
15/04/18 14:43:10 INFO util.GSet: 2.0% max memory 889 MB = 17.8 MB
15/04/18 14:43:10 INFO util.GSet: capacity      = 2^21 = 2097152 entries
15/04/18 14:43:10 INFO blockmanagement.BlockManager: dfs.block.access.token.enable=false
15/04/18 14:43:10 INFO blockmanagement.BlockManager: defaultReplication         = 1
15/04/18 14:43:10 INFO blockmanagement.BlockManager: maxReplication             = 512
15/04/18 14:43:10 INFO blockmanagement.BlockManager: minReplication             = 1
15/04/18 14:43:10 INFO blockmanagement.BlockManager: maxReplicationStreams      = 2
15/04/18 14:43:10 INFO blockmanagement.BlockManager: shouldCheckForEnoughRacks  = false
15/04/18 14:43:10 INFO blockmanagement.BlockManager: replicationRecheckInterval = 3000
15/04/18 14:43:10 INFO blockmanagement.BlockManager: encryptDataTransfer        = false
15/04/18 14:43:10 INFO blockmanagement.BlockManager: maxNumBlocksToLog          = 1000
15/04/18 14:43:10 INFO namenode.FSNamesystem: fsOwner             = hduser (auth:SIMPLE)
15/04/18 14:43:10 INFO namenode.FSNamesystem: supergroup          = supergroup
15/04/18 14:43:10 INFO namenode.FSNamesystem: isPermissionEnabled = true
15/04/18 14:43:10 INFO namenode.FSNamesystem: HA Enabled: false
15/04/18 14:43:10 INFO namenode.FSNamesystem: Append Enabled: true
15/04/18 14:43:11 INFO util.GSet: Computing capacity for map INodeMap
15/04/18 14:43:11 INFO util.GSet: VM type       = 64-bit
15/04/18 14:43:11 INFO util.GSet: 1.0% max memory 889 MB = 8.9 MB
15/04/18 14:43:11 INFO util.GSet: capacity      = 2^20 = 1048576 entries
15/04/18 14:43:11 INFO namenode.NameNode: Caching file names occuring more than 10 times
15/04/18 14:43:11 INFO util.GSet: Computing capacity for map cachedBlocks
15/04/18 14:43:11 INFO util.GSet: VM type       = 64-bit
15/04/18 14:43:11 INFO util.GSet: 0.25% max memory 889 MB = 2.2 MB
15/04/18 14:43:11 INFO util.GSet: capacity      = 2^18 = 262144 entries
15/04/18 14:43:11 INFO namenode.FSNamesystem: dfs.namenode.safemode.threshold-pct =
0.9990000128746033
15/04/18 14:43:11 INFO namenode.FSNamesystem: dfs.namenode.safemode.min.datanodes = 0
15/04/18 14:43:11 INFO namenode.FSNamesystem: dfs.namenode.safemode.extension     = 30000
15/04/18 14:43:11 INFO namenode.FSNamesystem: Retry cache on namenode is enabled
15/04/18 14:43:11 INFO namenode.FSNamesystem: Retry cache will use 0.03 of total heap and
retry cache entry expiry time is 600000 millis
15/04/18 14:43:11 INFO util.GSet: Computing capacity for map NameNodeRetryCache
15/04/18 14:43:11 INFO util.GSet: VM type       = 64-bit
15/04/18 14:43:11 INFO util.GSet: 0.029999999329447746% max memory 889 MB = 273.1 KB
```

```
15/04/18 14:43:11 INFO util.GSet: capacity      = 2^15 = 32768 entries
15/04/18 14:43:11 INFO namenode.NNConf: ACLs enabled? false
15/04/18 14:43:11 INFO namenode.NNConf: XAttrs enabled? true
15/04/18 14:43:11 INFO namenode.NNConf: Maximum size of an xattr: 16384
15/04/18 14:43:12 INFO namenode.FSImage: Allocated new BlockPoolId: BP-130729900-
192.168.1.1-1429393391595
15/04/18 14:43:12 INFO common.Storage: Storage directory
/usr/local/hadoop_store/hdfs/namenode has been successfully formatted.
15/04/18 14:43:12 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with
txid >= 0
15/04/18 14:43:12 INFO util.ExitUtil: Exiting with status 0
15/04/18 14:43:12 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at laptop/192.168.1.1
************************************************************/
```

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.

Starting Hadoop

# Start the hadoop namenodes

```
$ start-dfs.sh
```

```
Starting namenodes on [localhost]
localhost: starting namenode, logging to /usr/local/hadoop-2.6.0/logs/hadoop-hduser-namenode-
laharig5080.out
localhost: starting datanode, logging to /usr/local/hadoop-2.6.0/logs/hadoop-hduser-datanode-
laharig5080.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop-2.6.0/logs/hadoop-hduser-
secondarynamenode-laharig5080.out
16/07/14 01:02:08 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your
platform... using builtin-java classes where applicable
```

# Start the hadoop yarn daemons

```
$ start-yarn.sh
```

```
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop-2.6.0/logs/yarn-hduser-resourcemanager-laharig5080.out
localhost: starting nodemanager, logging to /usr/local/hadoop-2.6.0/logs/yarn-hduser-nodemanager-laharig5080.out
```

# Check whether hadoop is running in our system or not

```
$jps
```

```
2464 NameNode
2961 ResourceManager
2583 DataNode
3082 NodeManager
3372 Jps
```

Stopping Hadoop

# Stop the hadoop namenodes

```
$ stop-dfs.sh
```

```
Stopping namenodes on [localhost]
localhost: stopping namenode
localhost: stopping datanode
Stopping secondary namenodes [0.0.0.0]
0.0.0.0: no secondarynamenode to stop
16/07/14 01:05:23 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
```

Experiment – 2

AIM: To implement Word count Map Reduce program using standalone hadoop.

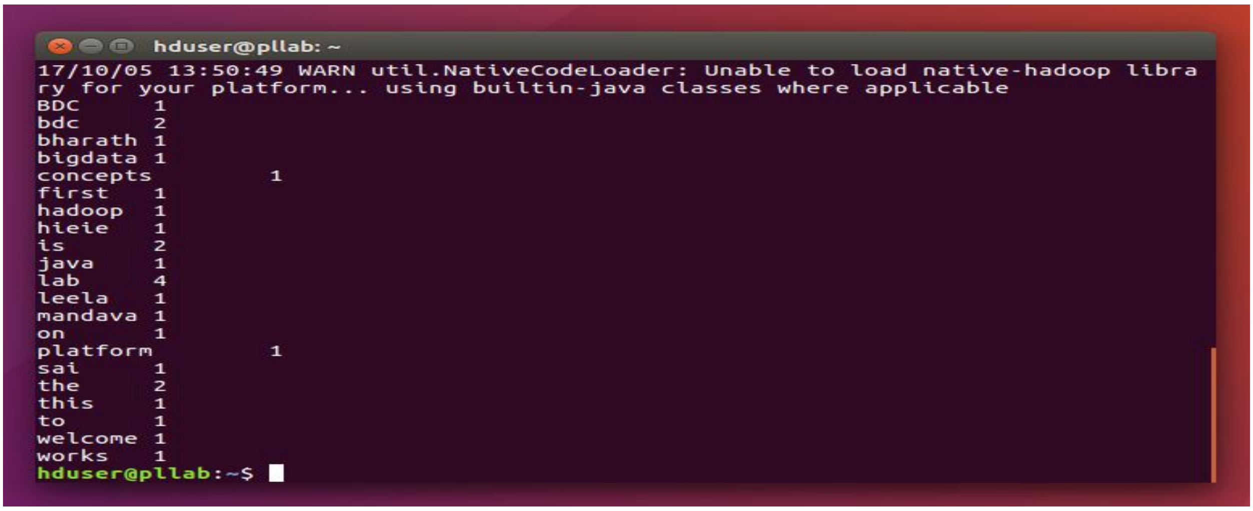
Program :

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class WordCount1 {
  public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable>{
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }
  public static class IntSumReducer extends Reducer <Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();
    public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException,
InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }
  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount1.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Steps to execute the Hadoop application:

```
export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar
hadoop com.suntools.javac.Main WordCount1.java
jar cf wcc.jar WordCount1*.class
hadoop fs -mkdir -p /user/input
hadoop fs -copyFromLocal input  /user/input1
hadoop jar wcc.jar WordCount1 /user/input1 /user/output1
hadoop fs  -cat /user/output1/part-r-00000
```

OUTPUT:



```
17/10/05 13:50:49 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
BDC     1
bdc     2
bharath 1
bigdata 1
concepts        1
first   1
hadoop  1
hieie   1
is      2
java    1
lab     4
leela   1
mandava 1
on      1
platform        1
sai     1
the     2
this    1
to      1
welcome 1
works   1
hduser@pllab:~$ 
```

Experiment – 3

AIM: To implement Wordcount Map Reduce program with combiner step using hadoop.

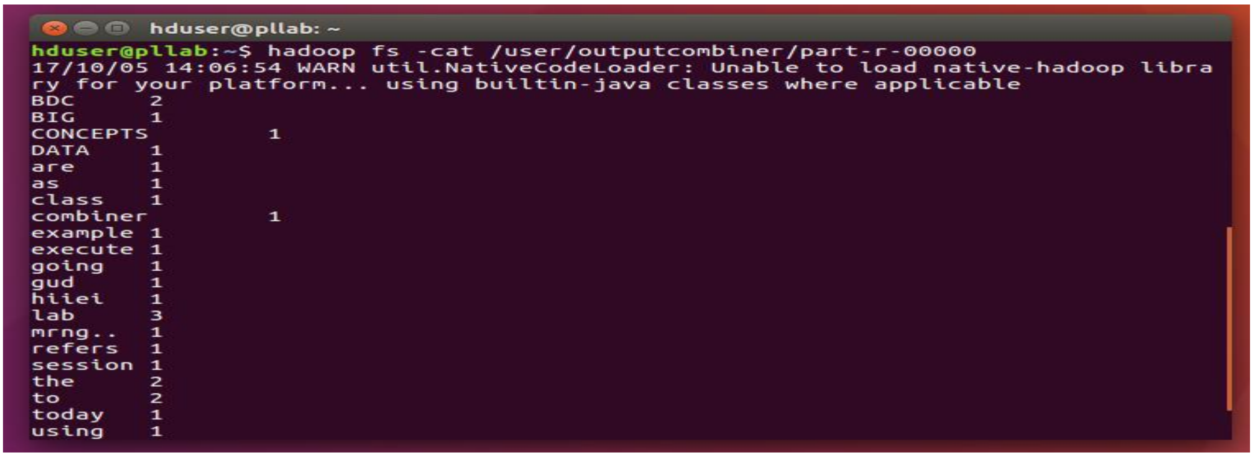
Program :

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class WordCount2 {
  public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable>{
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }
  public static class IntSumReducer extends Reducer <Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();
    public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException,
InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }
  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount2.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Steps to execute the Hadoop application:

- export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar
- hadoop com.suntools.javac.Main WordCount2.java
- jar cf wcc.jar WordCount2*.class
- hadoop fs -mkdir -p /user/input
- hadoop fs -copyFromLocal input /user/input1
- hadoop jar wcc.jar WordCount2 /user/input1 /user/output1
- Hadoop fs -cat /user/output1/part-r-00000

OUTPUT:



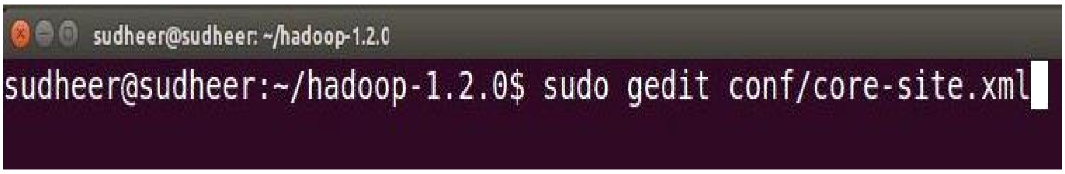
```
hduser@pllab: ~
hduser@pllab:~$ hadoop fs -cat /user/outputcombiner/part-r-00000
17/10/05 14:06:54 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
BDC     2
BIG     1
CONCEPTS        1
DATA    1
are     1
as      1
class   1
combiner        1
example 1
execute 1
going   1
gud     1
hiiei   1
lab     3
mrng..  1
refers  1
session 1
the     2
to      2
today   1
using   1
```

Experiment – 4

AIM: To set up HDFS.

Configure core-site.xml

Command: sudo gedit conf/core-site.xml



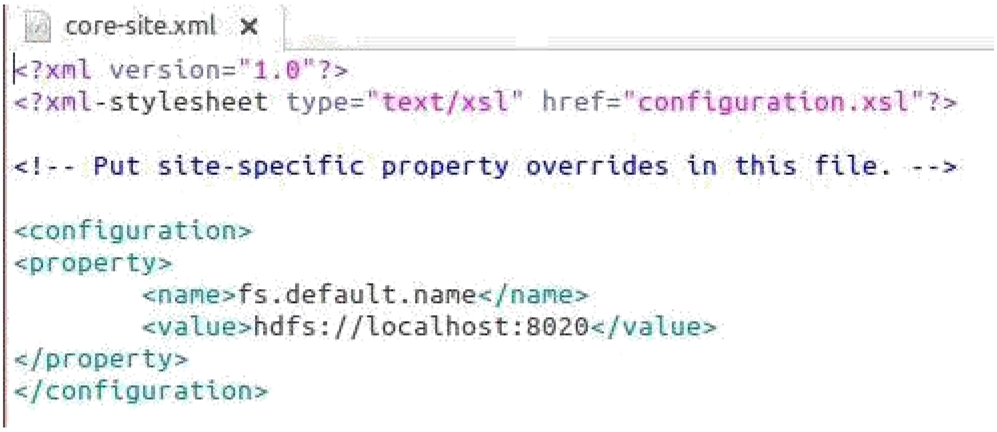
```
sudheer@sudheer: ~/hadoop-1.2.0
sudheer@sudheer:~/hadoop-1.2.0$ sudo gedit conf/core-site.xml
```

<property>

<name>fs.default.name</name>

<value>hdfs://localhost:8020</value>

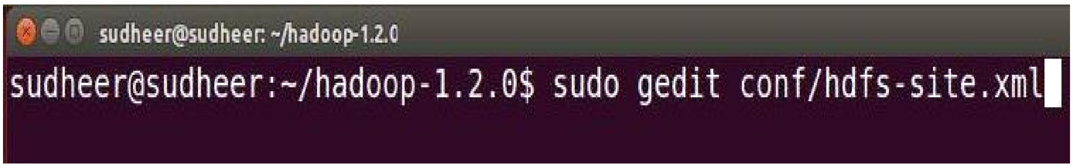
</property>



```
core-site.xml
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

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

Configure hdfs-site.xml

Command: sudo gedit conf/hdfs-site.xml



```
sudheer@sudheer: ~/hadoop-1.2.0
sudheer@sudheer:~/hadoop-1.2.0$ sudo gedit conf/hdfs-site.xml
```

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.permissions</name>

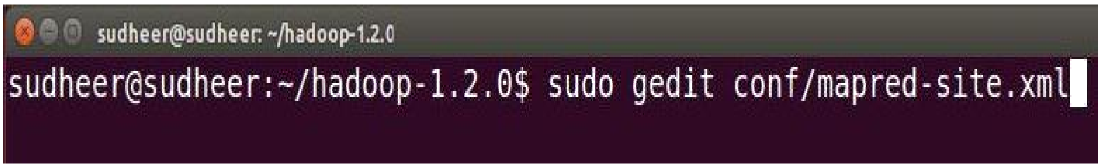
<value>false</value>

</property>

```
<property>
<name>dfs.name.dir</name>
        <value>${Hadoop.tmp.dir}/dfs/name</value> </property>
```

Configure mapred-site.xml

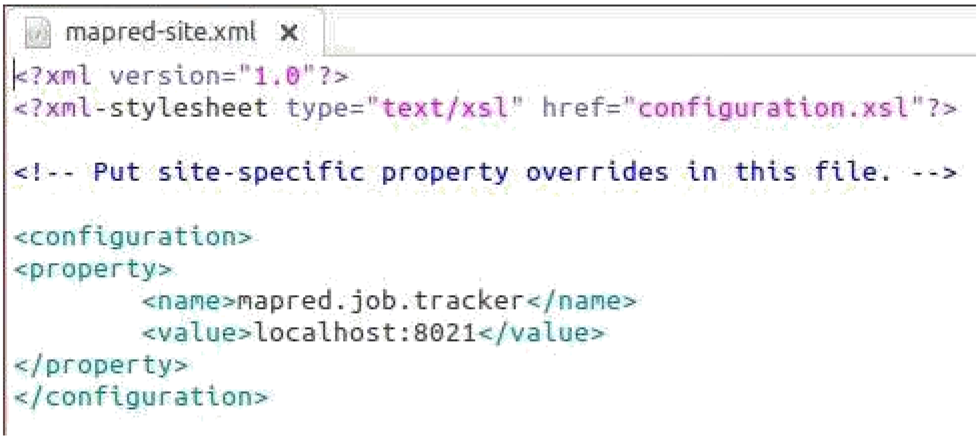
Command: sudo gedit conf/mapred-site.xml



sudheer@sudheer: ~/hadoop-1.2.0

```
sudheer@sudheer:~/hadoop-1.2.0$ sudo gedit conf/mapred-site.xml
```

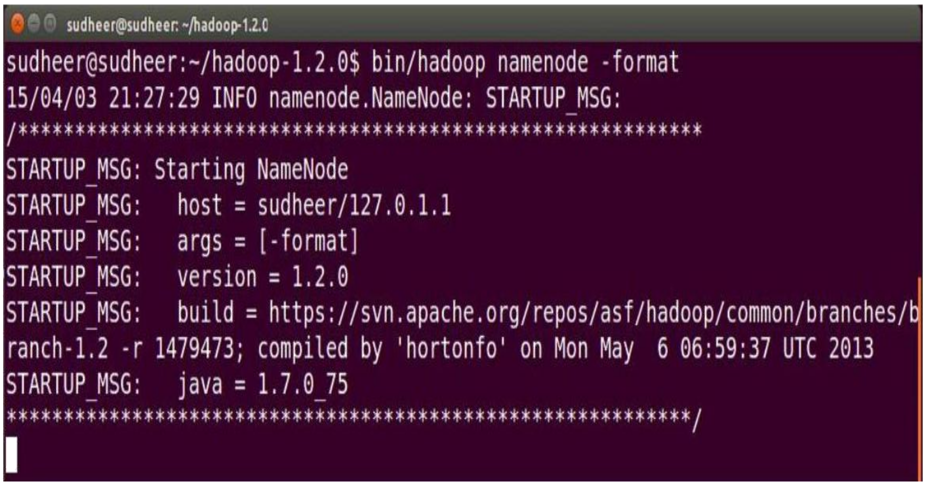
```
<property>
        <name>mapred.job.tracker</name>
        <value>localhost:8021</value>
</property>
```



```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

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

Format the name node

Command: bin/hadoop namenode -format



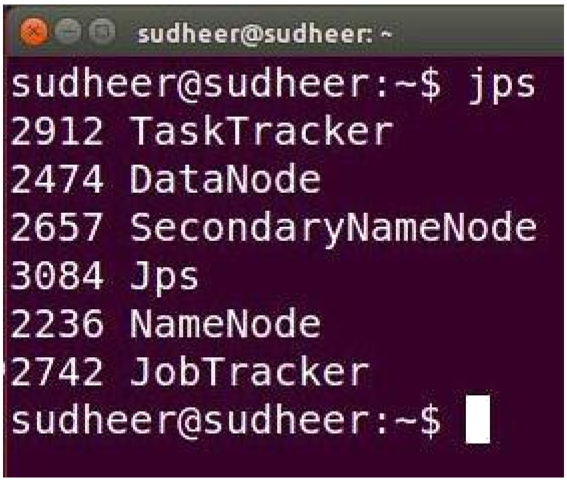
sudheer@sudheer: ~/hadoop-1.2.0

```
sudheer@sudheer:~/hadoop-1.2.0$ bin/hadoop namenode -format
15/04/03 21:27:29 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = sudheer/127.0.1.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 1.2.0
STARTUP_MSG:   build = https://svn.apache.org/repos/asf/hadoop/common/branches/b
ranch-1.2 -r 1479473; compiled by 'hortonfo' on Mon May  6 06:59:37 UTC 2013
STARTUP_MSG:   java = 1.7.0_75
************************************************************/
```

Start the namenode, datanode **Command:** bin/start-dfs.sh

Start the task tracker and job tracker **Command:** bin/start-mapred.sh

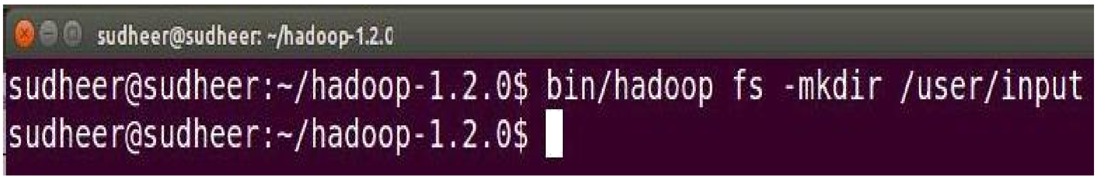
To check if Hadoop started correctly **Command:** jps



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

Create input directory on hdfs

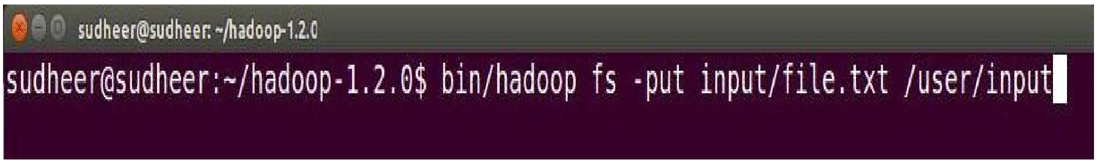
Command: bin/hadoop fs -mkdir /user/input



```
sudheer@sudheer:~/hadoop-1.2.0$ bin/hadoop fs -mkdir /user/input
sudheer@sudheer:~/hadoop-1.2.0$
```

Put the file from local file system to hdfs

Command: bin/hadoop fs -put input/file.txt /user/input



```
sudheer@sudheer:~/hadoop-1.2.0$ bin/hadoop fs -put input/file.txt /user/input
```

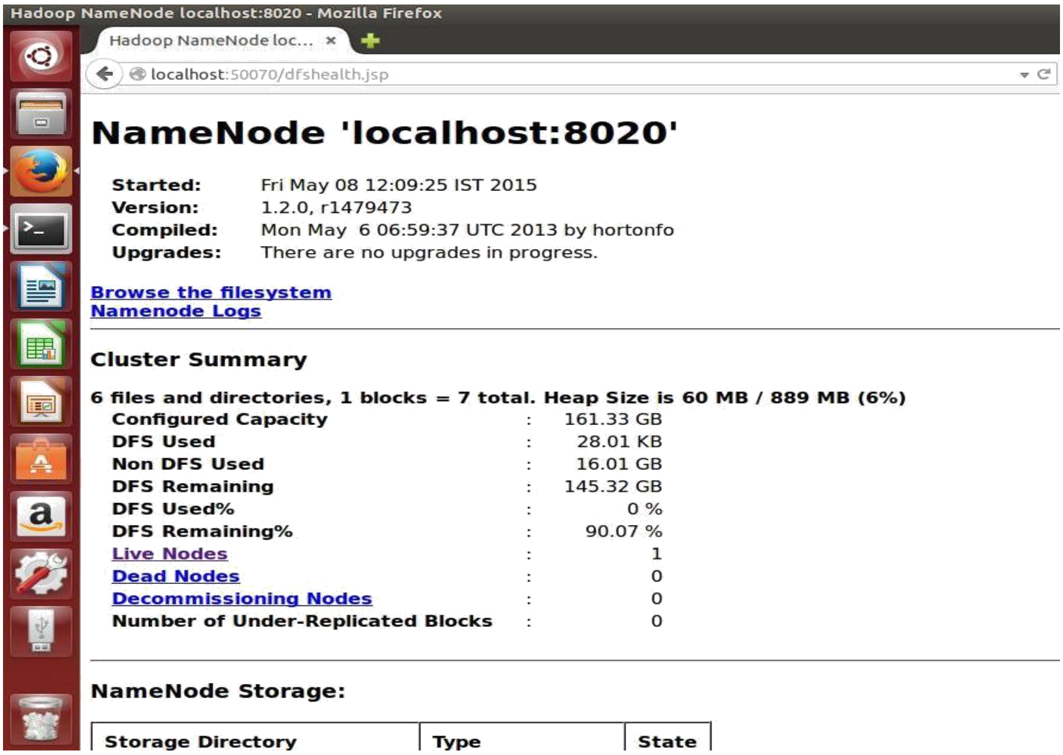
Apply the WordCount program on input directory

Command: bin/hadoop jar wc.jar WordCount /user/input /user/output

Experiment – 5

AIM: To monitor User Interface using HDFS.

HDFS Namenode on UI http://locahost:50070/



Hadoop NameNode localhost:8020 - Mozilla Firefox

localhost:50070/dfshealth.jsp

NameNode 'localhost:8020'

Started:	Fri May 08 12:09:25 IST 2015
Version:	1.2.0, r1479473
Compiled:	Mon May 6 06:59:37 UTC 2013 by hortonfo
Upgrades:	There are no upgrades in progress.

Browse the filesystem
Namenode Logs

Cluster Summary

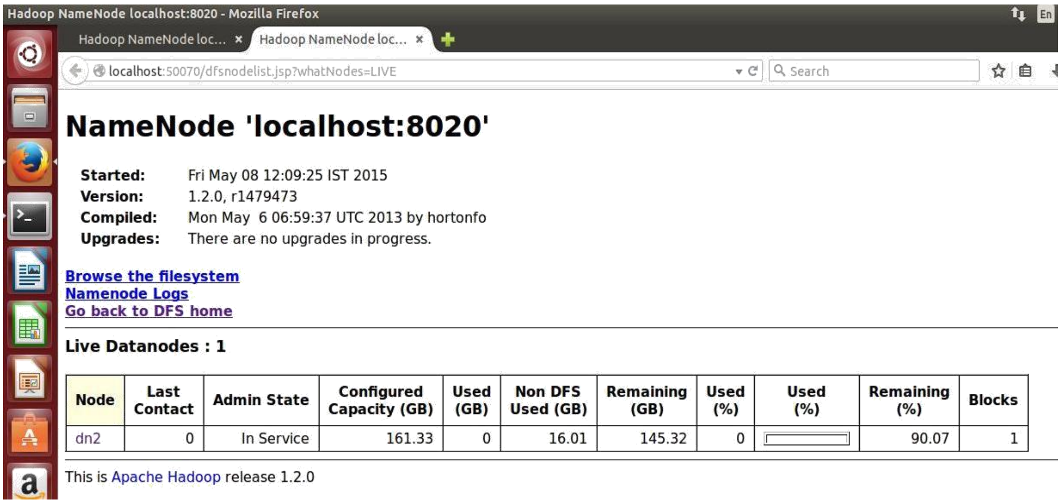
6 files and directories, 1 blocks = 7 total. Heap Size is 60 MB / 889 MB (6%)

Configured Capacity	:	161.33 GB
DFS Used	:	28.01 KB
Non DFS Used	:	16.01 GB
DFS Remaining	:	145.32 GB
DFS Used%	:	0 %
DFS Remaining%	:	90.07 %
Live Nodes	:	1
Dead Nodes	:	0
Decommissioning Nodes	:	0
Number of Under-Replicated Blocks	:	0

NameNode Storage:

Storage Directory	Type	State

HDFS Live Nodes list



Hadoop NameNode localhost:8020 - Mozilla Firefox

localhost:50070/dfsnodelist.jsp?whatNodes=LIVE

NameNode 'localhost:8020'

Started:	Fri May 08 12:09:25 IST 2015
Version:	1.2.0, r1479473
Compiled:	Mon May 6 06:59:37 UTC 2013 by hortonfo
Upgrades:	There are no upgrades in progress.

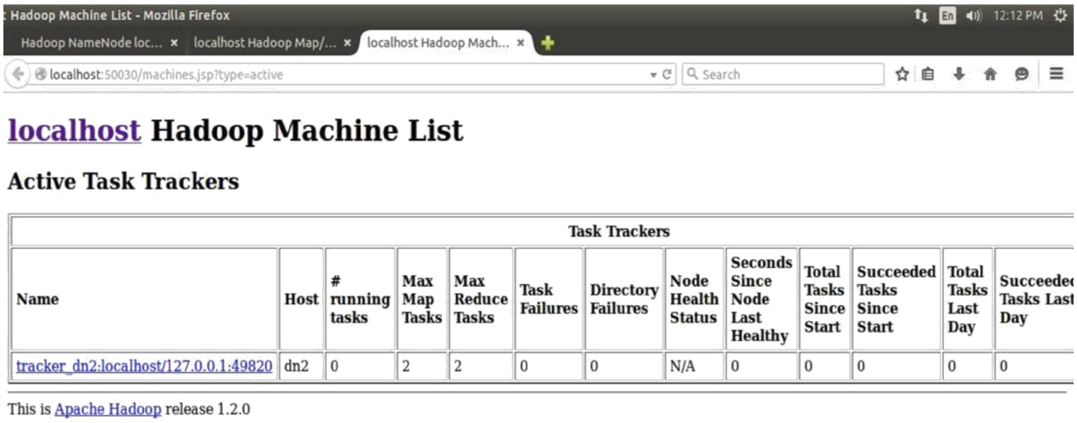
Browse the filesystem
Namenode Logs
Go back to DFS home

Live Datanodes : 1

Node	Last Contact	Admin State	Configured Capacity (GB)	Used (GB)	Non DFS Used (GB)	Remaining (GB)	Used (%)	Used (%)	Remaining (%)	Blocks
dn2	0	In Service	161.33	0	16.01	145.32	0		90.07	1

This is Apache Hadoop release 1.2.0

HDFS Machine list



Hadoop Machine List - Mozilla Firefox

localhost:50030/machines.jsp?type=active

localhost Hadoop Machine List

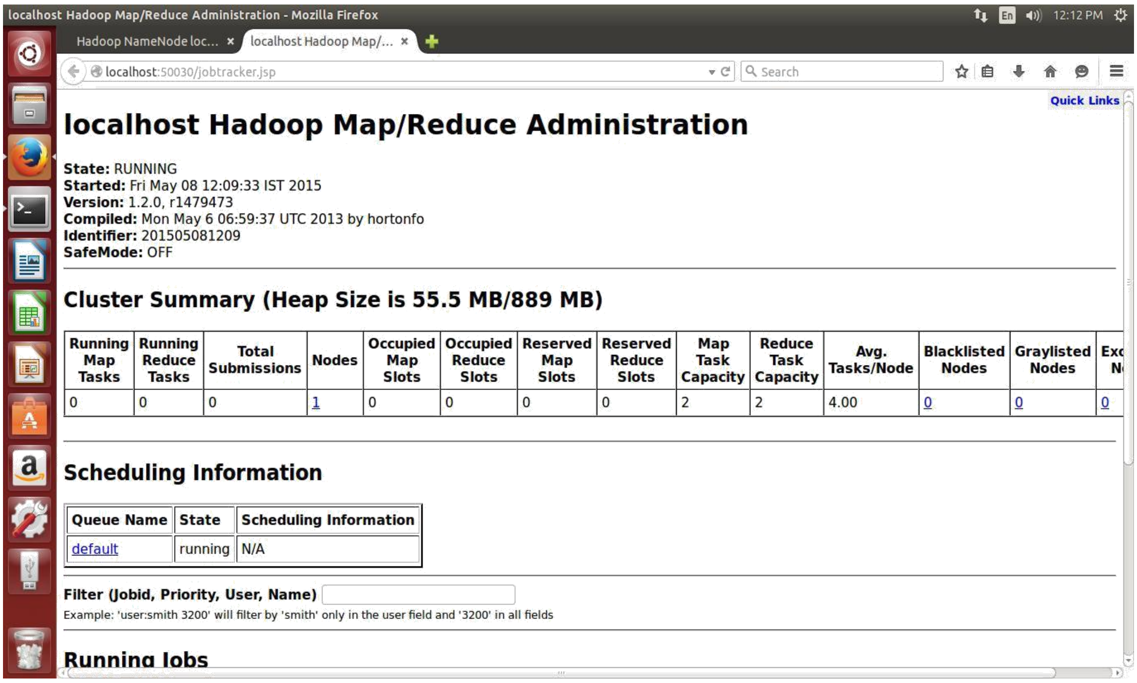
Active Task Trackers

Task Trackers

Name	Host	# running tasks	Max Map Tasks	Max Reduce Tasks	Task Failures	Directory Failures	Node Health Status	Seconds Since Node Last Healthy	Total Tasks Since Start	Succeeded Tasks Since Start	Total Tasks Last Day	Succeeded Tasks Last Day
tracker_dn2:localhost/127.0.0.1:49820	dn2	0	2	2	0	0	N/A	0	0	0	0	0

This is Apache Hadoop release 1.2.0

HDFS Jobtracker http://locahost:50030


localhost Hadoop Map/Reduce Administration - Mozilla Firefox
12:12 PM
Hadoop NameNode loc...
localhost Hadoop Map/...
localhost:50030/jobtracker.jsp
Search
Quick Links

localhost Hadoop Map/Reduce Administration

State: RUNNING
Started: Fri May 08 12:09:33 IST 2015
Version: 1.2.0, r1479473
Compiled: Mon May 6 06:59:37 UTC 2013 by hortonfo
Identifier: 201505081209
SafeMode: OFF

Cluster Summary (Heap Size is 55.5 MB/889 MB)

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes	Graylisted Nodes	Exc N
0	0	0	1	0	0	0	0	2	2	4.00	0	0	0

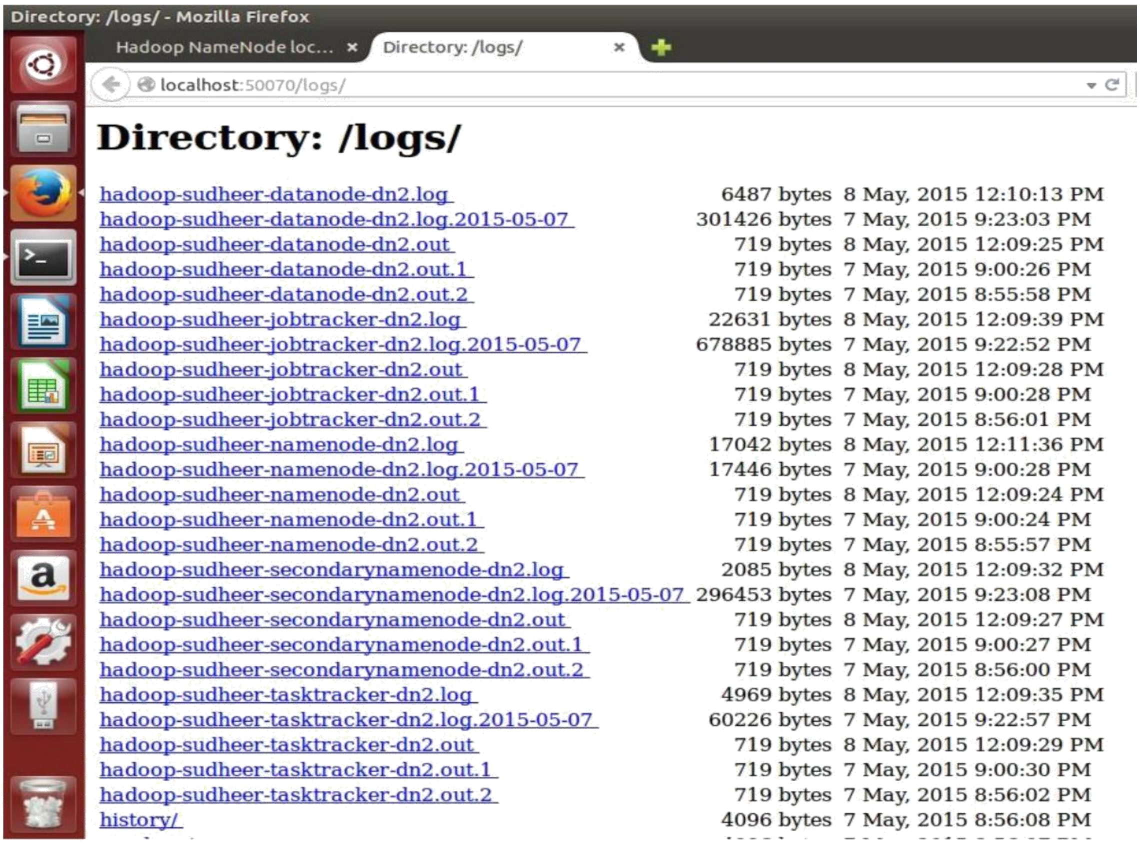
Scheduling Information

Queue Name	State	Scheduling Information
default	running	N/A

Filter (Jobid, Priority, User, Name)
Example: 'user:smith 3200' will filter by 'smith' only in the user field and '3200' in all fields

Running Jobs

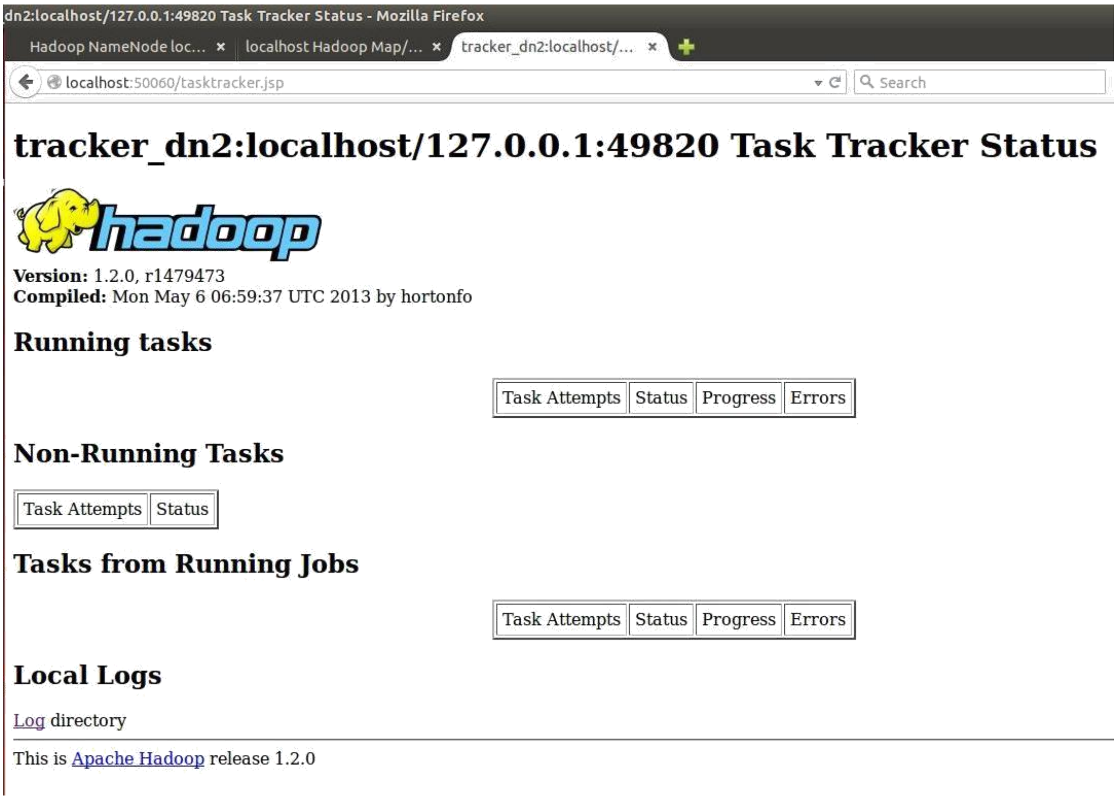
HDFS Logs http://locahost:50070/logs/



Directory: /logs/

hadoop-sudheer-datanode-dn2.log	6487 bytes	8 May, 2015 12:10:13 PM
hadoop-sudheer-datanode-dn2.log.2015-05-07	301426 bytes	7 May, 2015 9:23:03 PM
hadoop-sudheer-datanode-dn2.out	719 bytes	8 May, 2015 12:09:25 PM
hadoop-sudheer-datanode-dn2.out.1	719 bytes	7 May, 2015 9:00:26 PM
hadoop-sudheer-datanode-dn2.out.2	719 bytes	7 May, 2015 8:55:58 PM
hadoop-sudheer-jobtracker-dn2.log	22631 bytes	8 May, 2015 12:09:39 PM
hadoop-sudheer-jobtracker-dn2.log.2015-05-07	678885 bytes	7 May, 2015 9:22:52 PM
hadoop-sudheer-jobtracker-dn2.out	719 bytes	8 May, 2015 12:09:28 PM
hadoop-sudheer-jobtracker-dn2.out.1	719 bytes	7 May, 2015 9:00:28 PM
hadoop-sudheer-jobtracker-dn2.out.2	719 bytes	7 May, 2015 8:56:01 PM
hadoop-sudheer-namenode-dn2.log	17042 bytes	8 May, 2015 12:11:36 PM
hadoop-sudheer-namenode-dn2.log.2015-05-07	17446 bytes	7 May, 2015 9:00:28 PM
hadoop-sudheer-namenode-dn2.out	719 bytes	8 May, 2015 12:09:24 PM
hadoop-sudheer-namenode-dn2.out.1	719 bytes	7 May, 2015 9:00:24 PM
hadoop-sudheer-namenode-dn2.out.2	719 bytes	7 May, 2015 8:55:57 PM
hadoop-sudheer-secondarynamenode-dn2.log	2085 bytes	8 May, 2015 12:09:32 PM
hadoop-sudheer-secondarynamenode-dn2.log.2015-05-07	296453 bytes	7 May, 2015 9:23:08 PM
hadoop-sudheer-secondarynamenode-dn2.out	719 bytes	8 May, 2015 12:09:27 PM
hadoop-sudheer-secondarynamenode-dn2.out.1	719 bytes	7 May, 2015 9:00:27 PM
hadoop-sudheer-secondarynamenode-dn2.out.2	719 bytes	7 May, 2015 8:56:00 PM
hadoop-sudheer-tasktracker-dn2.log	4969 bytes	8 May, 2015 12:09:35 PM
hadoop-sudheer-tasktracker-dn2.log.2015-05-07	60226 bytes	7 May, 2015 9:22:57 PM
hadoop-sudheer-tasktracker-dn2.out	719 bytes	8 May, 2015 12:09:29 PM
hadoop-sudheer-tasktracker-dn2.out.1	719 bytes	7 May, 2015 9:00:30 PM
hadoop-sudheer-tasktracker-dn2.out.2	719 bytes	7 May, 2015 8:56:02 PM
history/	4096 bytes	7 May, 2015 8:56:08 PM

HDFS Tasktracker http://locahost:50060/



dn2:localhost/127.0.0.1:49820 Task Tracker Status - Mozilla Firefox
Hadoop NameNode loc…
localhost Hadoop Map/…
tracker_dn2:localhost/…
localhost:50060/tasktracker.jsp
Search

tracker_dn2:localhost/127.0.0.1:49820 Task Tracker Status

Version: 1.2.0, r1479473
Compiled: Mon May 6 06:59:37 UTC 2013 by hortonfo

Running tasks

Task Attempts	Status	Progress	Errors

Non-Running Tasks

Task Attempts	Status

Tasks from Running Jobs

Task Attempts	Status	Progress	Errors

Local Logs

Log directory

This is Apache Hadoop release 1.2.0

Experiment – 6

AIM: To perform HDFS basic Command-line file operations

Create a directory in HDFS at given path(s):

Command: hadoop fs -mkdir <paths>

List the contents of a directory:

Command: hadoop fs -ls <args>

Upload and download a file in HDFS: *Upload:*

Command: hadoop fs -put <localsrc> <HDFS_dest_path>

Download:

Command: hadoop fs -get <HDFS_src> <localdst>

See contents of a file:

Command: hadoop fs -cat <path[filename]>

Copy a file from source to destination:

Command: hadoop fs -cp <source> <dest>

Copy a file from/To Local file system to HDFS:

Command: hadoop fs -copyFromLocal <localsrc>

URI

Command: hadoop fs –copyToLocal [-ignorecrc] [-crc] URI <localsrc>

Move file from source to destination:

Command: hadoop fs -mv <src> <dest>

Remove a file or directory in HDFS:

Remove files specified as argument. Delete directory only when it is empty.

Command: hadoop fs -rm <arg>

Recursive version of delete

Command: hadoop fs -rmr <arg>

Display last few lines of a file:

Command: hadoop fs -tail <path[filename]>

Display the aggregate length of a file:

Command: hadoop fs -du <path>

Getting help:

Command: hadoop fs -help

Experiment – 7

AIM: To set up Hadoop in a distributed cluster environment.

Configure /etc/hosts

Command: sudo gedit /etc/hosts

```
127.0.0.1     localhost
#127.0.1.1    dn2

# The following lines are desirable for IPv6 capable hosts::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
192.168.1.5          nn

192.168.1.6          dn1
192.168.1.7          dn2
192.168.1.8          dn3
```

Install ssh server on all nodes

Command: sudo apt-get install openssh-server

Create a ssh key (on Namenode)

Command: ssh-keygen -t rsa -P ""

Create a password-less ssh login

Command: ssh-copy-id -i $HOME/.ssh/id_rsa.pub huser@192.168.1.5

Command: ssh-copy-id -i $HOME/.ssh/id_rsa.pub huser@192.168.1.6

Command: ssh-copy-id -i $HOME/.ssh/id_rsa.pub huser@192.168.1.7

Command: ssh-copy-id -i $HOME/.ssh/id_rsa.pub huser@192.168.1.8

Test ssh login

Command: ssh 192.168.1.5

Command: ssh 192.168.1.6

Command: ssh 192.168.1.7

Command: ssh 192.168.1.8

Extract Hadoop-1.2.0

Command: tar -xvf hadoop-1.2.0.tar.gz

Command: cd hadoop-1.2.0

Edit Hadoop-env.sh

Command: sudo gedit conf/hadoop-env.sh

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

Configure core-site.xml

Command: bin/hadoop conf/core-site.xml <configuration>

```
<property>
<name>fs.default.name</name>
<value>hdfs://192.168.1.5:8020</value>
</property>
</configuration>
```

Configure hdfs-site.xml

Command: bin/hadoop conf/hdfs-site.xml

```
<configuration>
<property>
<name>dfs.replication</name>
<value>3</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>
</configuration>
```

Configure mapred-site.xml

Command: bin/hadoop conf/mapred-site.xml <configuration>

```
<property>
<name>mapred.job.tracker</name>
<value>192.168.1.5:8021</value>
</property>
</configuration>
```

Configure masters

Command: bin/hadoop conf/masters 192.168.1.5

Configure slaves

Command: bin/hadoop conf/slaves 192.168.1.6

192.168.1.7

192.168.1.8

Format Namenode

Command: bin/hadoop namenode -format

Start Namenode and Datanode

Command: bin/start-dfs.sh

Start Jobtracker and Tasktracker

Command: bin/start-mapred.sh

To check if Hadoop started correctly

Command: jps

Experiment – 8

AIM: To run the Word Count program in a distributed cluster environment

Create input directory on hdfs

Command: bin/hadoop fs -mkdir input

Put the file from local file system to hdfs

Command: bin/hadoop fs -put pdf/* input

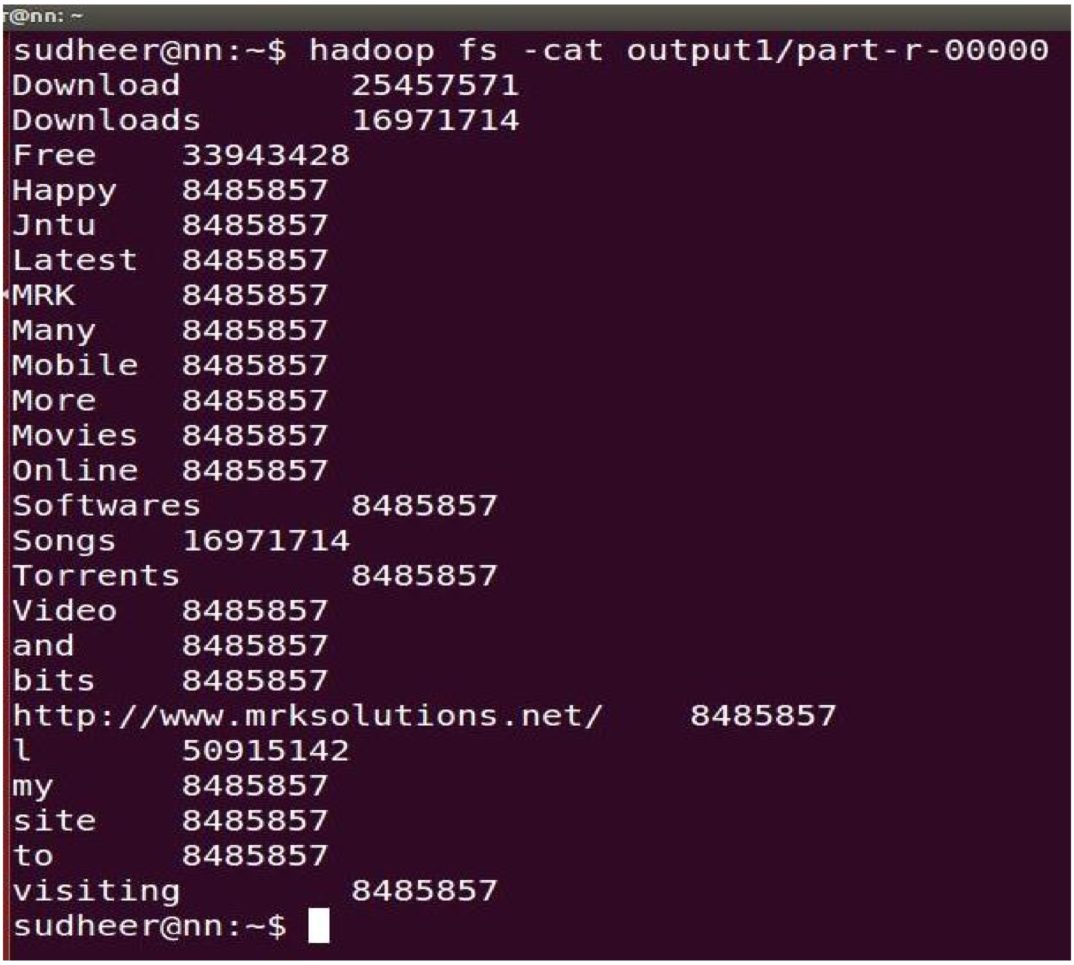
Apply the WordCount program on input directory

Command: bin/hadoop jar wc.jar WordCount input output1

To see the output

Command: bin/hadoop fs -ls /user/output1/

Command: bin/hadoop fs -cat /user/outpu1t/part-r-00000

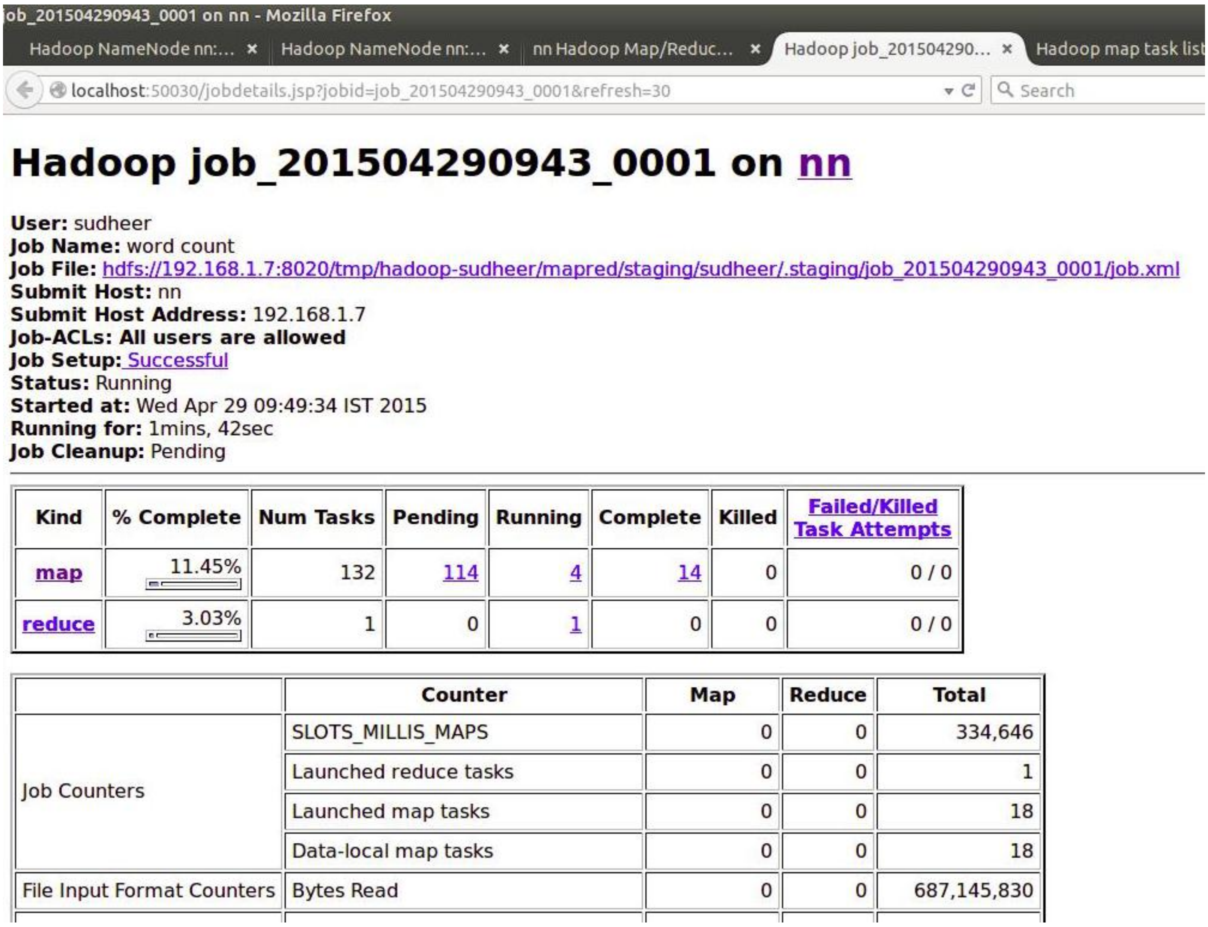


```
sudheer@nn:~$ hadoop fs -cat output1/part-r-00000
Download        25457571
Downloads       16971714
Free    33943428
Happy   8485857
Jntu    8485857
Latest  8485857
MRK     8485857
Many    8485857
Mobile  8485857
More    8485857
Movies  8485857
Online  8485857
Softwares       8485857
Songs   16971714
Torrents        8485857
Video   8485857
and     8485857
bits    8485857
http://www.mrksolutions.net/    8485857
l       50915142
my      8485857
site    8485857
to      8485857
visiting        8485857
sudheer@nn:~$
```

Experiment – 9

AIM: To monitor UI using Map Reduce

Open the jobtracker on browser http://localhost:50030/ click on running job details



ob_201504290943_0001 on nn - Mozilla Firefox

localhost:50030/jobdetails.jsp?jobid=job_201504290943_0001&refresh=30

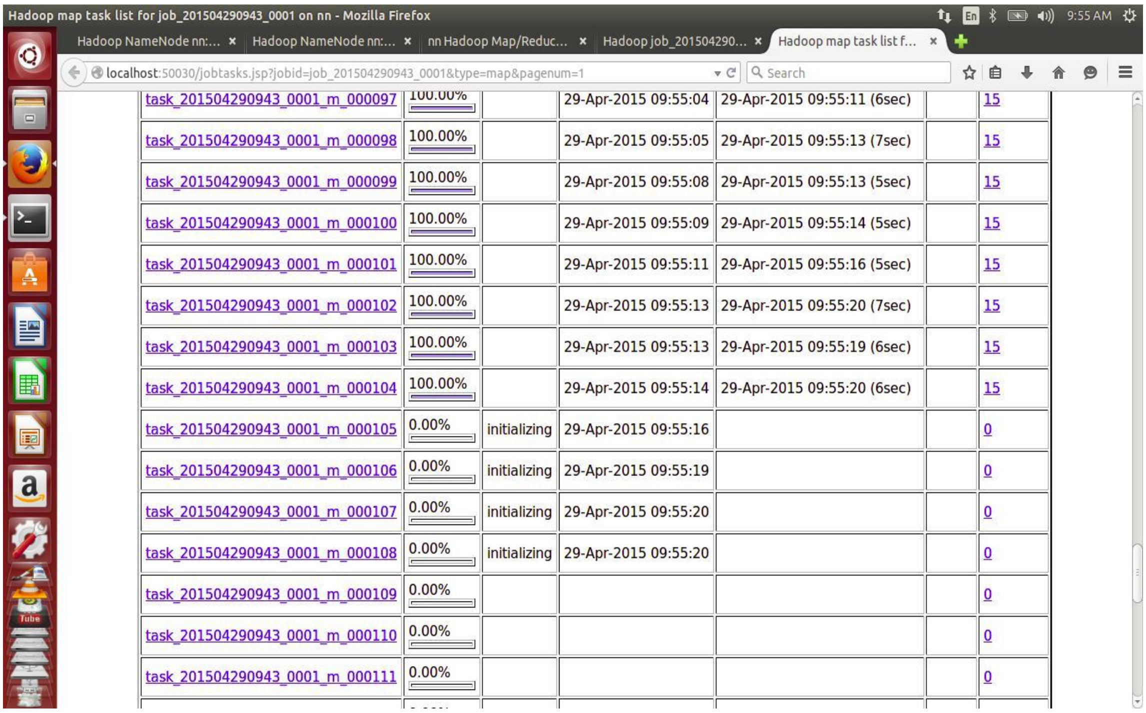
Hadoop job_201504290943_0001 on nn

User: sudheer
Job Name: word count
Job File: hdfs://192.168.1.7:8020/tmp/hadoop-sudheer/mapred/staging/sudheer/.staging/job_201504290943_0001/job.xml
Submit Host: nn
Submit Host Address: 192.168.1.7
Job-ACLs: All users are allowed
Job Setup: Successful
Status: Running
Started at: Wed Apr 29 09:49:34 IST 2015
Running for: 1mins, 42sec
Job Cleanup: Pending

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	11.45%	132	114	4	14	0	0 / 0
reduce	3.03%	1	0	1	0	0	0 / 0

	Counter	Map	Reduce	Total
Job Counters	SLOTS_MILLIS_MAPS	0	0	334,646
	Launched reduce tasks	0	0	1
	Launched map tasks	0	0	18
	Data-local map tasks	0	0	18
File Input Format Counters	Bytes Read	0	0	687,145,830

The map tasks

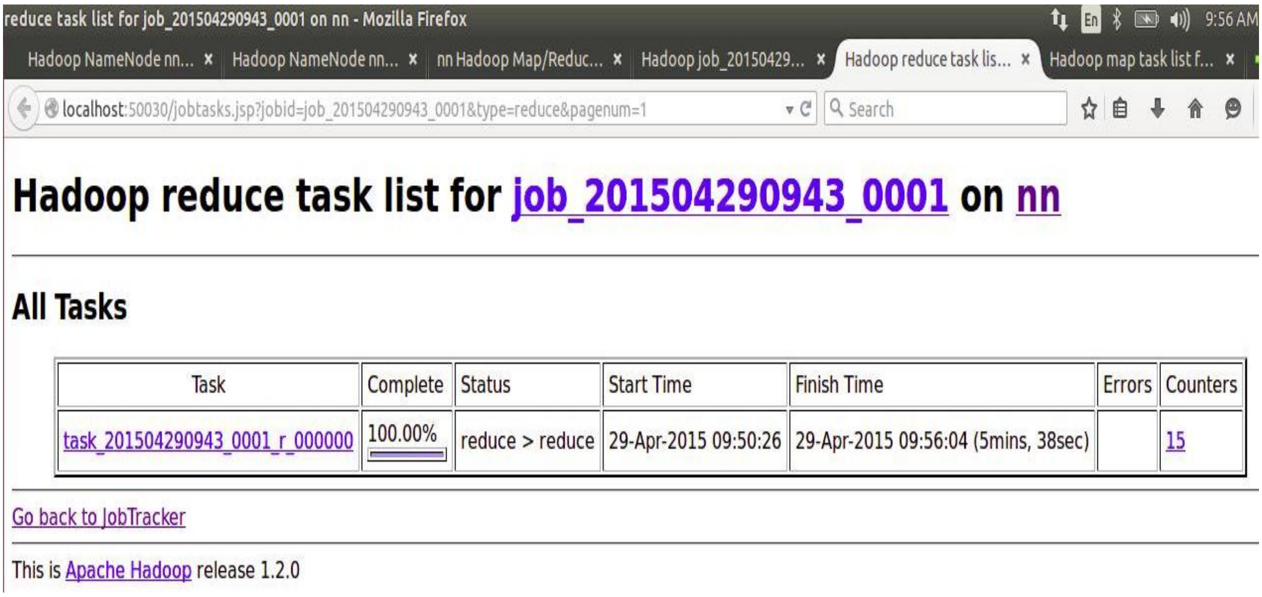


Hadoop map task list for job_201504290943_0001 on nn - Mozilla Firefox

localhost:50030/jobtasks.jsp?jobid=job_201504290943_0001&type=map&pagenum=1

task_201504290943_0001_m_000097	100.00%		29-Apr-2015 09:55:04	29-Apr-2015 09:55:11 (6sec)		15
task_201504290943_0001_m_000098	100.00%		29-Apr-2015 09:55:05	29-Apr-2015 09:55:13 (7sec)		15
task_201504290943_0001_m_000099	100.00%		29-Apr-2015 09:55:08	29-Apr-2015 09:55:13 (5sec)		15
task_201504290943_0001_m_000100	100.00%		29-Apr-2015 09:55:09	29-Apr-2015 09:55:14 (5sec)		15
task_201504290943_0001_m_000101	100.00%		29-Apr-2015 09:55:11	29-Apr-2015 09:55:16 (5sec)		15
task_201504290943_0001_m_000102	100.00%		29-Apr-2015 09:55:13	29-Apr-2015 09:55:20 (7sec)		15
task_201504290943_0001_m_000103	100.00%		29-Apr-2015 09:55:13	29-Apr-2015 09:55:19 (6sec)		15
task_201504290943_0001_m_000104	100.00%		29-Apr-2015 09:55:14	29-Apr-2015 09:55:20 (6sec)		15
task_201504290943_0001_m_000105	0.00%	initializing	29-Apr-2015 09:55:16			0
task_201504290943_0001_m_000106	0.00%	initializing	29-Apr-2015 09:55:19			0
task_201504290943_0001_m_000107	0.00%	initializing	29-Apr-2015 09:55:20			0
task_201504290943_0001_m_000108	0.00%	initializing	29-Apr-2015 09:55:20			0
task_201504290943_0001_m_000109	0.00%					0
task_201504290943_0001_m_000110	0.00%					0
task_201504290943_0001_m_000111	0.00%					0

The Reduce Tasks


reduce task list for job_201504290943_0001 on nn - Mozilla Firefox
En
9:56 AM
Hadoop NameNode nn... ×
Hadoop NameNode nn... ×
nn Hadoop Map/Reduc... ×
Hadoop job_20150429... ×
Hadoop reduce task lis... ×
Hadoop map task list f... ×
localhost:50030/jobtasks.jsp?jobid=job_201504290943_0001&type=reduce&pagenum=1
Search

Hadoop reduce task list for job_201504290943_0001 on nn

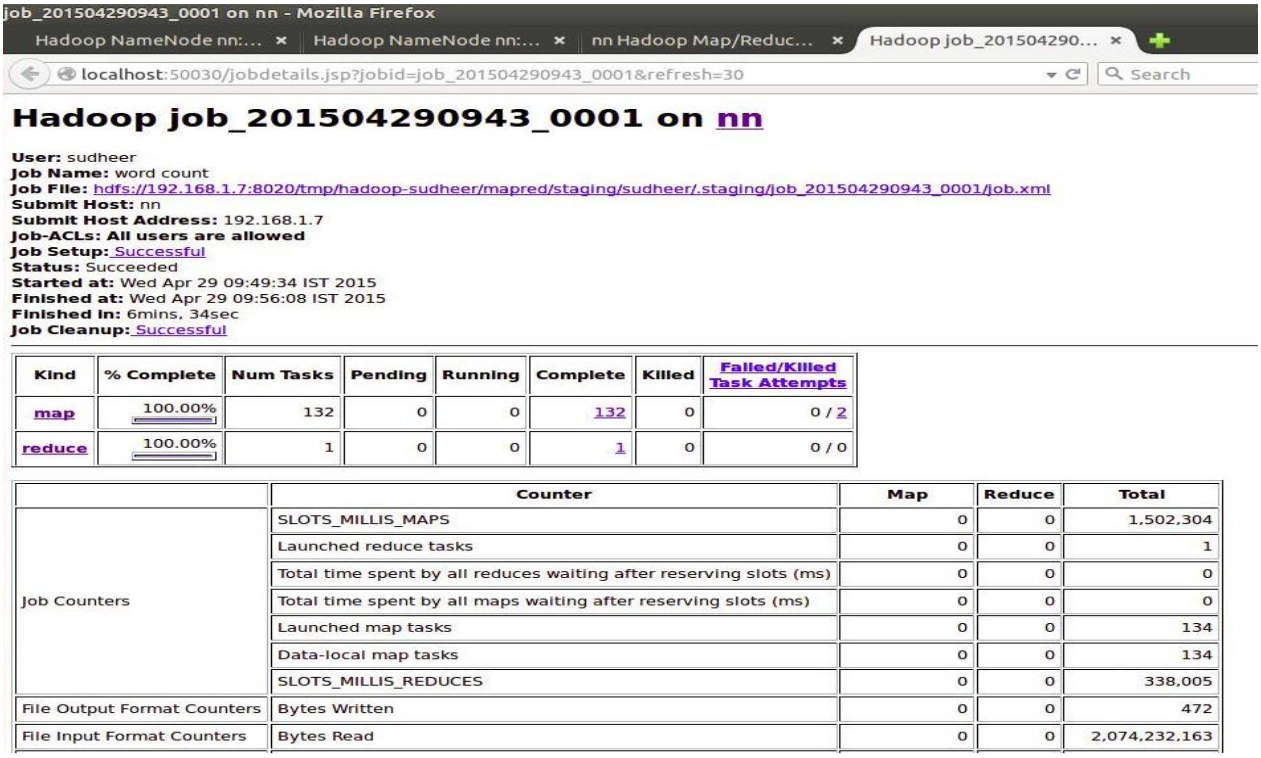
All Tasks

Task	Complete	Status	Start Time	Finish Time	Errors	Counters
task_201504290943_0001_r_000000	100.00%	reduce > reduce	29-Apr-2015 09:50:26	29-Apr-2015 09:56:04 (5mins, 38sec)		15

Go back to JobTracker

This is Apache Hadoop release 1.2.0

The job is Completed


job_201504290943_0001 on nn - Mozilla Firefox
Hadoop NameNode nn:... ×
Hadoop NameNode nn:... ×
nn Hadoop Map/Reduc... ×
Hadoop job_201504290... ×
localhost:50030/jobdetails.jsp?jobid=job_201504290943_0001&refresh=30
Search

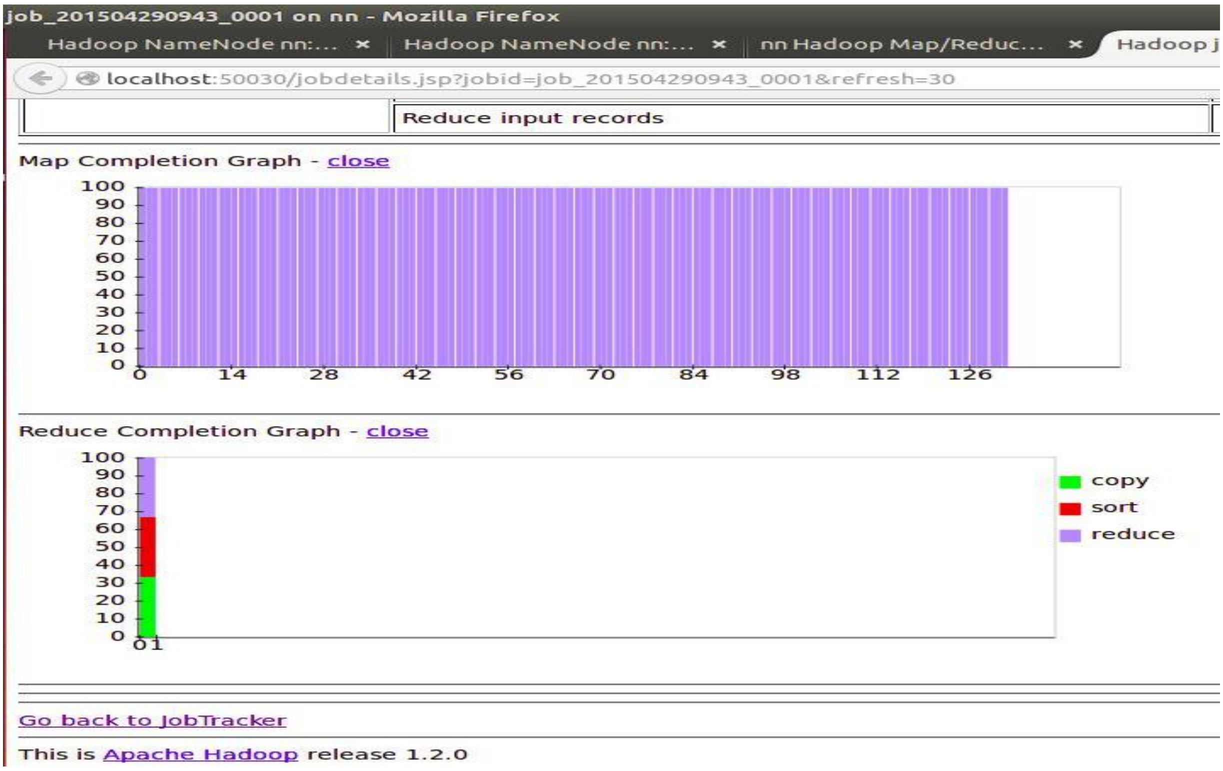
Hadoop job_201504290943_0001 on nn

User: sudheer
Job Name: word count
Job File: hdfs://192.168.1.7:8020/tmp/hadoop-sudheer/mapred/staging/sudheer/.staging/job_201504290943_0001/job.xml
Submit Host: nn
Submit Host Address: 192.168.1.7
Job-ACLs: All users are allowed
Job Setup: Successful
Status: Succeeded
Started at: Wed Apr 29 09:49:34 IST 2015
Finished at: Wed Apr 29 09:56:08 IST 2015
Finished in: 6mins, 34sec
Job Cleanup: Successful

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	132	0	0	132	0	0 / 2
reduce	100.00%	1	0	0	1	0	0 / 0

	Counter	Map	Reduce	Total
Job Counters	SLOTS_MILLIS_MAPS	0	0	1,502,304
	Launched reduce tasks	0	0	1
	Total time spent by all reduces waiting after reserving slots (ms)	0	0	0
	Total time spent by all maps waiting after reserving slots (ms)	0	0	0
	Launched map tasks	0	0	134
	Data-local map tasks	0	0	134
	SLOTS_MILLIS_REDUCES	0	0	338,005
File Output Format Counters	Bytes Written	0	0	472
File Input Format Counters	Bytes Read	0	0	2,074,232,163

The Map and Reduce Completion graphs



Map Completion Graph - close

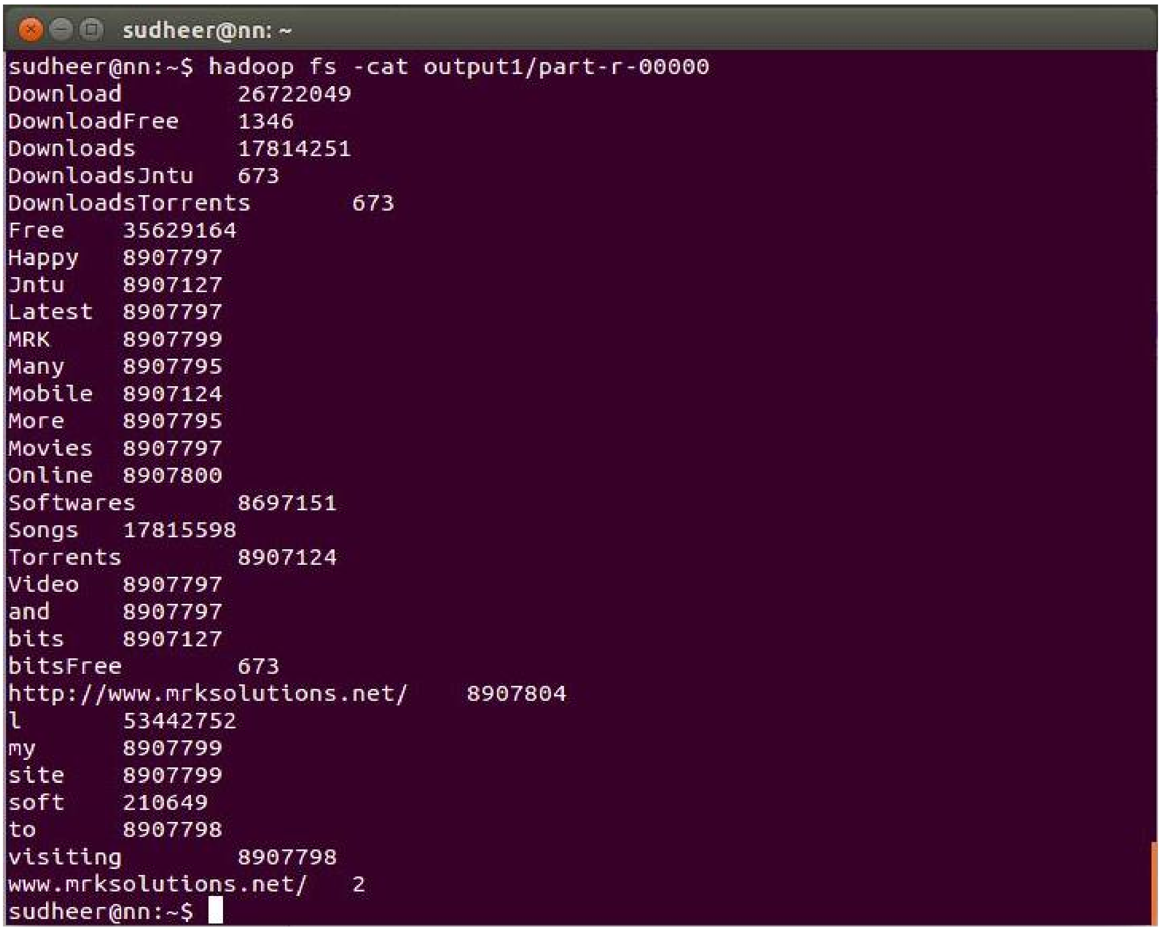
Reduce Completion Graph - close

🟩 copy
🟥 sort
🟪 reduce

Go back to JobTracker

This is Apache Hadoop release 1.2.0

The output of Word count Map reduce



```
sudheer@nn:~$ hadoop fs -cat output1/part-r-00000
Download        26722049
DownloadFree    1346
Downloads       17814251
DownloadsJntu   673
DownloadsTorrents       673
Free    35629164
Happy   8907797
Jntu    8907127
Latest  8907797
MRK     8907799
Many    8907795
Mobile  8907124
More    8907795
Movies  8907797
Online  8907800
Softwares       8697151
Songs   17815598
Torrents        8907124
Video   8907797
and     8907797
bits    8907127
bitsFree        673
http://www.mrksolutions.net/    8907804
l       53442752
my      8907799
site    8907799
soft    210649
to      8907798
visiting        8907798
www.mrksolutions.net/   2
sudheer@nn:~$
```

Hadoop Map Reduce Applications

Experiment –10

AIM: To Choose appropriate Hadoop data types.

Hadoop uses the Writable interface based classes as the data types for the Map Reduce computations. These data types are used throughout the map reduce computational flow, starting with reading the input data, transferring intermediate data between Map and Reduce tasks, and finally, when writing output data.

In order to be used as a value data type of a Map Reduce computation, a data type must implement the org.apache.hadoop.io.Writable interface. The Writable interface defines how hadoop should serialize or deserialize the values transmitting and storing the data.

Some of primitive data types provided by hadoop:

Hadoop	**java**
IntWritable	int
LongWritable	long
BooleanWritable	boolean
FloatWritable	float
ByteWritable	byte
Text	String

Configure the input and output data types of your Hadoop Map Reduce application:

Specify the data types for the input (key: LongWritable, value: Text) and output (key: Text, value: IntWritable) key-value pairs of your mapper using the generic-type variables.

```
public class SampleMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
public void map(LongWritable key, Text value, Context context) … {
}
```

Specify the data types for the input (key: Text, value: IntWritable) and output (key: Text, value: IntWritable) key-value pairs of your reducer using the generic-type variables. The reducer's input key-value pair data types should match the mapper's output key-value pairs.

```
public class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {
public void reduce(Text key, Iterable<IntWritable> values, Context context) {
}}
```

Specify the output data types of the Map Reduce computation using the Job object as shown in the following code snippet.

```
Job job = new Job(..);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
```

Experiment –11

AIM: To Implement a custom Hadoop Writable data type.

There can be use cases where none of the built-in data types matches your requirements or a custom data type optimized for your use case may perform better than a Hadoop built-in data type. In such scenarios, we can easily write a custom Writable data type by implementing the org.apache.hadoop.io.Writable Interface to define the serialization format of your data type.

Program to implement custom datatypes:

```
public static class CustomWritable implements WritableComparable<CustomWritable> { private
Text siteURL;
private IntWritable reqNo;
//Default Constructor
public CustomWritable() {
this.siteURL = new Text();
this.reqNo = new IntWritable();
}
//Custom Constructor
public CustomWritable(IntWritable reqno, Text url) { this.siteURL = url;
this.reqNo = reqno;
}
//Setter method to set the values of CustomWritable object public void set(IntWritable reqno, Text
url) {
this.siteURL = url;
this.reqNo = reqno;
}
//to get IP address from WebLog Record
public Text getWord() {
return siteURL;
}
@Override
//overriding default readFields method.
//It de-serializes the byte stream data
public void readFields(DataInput in) throws IOException { reqNo.readFields(in);
siteURL.readFields(in);
}
@Override
//It serializes object data into byte stream data
public void write(DataOutput out) throws IOException { reqNo.write(out);
siteURL.write(out);
}
@Override
public int compareTo(CustomWritable o) {
if (siteURL.compareTo(o.siteURL)==0)
{
return (reqNo.compareTo(o.reqNo));
}
else return (siteURL.compareTo(o.siteURL));
}
@Override
public boolean equals(Object o) {
if (o instanceof CustomWritable)
{
CustomWritable other = (CustomWritable) o;
return siteURL.equals(other.siteURL) && reqNo.equals(other.reqNo);
}
return false;
}
@Override
public int hashCode() {
return siteURL.hashCode();
}
}
```

Experiment –12

AIM: To Implement a custom Hadoop key type.

The instances of Hadoop MapReduce key types should have the ability to compare against each other for sorting purposes. In order to be used as a key type in a MapReduce a computation, a Hadoop Writable data type should implement the org.apache.hadoop.io.WritableComparable<T> interface. The WritableComparable interface extends the org.apache.hadoop.io.Writable interface and adds the compareTo() method to perform the comparisons.

The following are the steps to implement custom hadoop writable data types for WordCount

Program:

Step 1:

```
public static class CustomMapper extends Mapper <Object, Text, CustomWritable,
IntWritable> {
}
```

Step 2:

```
public static class CustomReducer extends Reducer < CustomWritable, IntWritable,
Text, IntWritable> {
}
```

Step 3:

```
Job job = new Job();
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
job.setMapOutputKeyClass(CustomWritable.class);
job.setMapOutputValueClass(IntWritable.class);
```

Experiment –13

AIM: To emit data of different value types from a mapper.

Emitting data products belonging to multiple value types from a mapper is useful when performing reducer-side joins as well as when we need to avoid the complexity of having multiple MapReduce computations to summarize different types of properties in a data set. However, Hadoop reducers do not allow multiple input value types. In these scenarios, we can use the Generic Writable class to wrap multiple value instances belonging to different data types.

The following are the steps to emitting data of different value types from a Mapper:

Step 1:

```
public class MultiValueWritable extends GenericWritable { private static Class[] CLASSES = new
Class[]{
IntWritable.class,
Text.class
};
public MultiValueWritable(){
}
public MultiValueWritable(Writable value){ set(value);
}
protected Class[] getTypes() {
return CLASSES;
}
}
```

Step 2:

```
public class LogProcessorMap extends
Mapper<Object, Text, Text, MultiValueWritable> {
private Text userHostText = new Text();
private Text requestText = new Text();
private IntWritableresponseSize = new IntWritable(); public void map(Object key, Text value,
Context context)…{
……// parse the value (log entry) using a regex.
userHostText.set(userHost);
requestText.set(request);
bytesWritable.set(responseSize);
context.write(userHostText,
newMultiValueWritable(requestText));
context.write(userHostText,
newMultiValueWritable(responseSize));
}
}
```

Step 3:

```
public class LogProcessorReduce extends Reducer<Text,MultiValueWritable,Text,Text> {
private Text result = new Text();
public void reduce(Text key,Iterable<MultiValueWritable> values, Context context)…{
int sum = 0;
StringBuilder requests = new StringBuilder();
for (MultiValueWritable multiValueWritable : values) { Writable writable =
multiValueWritable.get(); if (writable instanceof IntWritable){
sum += ((IntWritable)writable).get();
}else{
requests.append(((Text)writable).toString());
requests.append("\t");
}
}
result.set(sum + "\t"+requests);
context.write(key, result);
}
}
```

Step 4:

```
Configuration conf = new Configuration();
Job job = new Job(conf, "log-analysis");
…
job.setMapOutputValueClass(MultiValueWritable.class);
```

Experiment –14

AIM: To Choose a suitable Hadoop Input Format for your input data format

Hadoop supports processing of many different formats and types of data through InputFormat. The InputFormat of a Hadoop MapReduce computation generates the key-value pair inputs for the mappers by parsing the input data.

InputFormat also performs the splitting of the input data into logical partitions, essentially determining the number of Map tasks of a MapReduce computation and indirectly deciding the execution location of the Map tasks.

Hadoop generates a map task for each logical data partition and invokes the respective mappers with the key-value pairs of the logical splits as the input.

The following steps show you how to use FileInputFormat based KeyValueTextInputFormat as InputFormat for a Hadoop MapReduce computation.

1. In this example, we are going to specify the KeyValueTextInputFormat as InputFormat for a Hadoop MapReduce computation using the Job object as follows:

```
Configuration conf = new Configuration();
Job job = new Job(conf, "log-analysis");
……
job.SetInputFormat(KeyValueTextInputFormat.class)
```

2.Set the input paths to the job.

```
FileInputFormat.setInputPaths(job, new Path(inputPath));
```

Experiment –15

AIM: To Format the results of Map Reduce Computation using Hadoop Output Formats.

Hadoop uses the org.apache.hadoop.mapreduce.lib.output.TextOutputFormat<K,V> as the default OutputFormat for the MapReduce computations. TextOutputFormat writes the records of the output data to plain text files in HDFS using a separate line for each record.

TextOutputFormat uses the tab character to delimit between the key and the value of a record. TextOutputFormat extends FileOutputFormat, which is the base class for all file-based output formats.

The following steps show you how to use the FileOutputFormat based SequenceFileOutputFormat as the OutputFormat for a Hadoop MapReduce computation.

1. In this example, we are going to specify the org.apache.hadoop.mapreduce.lib.output.SequenceFileOutputFormat<K,V> as the OutputFormat for a Hadoop MapReduce computation using the Job object as follows:

```
Configuration conf = new Configuration();
Job job = new Job(conf, "log-analysis");
……
job.setOutputFormat(SequenceFileOutputFormat.class)
```

Set the output paths to the job.

```
FileOutputFormat.setOutputPath(job, new Path(outputPath));
```

Experiment –16

AIM: To perform Simple analytics using Map Reduce.

PROGRAM:

```
import java.io.IOException;
import java.util.Iterator;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
public class WebLogMessageSizeAggregator {
  public static final Pattern httplogPattern = Pattern
      .compile("([^\\s]+) - - \\[(.+)\\] \"([^\\s]+) (/[^\\s]*) HTTP/[^\\s]+\" [^\\s]+ ([0-9]+)");
  public static class AMapper extends Mapper<Object, Text, Text, IntWritable> {
    public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
      Matcher matcher = httplogPattern.matcher(value.toString());
      if (matcher.matches()) {
        int size = Integer.parseInt(matcher.group(5));
        context.write(new Text("msgSize"), new IntWritable(size));
      }
    }
  }
  public static class AReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
    public void reduce(Text key, Iterable<IntWritable> values, Context context) throws
IOException,
        InterruptedException {
```

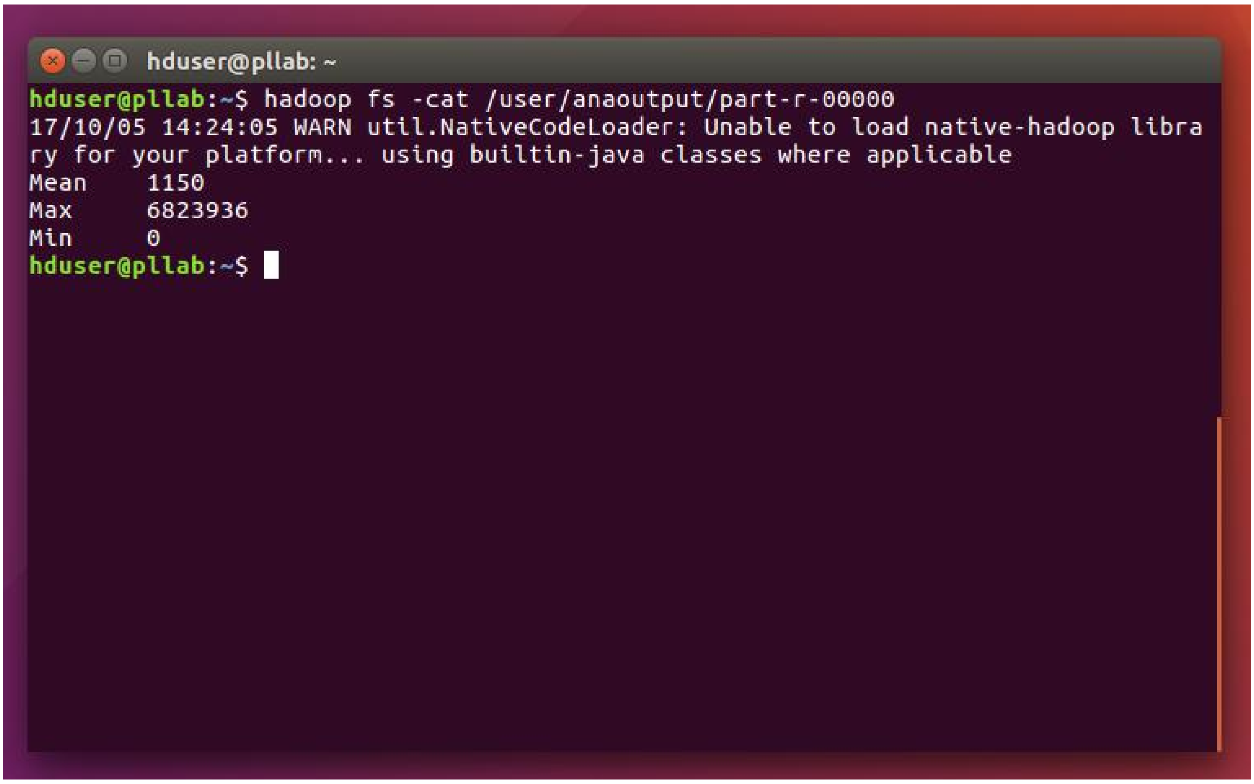
```
        double tot = 0;
        int count = 0;
        int min = Integer.MAX_VALUE;
        int max = 0;
        Iterator<IntWritable> iterator = values.iterator();
        while (iterator.hasNext()) {
            int value = iterator.next().get();
            tot = tot + value;
            count++;
            if (value < min) {
                min = value;
            }
            if (value > max) {
                max = value;
            }
        }
        context.write(new Text("Mean"), new IntWritable((int) tot / count));
        context.write(new Text("Max"), new IntWritable(max));
        context.write(new Text("Min"), new IntWritable(min));
    }
}
public static void main(String[] args) throws Exception {
    JobConf conf = new JobConf();
    String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
    if (otherArgs.length != 2) {
        System.err.println("Usage: <in> <out>");
        System.exit(2);
    }
    Job job = new Job(conf, "WebLogMessageSizeAggregator");
    job.setJarByClass(WebLogMessageSizeAggregator.class);
    job.setMapperClass(AMapper.class);
    job.setReducerClass(AReducer.class);
    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
```

```
        FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Steps to execute the Hadoop application:

- export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar
- hadoop com.suntools.javac.Main WebLogMessageSizeAggregator.java
- jar cf wcc.jar WebLogMessageSizeAggregator *.class
- hadoop fs -mkdir -p /user/input
- hadoop fs -copyFromLocal input /user/input1
- hadoop jar wcc.jar WebLogMessageSizeAggregator /user/input1 /user/output1
- hadoop fs -cat /user/output1/part-r-00000

OUTPUT:



```
hduser@pllab:~$ hadoop fs -cat /user/anaoutput/part-r-00000
17/10/05 14:24:05 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
Mean     1150
Max      6823936
Min      0
hduser@pllab:~$
```

Experiment –17

AIM: Performing Group-By using Map Reduce.

PROGRAM:

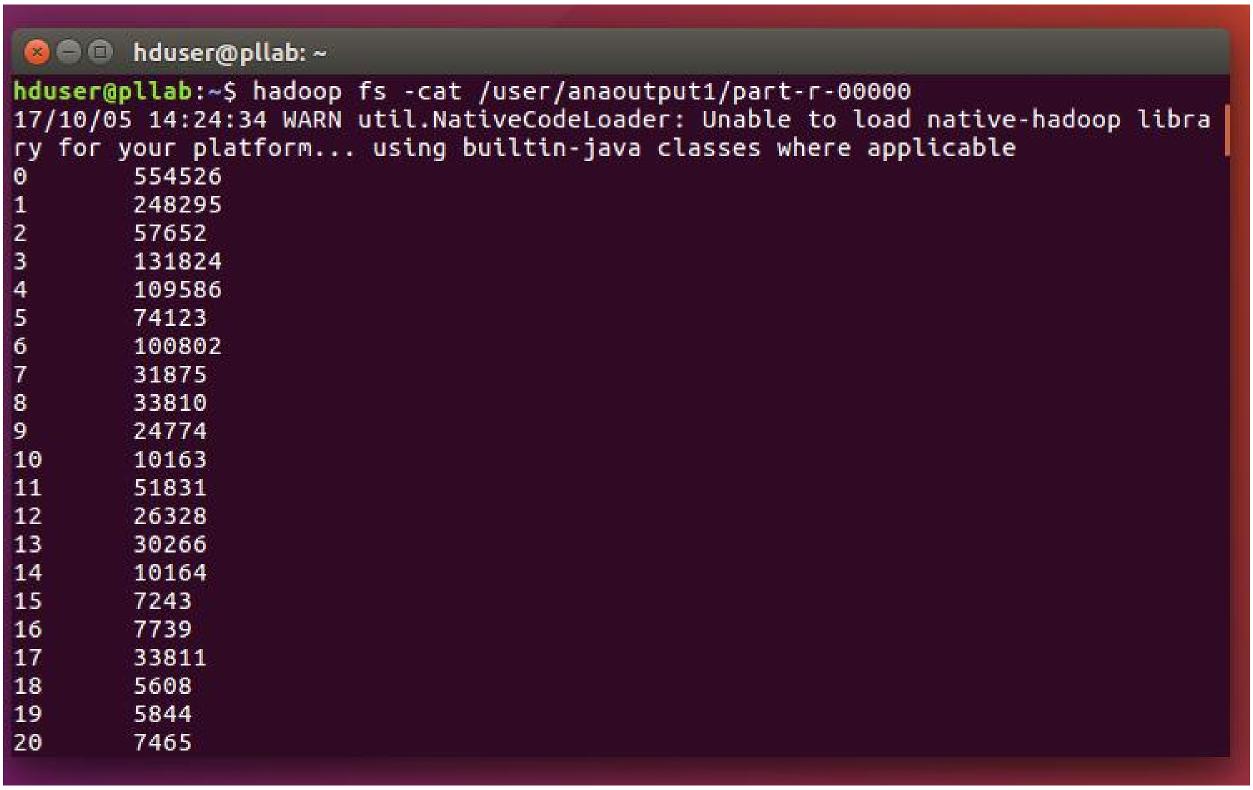
```
import java.io.IOException;
import java.util.Iterator;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
public class WeblogHitsByLinkProcessor  {
   public static final Pattern httplogPattern = Pattern
         .compile("([^\\s]+) - - \\[(.+)\\] \"([^\\s]+) (/[^\\s]*) HTTP/[^\\s]+\" [^\\s]+ ([0-9]+)");
   public static class AMapper extends Mapper<Object, Text, Text, IntWritable> {
         private final static IntWritable one = new IntWritable(1);
      private Text word = new Text();


      public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
         Matcher matcher = httplogPattern.matcher(value.toString());
         if (matcher.matches()) {
            String linkUrl = matcher.group(4);
            word.set(linkUrl);
            context.write(word, one);
         }
      }
   }
public static class AReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
private IntWritable result = new IntWritable();
public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException,
            InterruptedException {
         int sum = 0;
         for (IntWritable val : values) {
            sum += val.get();
         }
         result.set(sum);
         context.write(key, result);
      }
}
     public static void main(String[] args) throws Exception {
      JobConf conf = new JobConf();
      String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
      if (otherArgs.length != 2) {
         System.err.println("Usage: <in> <out>");
         System.exit(2);
      }
      Job job = new Job(conf, "WeblogHitsByLinkProcessor ");
      job.setJarByClass(WeblogHitsByLinkProcessor .class);
      job.setMapperClass(AMapper.class);
      job.setReducerClass(AReducer.class);
      job.setMapOutputKeyClass(Text.class);
      job.setMapOutputValueClass(IntWritable.class);
      FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
      FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
      System.exit(job.waitForCompletion(true) ? 0 : 1);
   }
}
```

Steps to execute the Hadoop application:

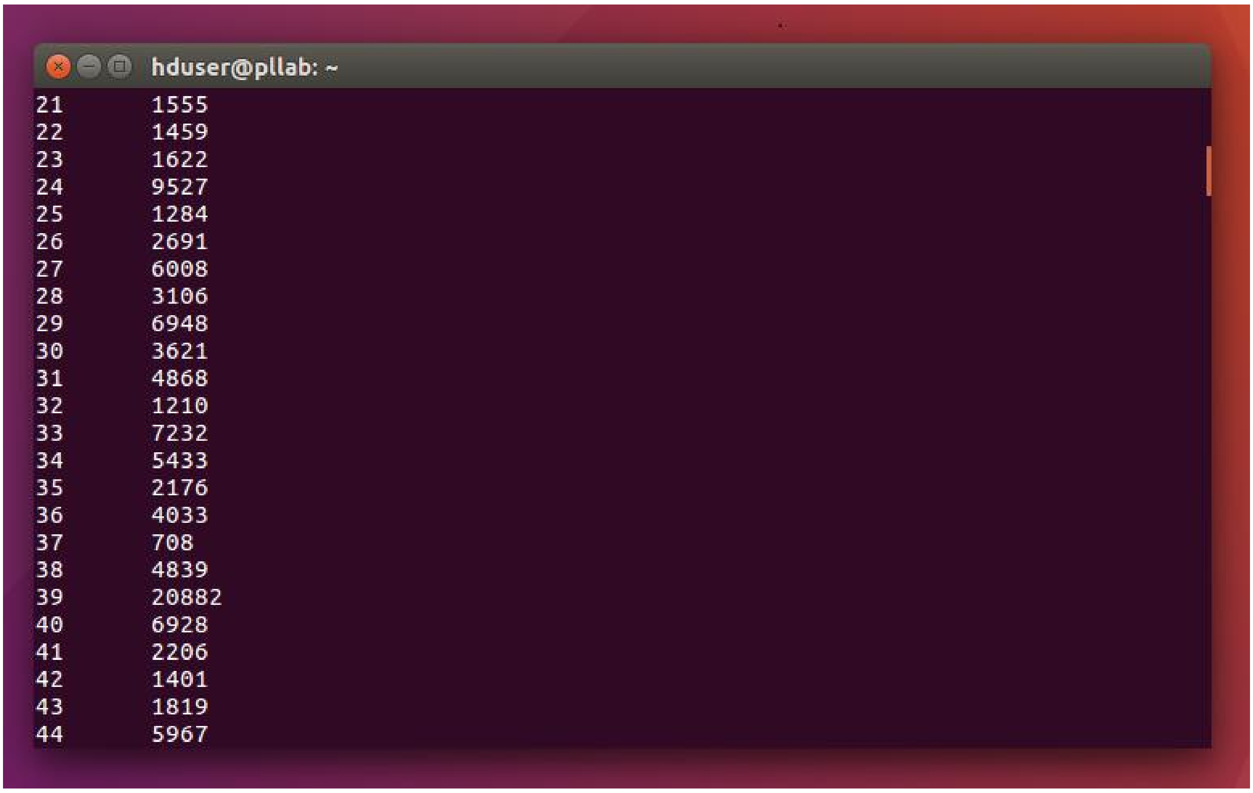
- export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar
- hadoop com.suntools.javac.Main WeblogHitsByLinkProcessor.java
- jar cf wcc.jar WeblogHitsByLinkProcessor *.class

- hadoop fs -mkdir -p /user/input
- hadoop fs -copyFromLocal input /user/input1
- hadoop jar wcc.jar WeblogHitsByLinkProcessor /user/input1 /user/output1
- hadoop fs -cat /user/output1/part-r-00000

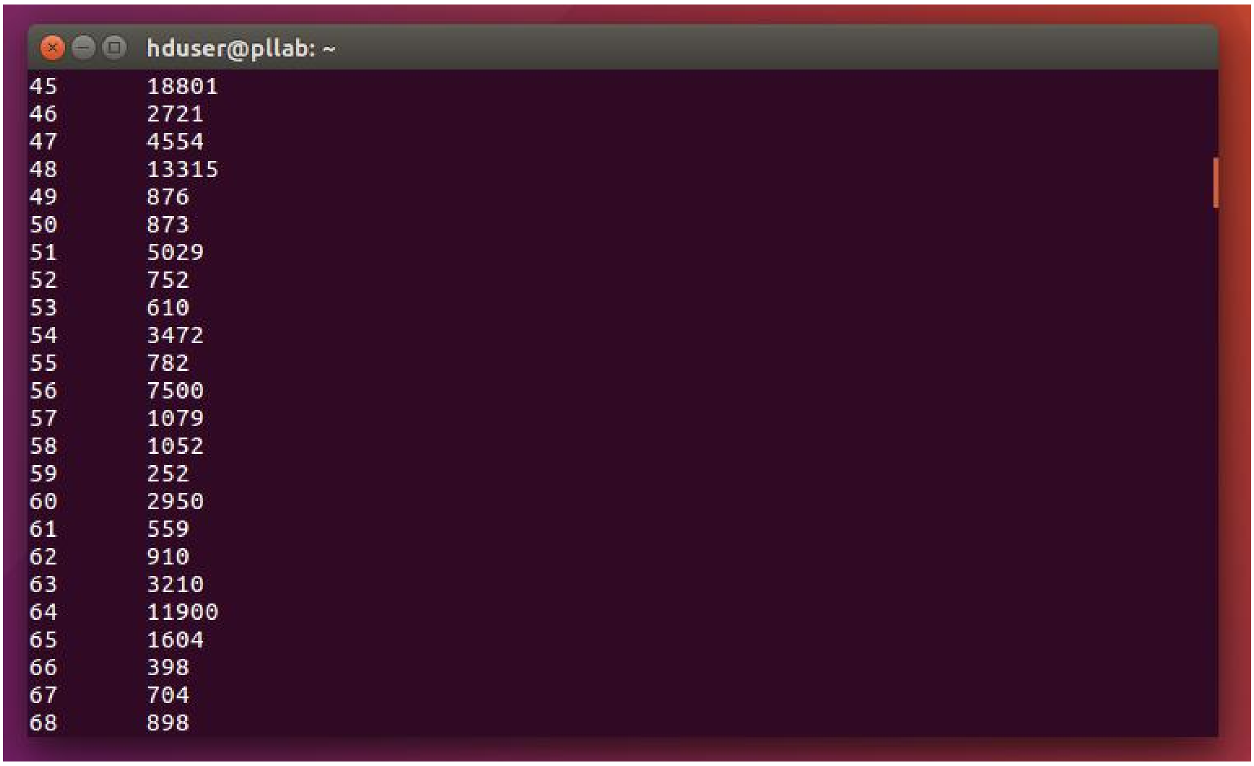
OUTPUT:



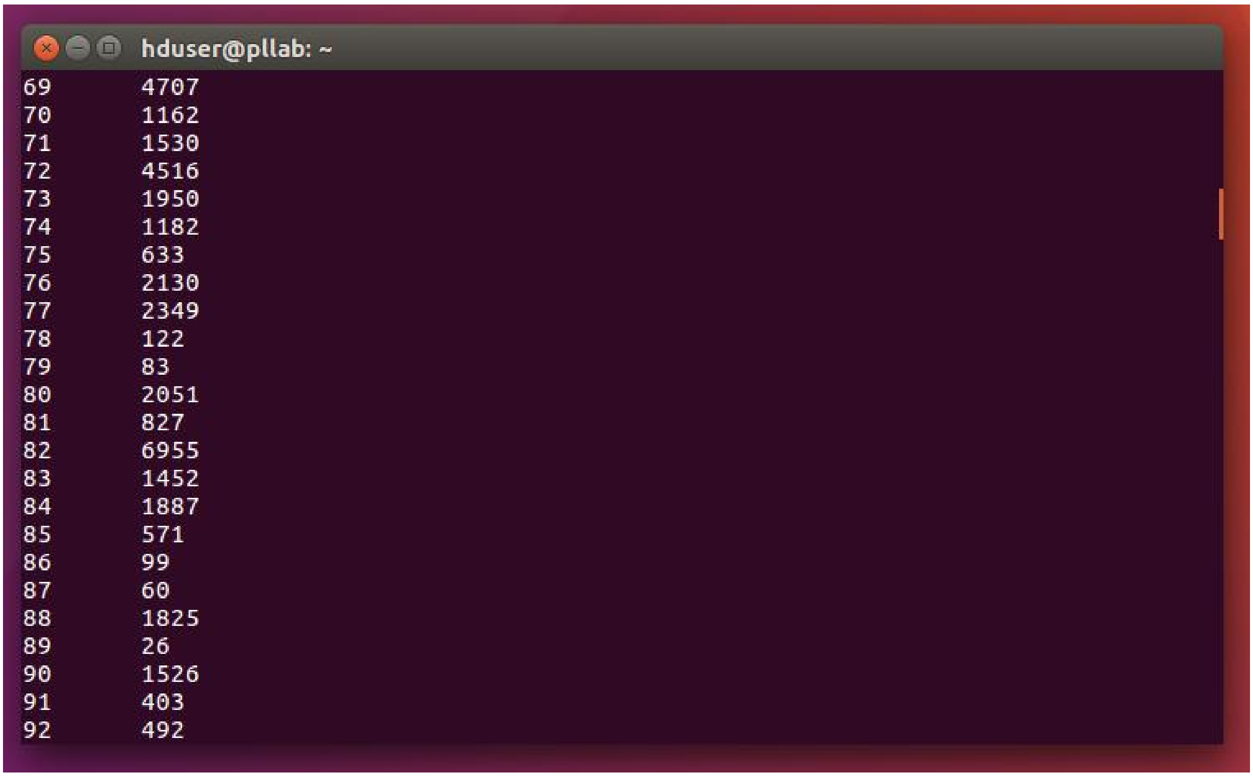
```
hduser@pllab:~$ hadoop fs -cat /user/anaoutput1/part-r-00000
17/10/05 14:24:34 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
0       554526
1       248295
2       57652
3       131824
4       109586
5       74123
6       100802
7       31875
8       33810
9       24774
10      10163
11      51831
12      26328
13      30266
14      10164
15      7243
16      7739
17      33811
18      5608
19      5844
20      7465
```



```
21      1555
22      1459
23      1622
24      9527
25      1284
26      2691
27      6008
28      3106
29      6948
30      3621
31      4868
32      1210
33      7232
34      5433
35      2176
36      4033
37      708
38      4839
39      20882
40      6928
41      2206
42      1401
43      1819
44      5967
```



```
45	18801
46	2721
47	4554
48	13315
49	876
50	873
51	5029
52	752
53	610
54	3472
55	782
56	7500
57	1079
58	1052
59	252
60	2950
61	559
62	910
63	3210
64	11900
65	1604
66	398
67	704
68	898
```



```
69	4707
70	1162
71	1530
72	4516
73	1950
74	1182
75	633
76	2130
77	2349
78	122
79	83
80	2051
81	827
82	6955
83	1452
84	1887
85	571
86	99
87	60
88	1825
89	26
90	1526
91	403
92	492
```

Experiment –18

AIM: Calculating frequency distributions and sorting using Map Reduce.

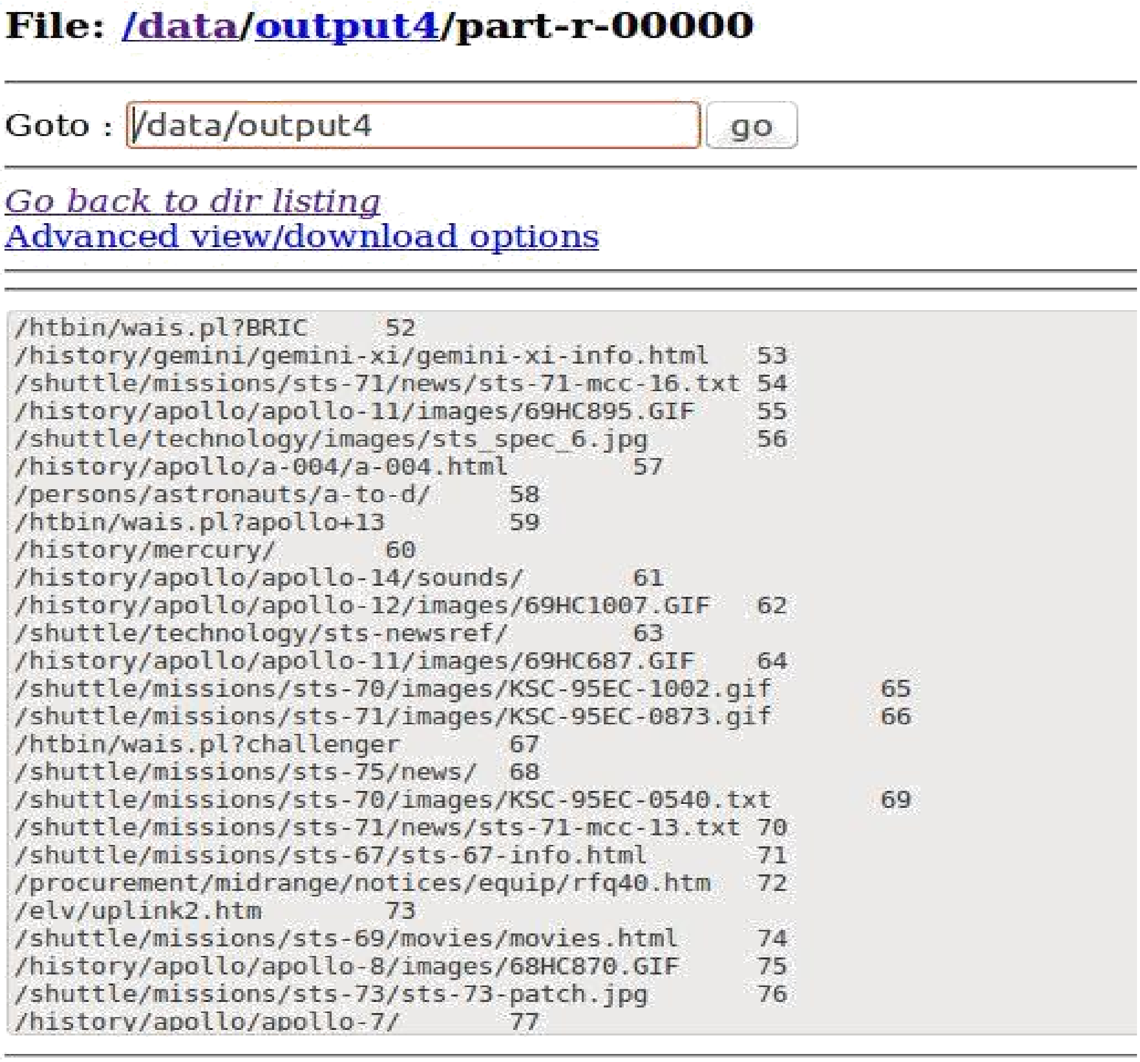
PROGRAM:

```
import java.io.IOException;
import java.util.Iterator;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
public class frequency  {

   public static final Pattern httplogPattern = Pattern
         .compile("([^\\s]+) - - \\[(.+)\\] \"([^\\s]+) (/[^\\s]*) HTTP/[^\\s]+\" [^\\s]+ ([0-9]+)");

   public static class AMapper extends Mapper<Object, Text, Text, IntWritable> {


         private final static IntWritable one = new IntWritable(1);
      private Text word = new Text();


      public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
         Matcher matcher = httplogPattern.matcher(value.toString());
         if (matcher.matches()) {
            String linkUrl = matcher.group(4);
            word.set(linkUrl);
            context.write(word, one);
         }
      }
   }

   public static class AReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
private IntWritable result = new IntWritable();
public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException,
            InterruptedException {
         int sum = 0;
         for (IntWritable val : values) {
            sum += val.get();
         }
         result.set(sum);
         context.write(key, result);
      }
}

      public static void main(String[] args) throws Exception {
      JobConf conf = new JobConf();
      String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
      if (otherArgs.length != 2) {
         System.err.println("Usage: <in> <out>");
         System.exit(2);
      }

      Job job = new Job(conf, "frequency ");
      job.setJarByClass(WeblogHitsByLinkProcessor .class);
      job.setMapperClass(AMapper.class);
      job.setReducerClass(AReducer.class);
      job.setMapOutputKeyClass(Text.class);
      job.setMapOutputValueClass(IntWritable.class);
      FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
      FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
      System.exit(job.waitForCompletion(true) ? 0 : 1);
   }
}
```

Steps to execute the Hadoop application:

- export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar
- hadoop com.suntools.javac.Main frequency.java
- jar cf wcc.jar frequency *.class
- hadoop fs -mkdir -p /user/input
- hadoop fs -copyFromLocal input /user/input1
- hadoop jar wcc.jar frequency /user/input1 /user/output1
- hadoop fs -cat /user/output1/part-r-00000

OUTPUT:



File: /data/output4/part-r-00000

Goto : /data/output4 go

Go back to dir listing
Advanced view/download options

```
/htbin/wais.pl?BRIC     52
/history/gemini/gemini-xi/gemini-xi-info.html   53
/shuttle/missions/sts-71/news/sts-71-mcc-16.txt 54
/history/apollo/apollo-11/images/69HC895.GIF    55
/shuttle/technology/images/sts_spec_6.jpg       56
/history/apollo/a-004/a-004.html        57
/persons/astronauts/a-to-d/     58
/htbin/wais.pl?apollo+13        59
/history/mercury/       60
/history/apollo/apollo-14/sounds/       61
/history/apollo/apollo-12/images/69HC1007.GIF   62
/shuttle/technology/sts-newsref/        63
/history/apollo/apollo-11/images/69HC687.GIF    64
/shuttle/missions/sts-70/images/KSC-95EC-1002.gif       65
/shuttle/missions/sts-71/images/KSC-95EC-0873.gif       66
/htbin/wais.pl?challenger       67
/shuttle/missions/sts-75/news/  68
/shuttle/missions/sts-70/images/KSC-95EC-0540.txt       69
/shuttle/missions/sts-71/news/sts-71-mcc-13.txt 70
/shuttle/missions/sts-67/sts-67-info.html       71
/procurement/midrange/notices/equip/rfq40.htm   72
/elv/uplink2.htm        73
/shuttle/missions/sts-69/movies/movies.html     74
/history/apollo/apollo-8/images/68HC870.GIF     75
/shuttle/missions/sts-73/sts-73-patch.jpg       76
/history/apollo/apollo-7/       77
```

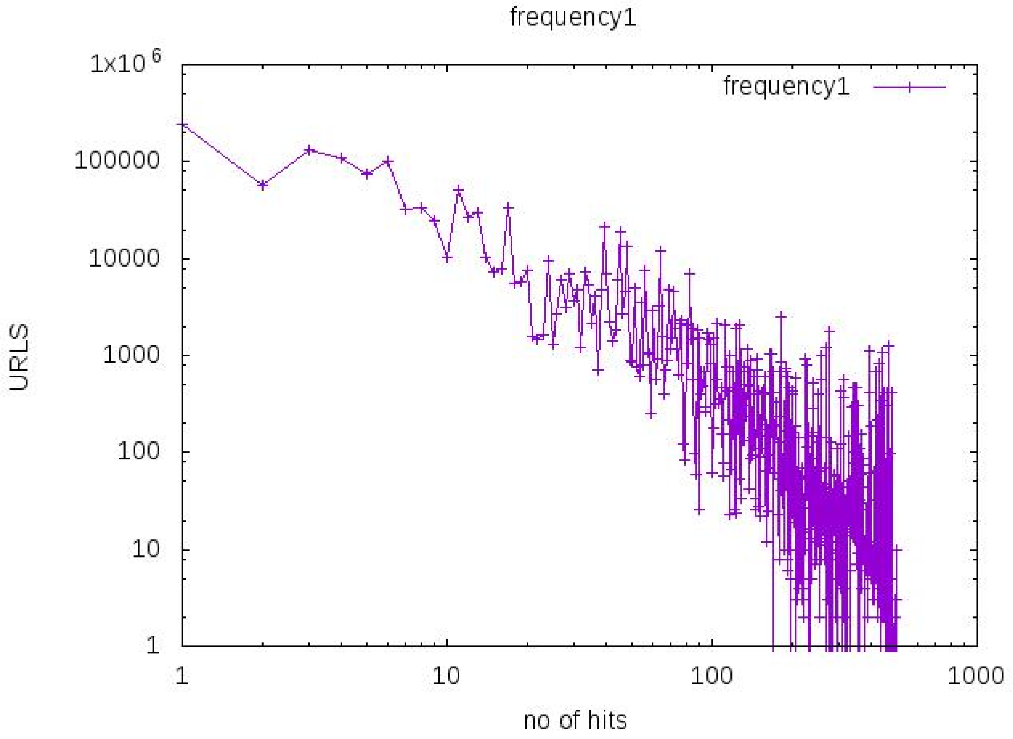
Experiment –19

AIM: Plotting the Hadoop results using GNU Plot

Download the results of the last recipe to a local computer by running the following **command**

```
> hadoop fs -get /data/output4/part-r-00000 2.data
> sudo gedit httpfreqdist.plot
set terminal png
set output "freqdist.png"
set title "Frequnecy Distribution of Hits by Url";
set ylabel "Number of Hits";
set xlabel "Urls (Sorted by hits)";
set key left top
set log y
set log x
plot "2.data" using 2 title "Frequency" with linespoints
>gnuplot httpfreqdist.plot
```

OUTPUT:



Experiment –20

AIM: Calculating histograms using Map Reduce.

Histograms are a common visualization technique that gives an empirical estimate of the probability density function (pdf) of a variable. Histograms are well-suited to a big data environment, because they can reduce the size of raw input data to a vector of counts. Each count is the number of observations that falls within each of a set of contiguous, numeric intervals or bins. The mapreduce function computes counts separately on multiple chunks of the data.

Then mapreduce sums the counts from all chunks. The map function and reduce function are both extremely simple in this example. Nevertheless, you can build flexible visualizations with the summary information that they collect.

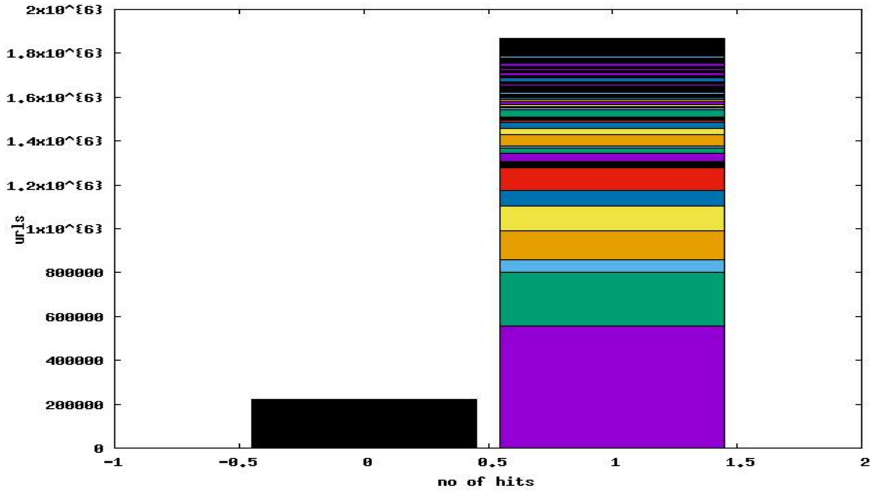
Visualize Results:

Plot the raw bin counts using the whole range of the data in gnuplot by using following commands having input from previous program output i.e., 2.data

Commands:

```
set term png medium
set output “histogram.png”
set xlabel “number of bits”
set ylabel “urls”
set style histogram clustered gap 2
set style histogram columnstacked
set boxwidth 0.9 relative
set style data histograms
set style fill solid 1.0 border-1
plot “2.data” using 1, “2.data” using 2
```

Output:



Experiment –21

AIM: Calculating scatter plots using Map Reduce.

Another useful tool while analyzing data is a Scatter plot: scatter plot is used to find the relationship between two measurements (dimensions). It plots the two dimensions against each other.

The following code segment shows the code for the mapper.

```
public void map(Object key, Text value, Context context) throws IOException,
InterruptedException
{
Matcher matcher = httplogPattern.matcher(value.toString());
if (matcher.matches())
{
int size = Integer.parseInt(matcher.group(5));
context.write(new IntWritable(size / 1024), one);
}
}
```

Map task receives each line in the log files as a different key-value pair. It parse the lines using regularexpressions and emits the file size as 1024-bytes blocks as the key and one as the values. Then, Hadoop collects the key-value pairs. sorts them, and then invokes the reducer once for each key. Each reducer walks through the values and calculates the count of page accesses for each file size.

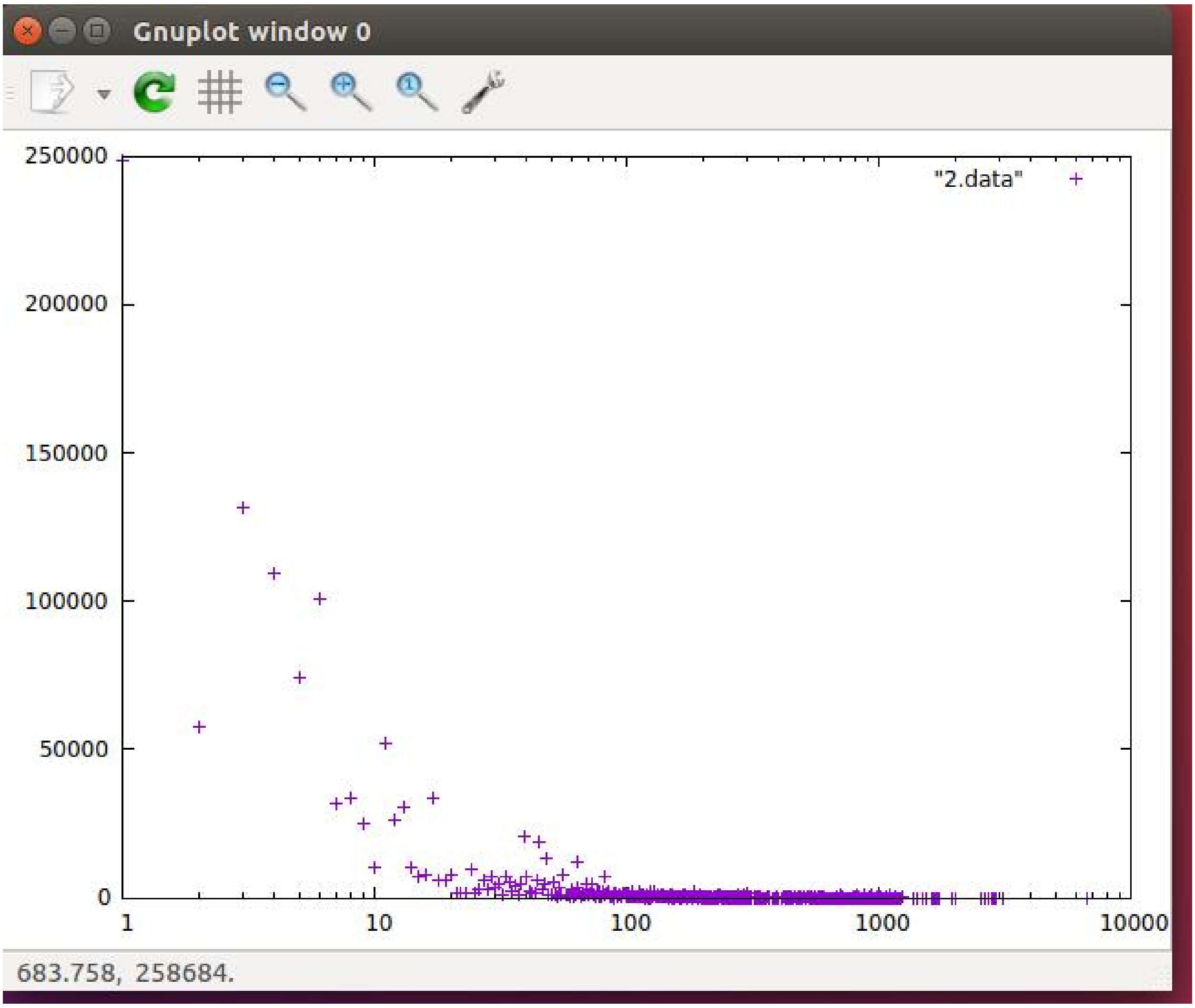
```
public void reduce(IntWritable key, Iterable values, Context context) throws IOException,
InterruptedException
{
int sum = 0;
for (IntWritableval : values)
{
sum += val.get();
}
context.write(key, new IntWritable(sum));
}
```

The following commands are used for plotter graph between the size of the web pages and the number of hits received by the web page in gnuplot

set logx

plot “2.data” using1:2title “2Node” with points.

OUTPUT:



Experiment –22

AIM: Parsing a Complex dataset with Hadoop.

JobConfigurationParser

A parser to parse and filter out interesting properties from job configuration.

```
// An example to parse and filter out job name
String conf_filename = .. // assume the job configuration filename here
// construct a list of interesting properties
List interestedProperties = new ArrayList();
interestedProperties.add("mapreduce.job.name");
JobConfigurationParser jcp = new JobConfigurationParser(interestedProperties);
InputStream in = new FileInputStream(conf_filename);
Properties parsedProperties = jcp.parse(in);
```

JobHistoryParser

A parser that parses job history files. It is an interface and actual implementations are defined as Enum in JobHistoryParserFactory . Note that RewindableInputStream is a wrapper class around InputStream to make the input stream rewindable.

```
// An example to parse a current job history file i.e a job history file for which the version is known
String filename = .. // assume the job history filename here
InputStream in = new FileInputStream(filename);
HistoryEvent event = null;
JobHistoryParser parser = new CurrentJHParser(in);
event = parser.nextEvent();
// process all the events
while (event != null) {
// ... process all event
event = parser.nextEvent();
}
// close the parser and the underlying stream
parser.close();
```

JobHistoryParserFactory

ProvidesjobHistoryParserFactory.getParser(org.apache.hadoop.tools.rumen.RewindableInputStream API to get a parser fro parsing the job history file

Note that this API can be used if the job history version is unknown.

```
// An example to parse a job history for which the version is not known
JobHistoryParserFactory.getParser()
String filename = .. // assume the job history filename here
InputStream in = new FileInputStream(filename);
RewindableInputStream ris = new RewindableInputStream(in);
// JobHistoryParserFactory will check and return a parser that can parse the file
JobHistoryParser parser = JobHistoryParserFactory.getParser(ris);
// now use the parser to parse the events
HistoryEvent event = parser.nextEvent();
while (event != null) {
// ... process the event
event = parser.nextEvent();
}
parser.close();
```

JobBuilder

Summarizes a job history file. JobHistoryUtils provides JobHistoryUtils.extractJobID(String) API for extracting job id from job history or job configuration files which can be used for instantiating JobBuilder . JobBuilder generates a LoggedJob object via JobBuilder.build().

```
// An example to summarize a current job history file 'filename' and the corresponding configuration file 'conf_filename'.
String filename = .. // assume the job history filename
String conf_filename = .. // assume the job configuration filename
InputStream jobConfInputStream = new FileInputStream(job_filename);
InputStream jobHistoryInputStream = new FileInputStream(conf_filename);
String jobID = TraceBuilder.extractJobID(job_filename);
JobBuilder jb = new JobBuilder(jobID);
// construct a list of interesting properties
List interestingProperties = new ArrayList();
// add the interesting properties here
interestingProperties.add("mapreduce.job.name");
JobConfigurationParser jcp =
new JobConfigurationParser(interestingProperties);
// parse the configuration file
jb.process(jcp.parse(jobConfInputStream));
```

```
// parse the job history file
JobHistoryParser parser = new
CurrentJHParser(jobHistoryInputStream);
try {
HistoryEvent e;
// read and process all the job history events
while ((e = parser.nextEvent()) != null) {
jobBuilder.process(e);
}
} finally {
parser.close();
}
LoggedJob job = jb.build();
```

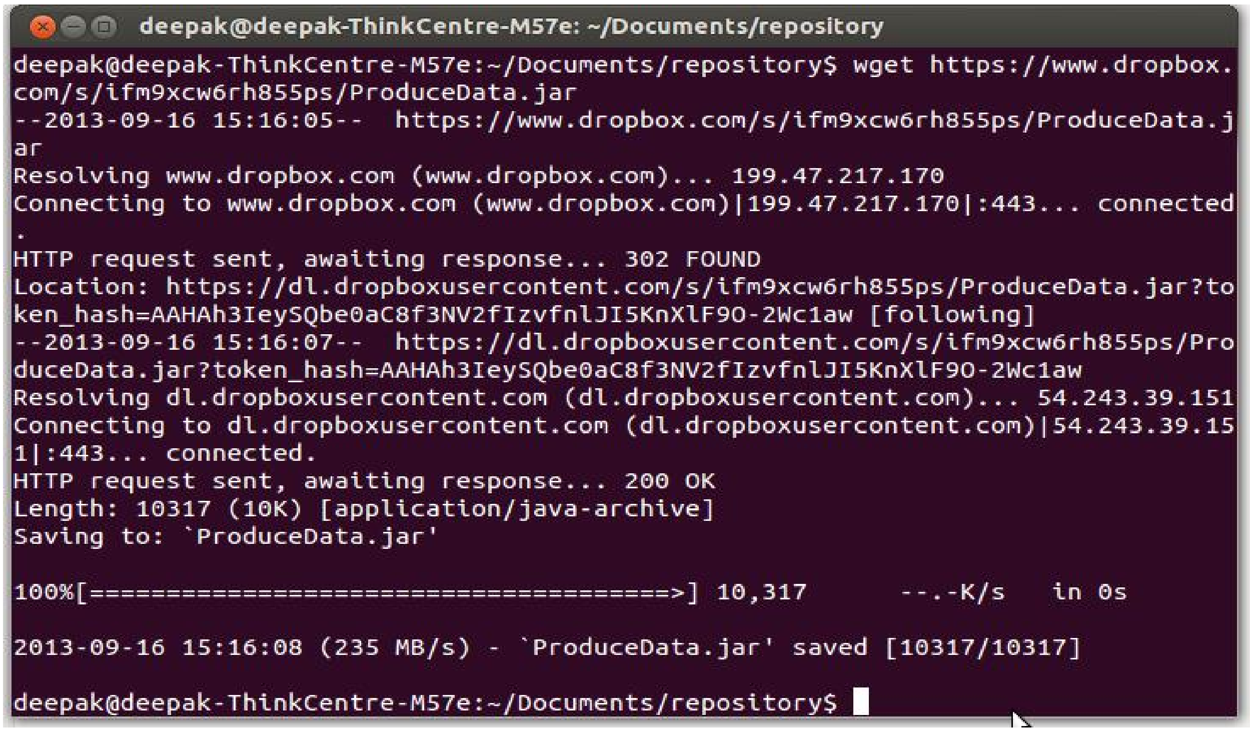
DefaultOutputter

Implements Outputter and writes JSON object in text format to the output file. DefaultOutputter can be initialized with the output filename.

```
// An example to summarize a current job history file represented by 'filename' and the
//configuration filename represented using  'conf_filename'. Also output the job summary to
'out.json' along with the cluster topology to 'topology.json'.
String filename = .. // assume the job history filename
String conf_filename = .. // assume the job configuration filename
Configuration conf = new Configuration();
DefaultOutputter do = new DefaultOutputter();
do.init("out.json", conf);
InputStream jobConfInputStream = new FileInputStream(filename);
InputStream jobHistoryInputStream = new FileInputStream(conf_filename);
// extract the job-id from the filename
String jobID = TraceBuilder.extractJobID(filename);
JobBuilder jb = new JobBuilder(jobID);
TopologyBuilder tb = new TopologyBuilder();
// construct a list of interesting properties
List interestingProperties = new ArrayList();
// add the interesting properties here
interestingProperties.add("mapreduce.job.name");
JobConfigurationParser jcp = new JobConfigurationParser(interestingProperties);
```

```
// parse the configuration file
tb.process(jcp.parse(jobConfInputStream));
// read the job history file and pass it to the TopologyBuilder.
JobHistoryParser parser = new CurrentJHParser(jobHistoryInputStream);
HistoryEvent e;
while ((e = parser.nextEvent()) != null) {
jb.process(e);
tb.process(e);
}
LoggedJob j = jb.build();
// serialize the job summary in json (text) format
do.output(j);
// close
do.close();
do.init("topology.json", conf);
// get the job summary using TopologyBuilder
LoggedNetworkTopology topology = topologyBuilder.build();
// serialize the cluster topology in json (text) format
do.output(topology);
// close
do.close();
```

OUTPUT:



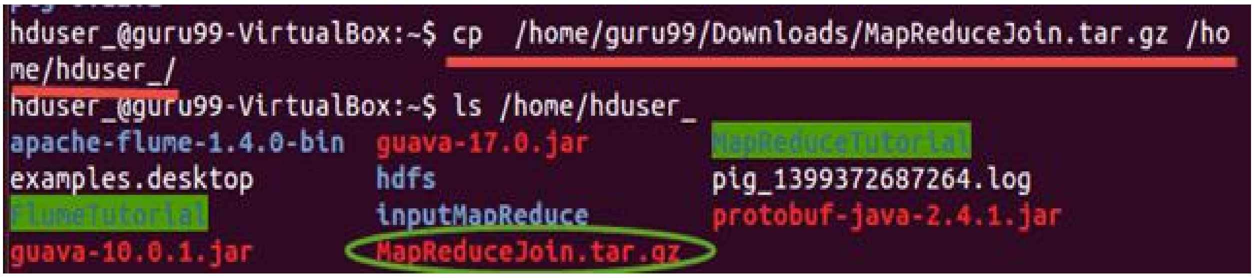
```
deepak@deepak-ThinkCentre-M57e:~/Documents/repository$ wget https://www.dropbox.
com/s/ifm9xcw6rh855ps/ProduceData.jar
--2013-09-16 15:16:05--  https://www.dropbox.com/s/ifm9xcw6rh855ps/ProduceData.j
ar
Resolving www.dropbox.com (www.dropbox.com)... 199.47.217.170
Connecting to www.dropbox.com (www.dropbox.com)|199.47.217.170|:443... connected
.
HTTP request sent, awaiting response... 302 FOUND
Location: https://dl.dropboxusercontent.com/s/ifm9xcw6rh855ps/ProduceData.jar?to
ken_hash=AAHAh3IeySQbe0aC8f3NV2fIzvfnlJI5KnXlF9O-2Wc1aw [following]
--2013-09-16 15:16:07--  https://dl.dropboxusercontent.com/s/ifm9xcw6rh855ps/Pro
duceData.jar?token_hash=AAHAh3IeySQbe0aC8f3NV2fIzvfnlJI5KnXlF9O-2Wc1aw
Resolving dl.dropboxusercontent.com (dl.dropboxusercontent.com)... 54.243.39.151
Connecting to dl.dropboxusercontent.com (dl.dropboxusercontent.com)|54.243.39.15
1|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10317 (10K) [application/java-archive]
Saving to: `ProduceData.jar'

100%[======================================>] 10,317      --.-K/s   in 0s

2013-09-16 15:16:08 (235 MB/s) - `ProduceData.jar' saved [10317/10317]

deepak@deepak-ThinkCentre-M57e:~/Documents/repository$ 
```

Experiment –23

AIM: Joining two datasets using Map Reduce.

Steps:

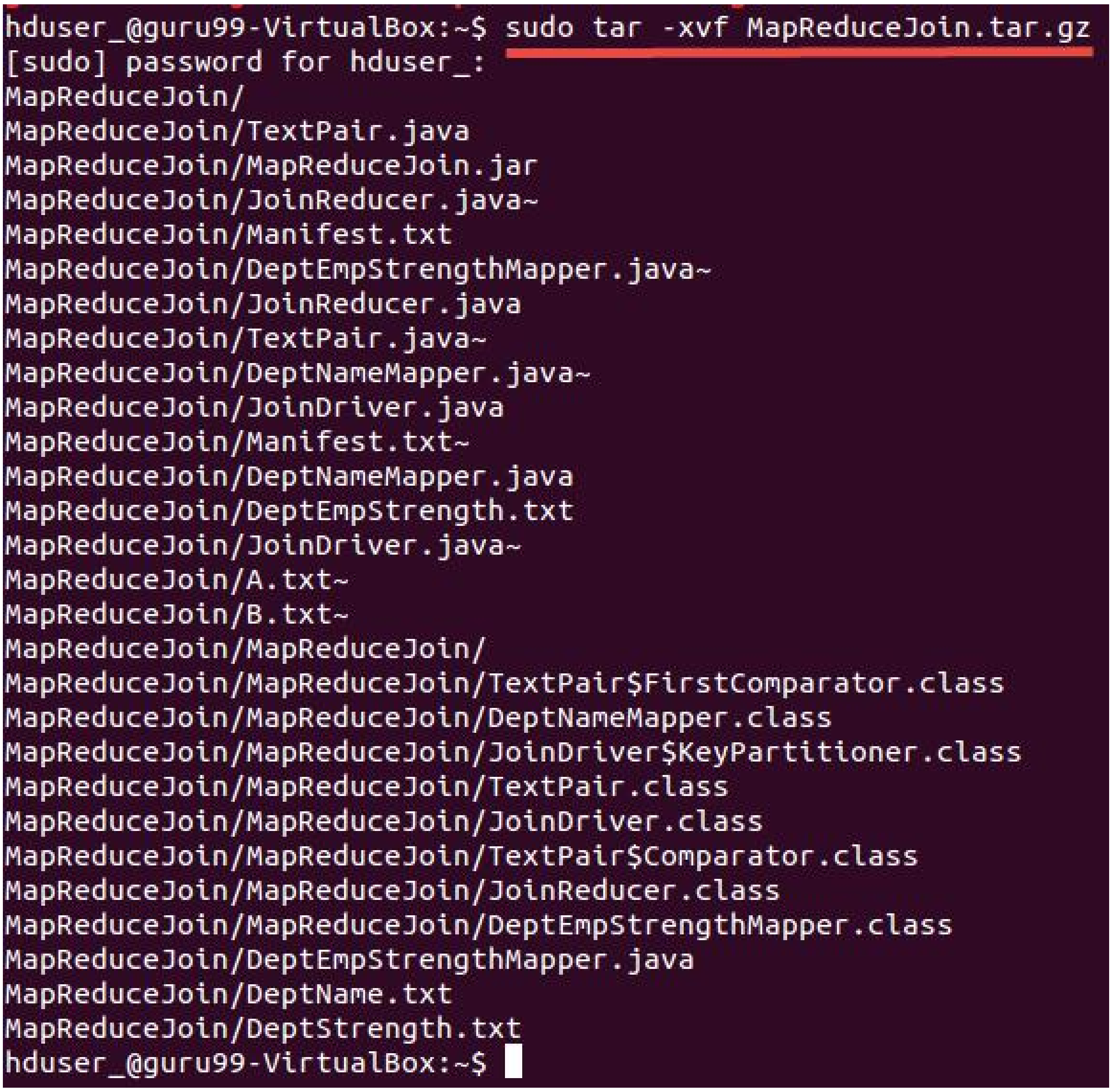
Step 1) Copy the zip file to location of your choice



```
hduser_@guru99-VirtualBox:~$ cp  /home/guru99/Downloads/MapReduceJoin.tar.gz /ho
me/hduser_/
hduser_@guru99-VirtualBox:~$ ls /home/hduser_
apache-flume-1.4.0-bin  guava-17.0.jar         MapReduceTutorial
examples.desktop        hdfs                   pig_1399372687264.log
FlumeTutorial           inputMapReduce         protobuf-java-2.4.1.jar
guava-10.0.1.jar        MapReduceJoin.tar.gz
```

Step 2) Uncompress the Zip File

sudo tar -xvf MapReduceJoin.tar.gz

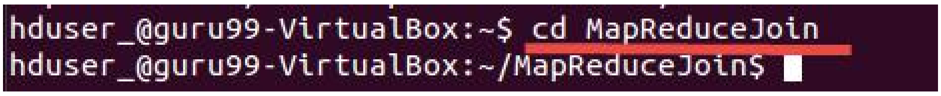


```
hduser_@guru99-VirtualBox:~$ sudo tar -xvf MapReduceJoin.tar.gz
[sudo] password for hduser_:
MapReduceJoin/
MapReduceJoin/TextPair.java
MapReduceJoin/MapReduceJoin.jar
MapReduceJoin/JoinReducer.java~
MapReduceJoin/Manifest.txt
MapReduceJoin/DeptEmpStrengthMapper.java~
MapReduceJoin/JoinReducer.java
MapReduceJoin/TextPair.java~
MapReduceJoin/DeptNameMapper.java~
MapReduceJoin/JoinDriver.java
MapReduceJoin/Manifest.txt~
MapReduceJoin/DeptNameMapper.java
MapReduceJoin/DeptEmpStrength.txt
MapReduceJoin/JoinDriver.java~
MapReduceJoin/A.txt~
MapReduceJoin/B.txt~
MapReduceJoin/MapReduceJoin/
MapReduceJoin/MapReduceJoin/TextPair$FirstComparator.class
MapReduceJoin/MapReduceJoin/DeptNameMapper.class
MapReduceJoin/MapReduceJoin/JoinDriver$KeyPartitioner.class
MapReduceJoin/MapReduceJoin/TextPair.class
MapReduceJoin/MapReduceJoin/JoinDriver.class
MapReduceJoin/MapReduceJoin/TextPair$Comparator.class
MapReduceJoin/MapReduceJoin/JoinReducer.class
MapReduceJoin/MapReduceJoin/DeptEmpStrengthMapper.class
MapReduceJoin/DeptEmpStrengthMapper.java
MapReduceJoin/DeptName.txt
MapReduceJoin/DeptStrength.txt
hduser_@guru99-VirtualBox:~$ 
```

Step 3)

Go to directory MapReduceJoin/

cd MapReduceJoin/

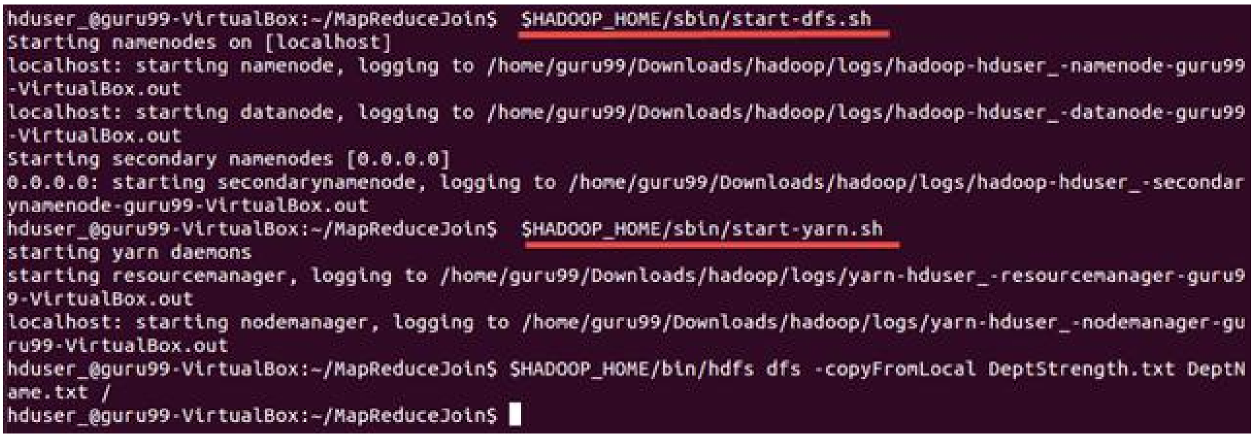


```
hduser_@guru99-VirtualBox:~$ cd MapReduceJoin
hduser_@guru99-VirtualBox:~/MapReduceJoin$
```

Step 4) Start Hadoop

$HADOOP_HOME/sbin/start-dfs.sh

$HADOOP_HOME/sbin/start-yarn.sh

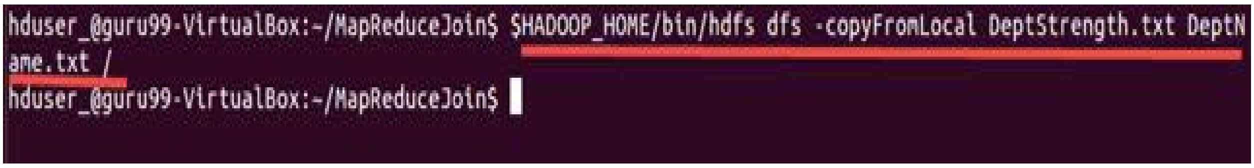


```
hduser_@guru99-VirtualBox:~/MapReduceJoin$  $HADOOP_HOME/sbin/start-dfs.sh
Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/guru99/Downloads/hadoop/logs/hadoop-hduser_-namenode-guru99
-VirtualBox.out
localhost: starting datanode, logging to /home/guru99/Downloads/hadoop/logs/hadoop-hduser_-datanode-guru99
-VirtualBox.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/guru99/Downloads/hadoop/logs/hadoop-hduser_-secondar
ynamenode-guru99-VirtualBox.out
hduser_@guru99-VirtualBox:~/MapReduceJoin$  $HADOOP_HOME/sbin/start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /home/guru99/Downloads/hadoop/logs/yarn-hduser_-resourcemanager-guru9
9-VirtualBox.out
localhost: starting nodemanager, logging to /home/guru99/Downloads/hadoop/logs/yarn-hduser_-nodemanager-gu
ru99-VirtualBox.out
hduser_@guru99-VirtualBox:~/MapReduceJoin$ $HADOOP_HOME/bin/hdfs dfs -copyFromLocal DeptStrength.txt DeptN
ame.txt /
hduser_@guru99-VirtualBox:~/MapReduceJoin$
```

Step 5) DeptStrength.txt and DeptName.txt are the input files used for this program.

These file needs to be copied to HDFS using below command-

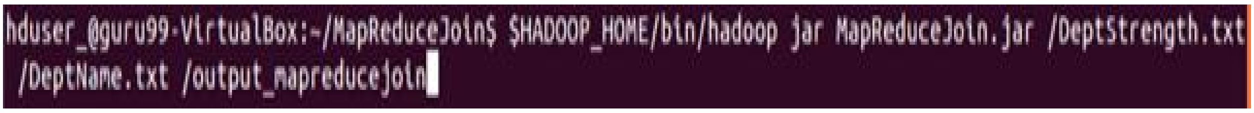
$HADOOP_HOME/bin/hdfs dfs -copyFromLocal DeptStrength.txt DeptName.txt /



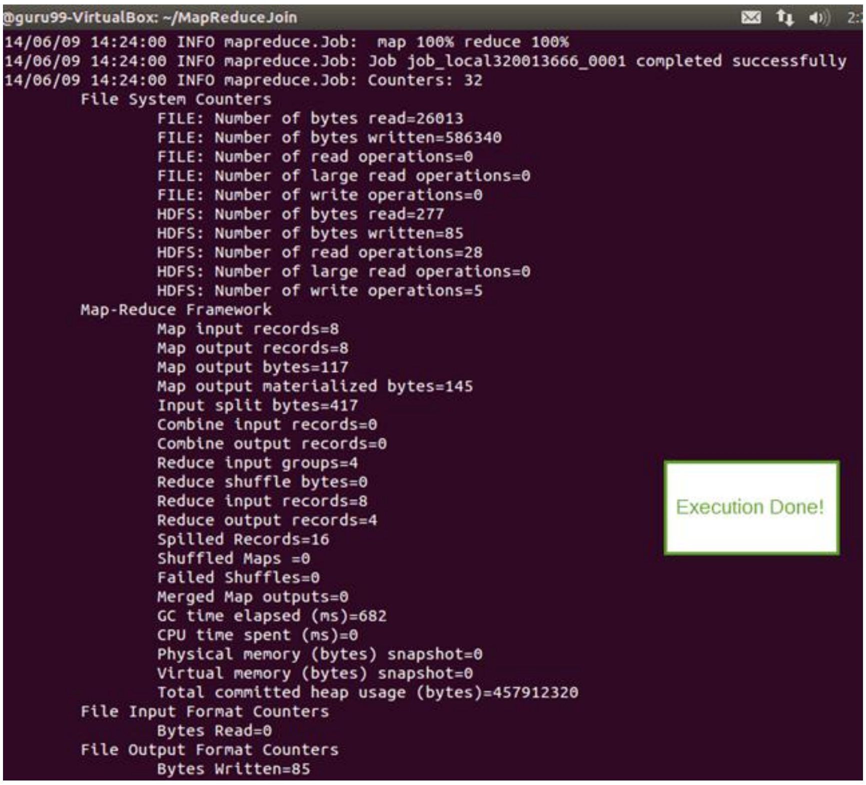
```
hduser_@guru99-VirtualBox:~/MapReduceJoin$ $HADOOP_HOME/bin/hdfs dfs -copyFromLocal DeptStrength.txt DeptN
ame.txt /
hduser_@guru99-VirtualBox:~/MapReduceJoin$
```

Step 6) Run the program using below command-

$HADOOP_HOME/bin/hadoop jar MapReduceJoin.jar /DeptStrength.txt /DeptName.txt /output_mapreducejoin



```
hduser_@guru99-VirtualBox:~/MapReduceJoin$ $HADOOP_HOME/bin/hadoop jar MapReduceJoin.jar /DeptStrength.txt
 /DeptName.txt /output_mapreducejoin
```



```
@guru99-VirtualBox: ~/MapReduceJoin
14/06/09 14:24:00 INFO mapreduce.Job:  map 100% reduce 100%
14/06/09 14:24:00 INFO mapreduce.Job: Job job_local320013666_0001 completed successfully
14/06/09 14:24:00 INFO mapreduce.Job: Counters: 32
        File System Counters
                FILE: Number of bytes read=26013
                FILE: Number of bytes written=586340
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=277
                HDFS: Number of bytes written=85
                HDFS: Number of read operations=28
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=5
        Map-Reduce Framework
                Map input records=8
                Map output records=8
                Map output bytes=117
                Map output materialized bytes=145
                Input split bytes=417
                Combine input records=0
                Combine output records=0
                Reduce input groups=4
                Reduce shuffle bytes=0
                Reduce input records=8
                Reduce output records=4
                Spilled Records=16
                Shuffled Maps =0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=682
                CPU time spent (ms)=0
                Physical memory (bytes) snapshot=0
                Virtual memory (bytes) snapshot=0
                Total committed heap usage (bytes)=457912320
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=85
```

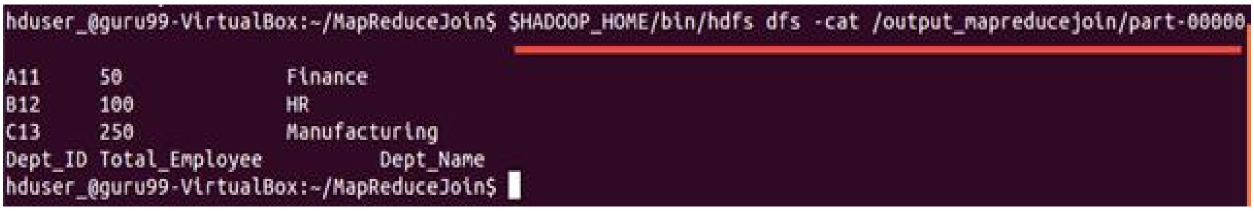
Step 7)

After execution, output file (named 'part-00000') will stored in the directory

/output_mapreducejoin on HDFS

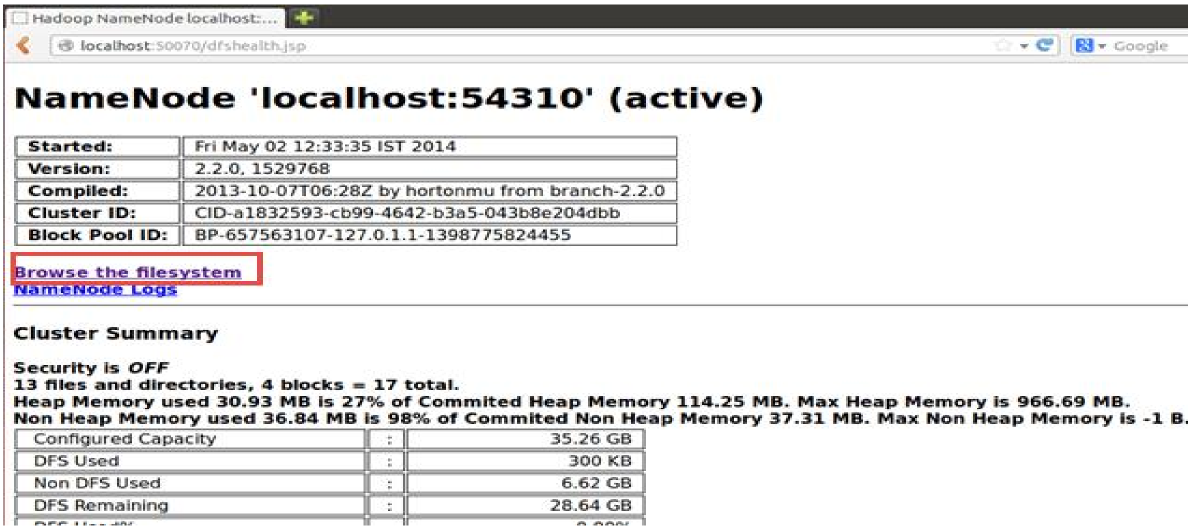
Results can be seen using the command line interface

$HADOOP_HOME/bin/hdfs dfs -cat /output_mapreducejoin/part-00000



```
hduser_@guru99-VirtualBox:~/MapReduceJoin$ $HADOOP_HOME/bin/hdfs dfs -cat /output_mapreducejoin/part-00000

A11     50              Finance
B12     100             HR
C13     250             Manufacturing
Dept_ID Total_Employee          Dept_Name
hduser_@guru99-VirtualBox:~/MapReduceJoin$
```

Results can also be seen via web interface as



NameNode 'localhost:54310' (active)

Started:	Fri May 02 12:33:35 IST 2014
Version:	2.2.0, 1529768
Compiled:	2013-10-07T06:28Z by hortonmu from branch-2.2.0
Cluster ID:	CID-a1832593-cb99-4642-b3a5-043b8e204dbb
Block Pool ID:	BP-657563107-127.0.1.1-1398775824455

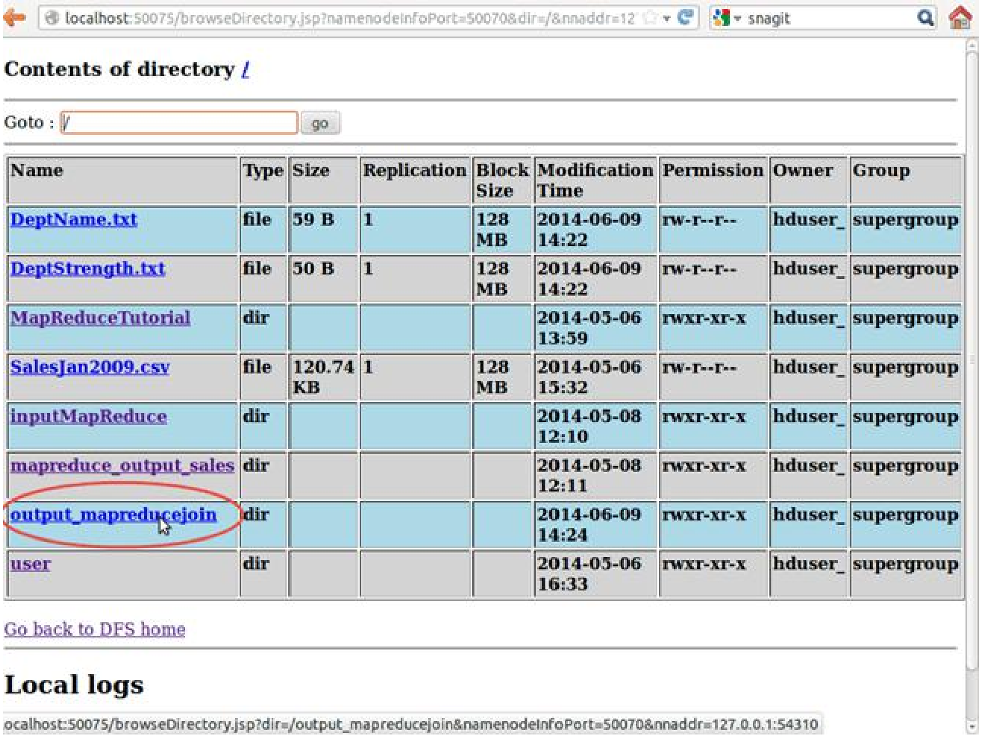
Browse the filesystem
NameNode Logs

Cluster Summary

Security is *OFF*
13 files and directories, 4 blocks = 17 total.
Heap Memory used 30.93 MB is 27% of Commited Heap Memory 114.25 MB. Max Heap Memory is 966.69 MB.
Non Heap Memory used 36.84 MB is 98% of Commited Non Heap Memory 37.31 MB. Max Non Heap Memory is -1 B.

Configured Capacity	:	35.26 GB
DFS Used	:	300 KB
Non DFS Used	:	6.62 GB
DFS Remaining	:	28.64 GB

Now select 'Browse the filesystem' and navigate upto /output_mapreducejoin



localhost:50075/browseDirectory.jsp?namenodeInfoPort=50070&dir=/&nnaddr=12
snagit

Contents of directory /

Goto : / go

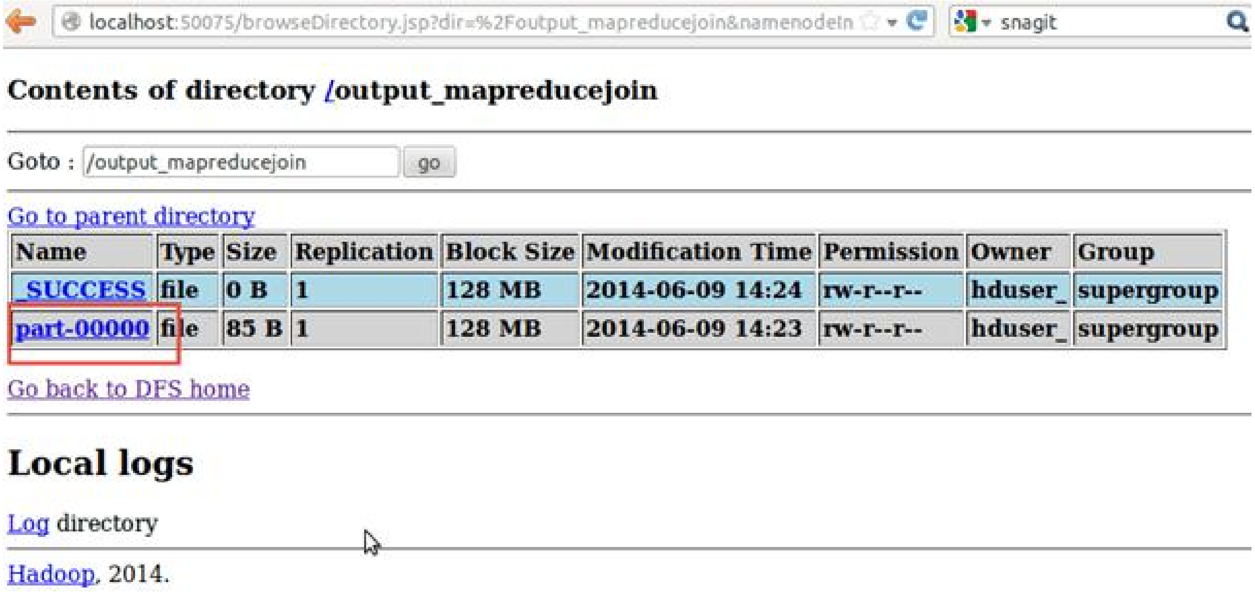
Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
DeptName.txt	file	59 B	1	128 MB	2014-06-09 14:22	rw-r--r--	hduser_	supergroup
DeptStrength.txt	file	50 B	1	128 MB	2014-06-09 14:22	rw-r--r--	hduser_	supergroup
MapReduceTutorial	dir				2014-05-06 13:59	rwxr-xr-x	hduser_	supergroup
SalesJan2009.csv	file	120.74 KB	1	128 MB	2014-05-06 15:32	rw-r--r--	hduser_	supergroup
inputMapReduce	dir				2014-05-08 12:10	rwxr-xr-x	hduser_	supergroup
mapreduce_output_sales	dir				2014-05-08 12:11	rwxr-xr-x	hduser_	supergroup
output_mapreducejoin	dir				2014-06-09 14:24	rwxr-xr-x	hduser_	supergroup
user	dir				2014-05-06 16:33	rwxr-xr-x	hduser_	supergroup

Go back to DFS home

Local logs

ocalhost:50075/browseDirectory.jsp?dir=/output_mapreducejoin&namenodeInfoPort=50070&nnaddr=127.0.0.1:54310

Open part-r-00000



localhost:50075/browseDirectory.jsp?dir=%2Foutput_mapreducejoin&namenodeIn
snagit

Contents of directory /output_mapreducejoin

Goto : /output_mapreducejoin go

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
_SUCCESS	file	0 B	1	128 MB	2014-06-09 14:24	rw-r--r--	hduser_	supergroup
part-00000	file	85 B	1	128 MB	2014-06-09 14:23	rw-r--r--	hduser_	supergroup

Go back to DFS home

Local logs

Log directory

Hadoop, 2014.