EXPERIMENT 1

**AIM**: Create SSH connection on system installed with Ubuntu 14.05 LTS

**PROCEDURE:**

1. SSH is a network protocol for securely communicating between computers.
2. SSH consists of 2 keys i.e. public key and private key.
3. The SSH command is used from logging into the remote machine, transferring files between the two machines, and for executing commands on the remote machine.
4. SSH uses port 22 to connect from one computer to another
5. Steps for installation of SSH

* sudo apt-get update
* sudo apt-get install default-jdk
* sudo java –version
* sudo apt-get install ssh
* sudo apt-get install rsync
* ssh-keygen -t dsa -P ' ' -f ~/.ssh/id\_dsa
* cat ~/.ssh/id\_dsa.pub >> ~/.ssh/authorized\_keys

**OUTPUT**:

**Step 1** – Update repositories.

root@mail:/# apt-get update

root@mail:/# apt-get upgrade

**Step 2** – Install [SSH Server](https://help.ubuntu.com/community/SSH)

root@mail:/# apt-get install openssh-server

**Basic Configuration:**

**Step 3** – After installation I will show how to configure ssh server. Open ssh config file with the following command:

root@mail:/# nano /etc/ssh/sshd\_config

**Step 4** – If you want to change ssh port you have to find ‘Port’ line and change the number of the port. For example I will change to 22222.

Port 22222

**Step 5** – I will set max login attempts to be 3. **After 3 wrong login attempts you will disconnect**. This is very important for security of your server and this can be used for prevention from **brute force attack** (see my **Theme 4**). Add this line bellow **Port**:

MaxAuthTries 3

**Step 6** – Allow certain users to login on your server and deny all other users. I will add ‘zimbra’ users because my Zimbra Mail Serve should have access. For more information about **Zimbra Mail Server configuration read theme 12**. Add the following line at the end of the file and after that save the file /etc/ssh/sshd\_config.

AllowUsers mslavov zimbra

**Step 7** – Restart ssh service with the following command:

root@mail:/# service ssh restart

**Advanced Configuration:**

**Step 8** – Create folder, change permission and navigate to new folder with the following commands:

root@mail:/# mkdir .ssh/; chmod 700 .ssh/; cd .ssh/;

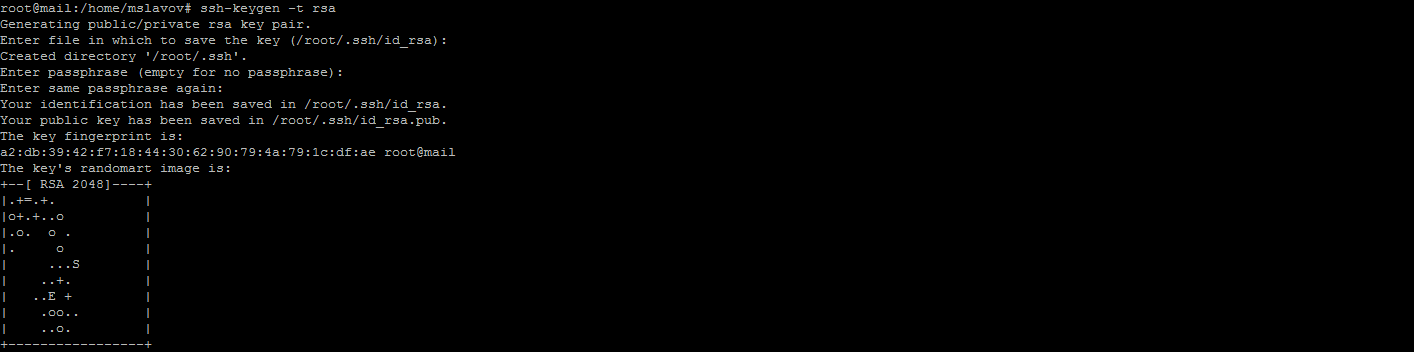
**Step 9** – Create folder, change permission and navigate to new folder with the following commands:

root@mail:/.ssh# touch authorized\_keys; chmod 600 authorized\_keys

**Step 10** – Show new files.

root@mail:/.ssh# ls -ltra

**Step 11** – Generate Keys – If you ‘Enter passphrase’ you must remember it and use it in the following steps:



**Step 12** – Append the public key to authorized\_keys and remove the uploaded copy.

root@mail:/.ssh# cat id\_rsa.pub >> authorized\_keys

EXPERIMENT 2

**AIM**:

Download and install hadoop stable version on the above system.

**PROCEDURE**:

1. Hadoop is a open-source framework that allows to store and process big data in a distributed environment across cluster of computers using simple programming models.
2. Hadoop file system was developed using distributed file system design.
3. Hadoop distributed file system holds very large amount of data and provides easier access.
4. Steps for installing hadoop stable version:

* $ wget -c <http://mirror.olnevhost.net/pub/apache/hadoop/common/current/hadoop-2.6.0.tar.gz>
* Download the hadoop tar file of either 2.6.0 or 2.6.5.
* sudo mv hadoop-2.6.0.tar.gz Desktop/
* cd Desktop/
* $ sudo tar -xvcf hadoop-2.6.0.tar.gz.
* $ sudo tar -zxcf hadoop-2.6.0.tar.gz
* sudo mv hadoop-2.6.0 /usr/local/hadoop
* update-alternatives -config java
* update-alternatives --config java
* sudo gedit ~/.bashrc (A bash file will be opened)
* sudo gedit /usr/local/hadoop/etc/hadoop/hadoop-env.sh
* On line 25 paste the path which we have copied from the bash file.
* source ~/.bashrc
* cd /usr/local/hadoop/etc/hadoop/
* sudo gedit core-site.xml
* At the end of the file write the following code
* <configuration>  
                    <property>  
                        <name>fs.defaultFS</name>  
                        <value>hdfs://localhost:9000</value>  
                    </property>  
            </configuration>
* sudo gedit yarn-site.xml
* <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>
* $ sudo cp mapred.site.xml.template mapred-site.xml.template mapred-site.xml
* sudo gedit mapred-site.xml
* <configuration>  
                    <property>  
                        <name>mapreduce.framework.name</name>  
                        <value>yarn</value>  
                    </property>  
            </configuration>
* sudo mkdir –p /usr/local/hadoop/hadoop\_data/hdfs/namenode/
* sudo mkdir –p /usr/local/hadoop/hadoop\_data/hdfs/datanode/
* sudo gedit /usr/local/hadoop/etc/hadoop/hdfs-site.xml
* <configuration>  
                    <property>  
                        <name>dfs.replication</name>  
                        <value>1</value>  
                    </property>  
                    <property>  
                        <name>dfs.namenode.name.dir</name>  
                   <value>file:/usr/local/hadoop/hadoop\_data/hdfs/namenode</value>  
                    </property>  
                    <property>  
                        <name>dfs.datanode.data.dir</name>  
                    <value>file:/usr/local/hadoop/hadoop\_store/hdfs/datanode</value>  
                    </property>  
            </configuration>
* sudo gedit ~/.bashrc
* on the line 129 correct the double quotes.
* source ~/.bashrc
* hdfs namenode-format
* start-all.sh
* jps
* Now we will get a list of installed hadoop files
* Jps
* Namenode
* secondaryNameNode
* node manager
* datanode
* resourcemanager

**Successfully installed hadoop.**

**OUTPUT:**

**Install Hadoop:**

hduser@laptop:~$ wget http://mirrors.sonic.net/apache/hadoop/common/hadoop-2.6.0/hadoop-2.6.0.tar.gz

hduser@laptop:~$ tar xvzf hadoop-2.6.0.tar.gz

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

hduser@laptop:~/hadoop-2.6.0$ sudo mv \* /usr/local/hadoop

[sudo] password for hduser:

hduser is not in the sudoers file. This incident will be reported.

hduser@laptop:~/hadoop-2.6.0$ su k

Password:

k@laptop:/home/hduser$ sudo adduser hduser sudo

[sudo] password for k:

Adding user `hduser' to group `sudo' ...

Adding user hduser to group sudo

Done.

**Now, the hduser has root priviledge, we can move the Hadoop installation to the /usr/local/hadoop directory without any problem:**

k@laptop:/home/hduser$ sudo su hduser

hduser@laptop:~/hadoop-2.6.0$ sudo mv \* /usr/local/hadoop

hduser@laptop:~/hadoop-2.6.0$ sudo chown -R hduser:hadoop /usr/local/hadoop

**Setup Configuration Files:**

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

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

**1. ~/.bashrc**:

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

hduser@laptop update-alternatives --config java

There is only one alternative in link group java (providing /usr/bin/java): /usr/lib/jvm/java-7-openjdk-amd64/jre/bin/java

Nothing to configure.

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

hduser@laptop:~$ vi ~/.bashrc

#HADOOP VARIABLES START

export JAVA\_HOME=/usr/lib/jvm/java-7-openjdk-amd64

export HADOOP\_INSTALL=/usr/local/hadoop

export PATH=$PATH:$HADOOP\_INSTALL/bin

export PATH=$PATH:$HADOOP\_INSTALL/sbin

export HADOOP\_MAPRED\_HOME=$HADOOP\_INSTALL

export HADOOP\_COMMON\_HOME=$HADOOP\_INSTALL

export HADOOP\_HDFS\_HOME=$HADOOP\_INSTALL

export YARN\_HOME=$HADOOP\_INSTALL

export HADOOP\_COMMON\_LIB\_NATIVE\_DIR=$HADOOP\_INSTALL/lib/native

export HADOOP\_OPTS="-Djava.library.path=$HADOOP\_INSTALL/lib"

#HADOOP VARIABLES END

hduser@laptop:~$ source ~/.bashrc

note that the JAVA\_HOME should be set as the path just before the '.../bin/':

hduser@ubuntu-VirtualBox:~$ javac -version

javac 1.7.0\_75

hduser@ubuntu-VirtualBox:~$ which javac

/usr/bin/javac

hduser@ubuntu-VirtualBox:~$ readlink -f /usr/bin/javac

/usr/lib/jvm/java-7-openjdk-amd64/bin/javac

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

We need to set JAVA\_HOME by modifying hadoop-env.sh file.

hduser@laptop:~$ vi /usr/local/hadoop/etc/hadoop/hadoop-env.sh

export JAVA\_HOME=/usr/lib/jvm/java-7-openjdk-amd64

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

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

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

hduser@laptop:~$ sudo mkdir -p /app/hadoop/tmp

hduser@laptop:~$ sudo chown hduser:hadoop /app/hadoop/tmp

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

hduser@laptop:~$ vi /usr/local/hadoop/etc/hadoop/core-site.xml

<configuration>

<property>

<name>hadoop.tmp.dir</name>

<value>/app/hadoop/tmp</value>

<description>A base for other temporary directories.</description>

</property>

<property>

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

<value>hdfs://localhost:54310</value>

<description>The name of the default file system. A URI whose

scheme and authority determine the FileSystem implementation. The

uri's scheme determines the config property (fs.SCHEME.impl) naming

the FileSystem implementation class. The uri's authority is used to

determine the host, port, etc. for a filesystem.</description>

</property>

</configuration>

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

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

hduser@laptop:~$ cp /usr/local/hadoop/etc/hadoop/mapred-site.xml.template /usr/local/hadoop/etc/hadoop/mapred-site.xml

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

<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. /usr/local/hadoop/etc/hadoop/hdfs-site.xml:**

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

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

hduser@laptop:~$ sudo mkdir -p /usr/local/hadoop\_store/hdfs/namenode

hduser@laptop:~$ sudo mkdir -p /usr/local/hadoop\_store/hdfs/datanode

hduser@laptop:~$ sudo chown -R hduser:hadoop /usr/local/hadoop\_store

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

hduser@laptop:~$ vi /usr/local/hadoop/etc/hadoop/hdfs-site.xml

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

<description>Default block replication.

The actual number of replications can be specified when the file is created.

The default is used if replication is not specified in create time.

</description>

</property>

<property>

<name>dfs.namenode.name.dir</name>

<value>file:/usr/local/hadoop\_store/hdfs/namenode</value>

</property>

<property>

<name>dfs.datanode.data.dir</name>

<value>file:/usr/local/hadoop\_store/hdfs/datanode</value>

</property>

</configuration>

**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:

hduser@laptop:~$ 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:**

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

k@laptop:~$ cd /usr/local/hadoop/sbin

k@laptop:/usr/local/hadoop/sbin$ ls

distribute-exclude.sh start-all.cmd stop-balancer.sh

hadoop-daemon.sh start-all.sh stop-dfs.cmd

hadoop-daemons.sh start-balancer.sh stop-dfs.sh

hdfs-config.cmd start-dfs.cmd stop-secure-dns.sh

hdfs-config.sh start-dfs.sh stop-yarn.cmd

httpfs.sh start-secure-dns.sh stop-yarn.sh

kms.sh start-yarn.cmd yarn-daemon.sh

mr-jobhistory-daemon.sh start-yarn.sh yarn-daemons.sh

refresh-namenodes.sh stop-all.cmd

slaves.sh stop-all.sh

k@laptop:/usr/local/hadoop/sbin$ sudo su hduser

hduser@laptop:/usr/local/hadoop/sbin$ start-all.sh

hduser@laptop:~$ start-all.sh

This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh

15/04/18 16:43:13 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Starting namenodes on [localhost]

localhost: starting namenode, logging to /usr/local/hadoop/logs/hadoop-hduser-namenode-laptop.out

localhost: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hduser-datanode-laptop.out

Starting secondary namenodes [0.0.0.0]

0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-hduser-secondarynamenode-laptop.out

15/04/18 16:43:58 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

starting yarn daemons

starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hduser-resourcemanager-laptop.out

localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hduser-nodemanager-laptop.out

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

hduser@laptop:/usr/local/hadoop/sbin$ jps

9026 NodeManager

7348 NameNode

9766 Jps

8887 ResourceManager

7507 DataNode

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

EXPERIMENT 2

AIM:

Write a necessary HDFC commands to place the data from local file system to HDFS.

|  |  |
| --- | --- |
| 1 | List the available clusters with the cluster list command. |
| 2 | Connect to the Hadoop cluster whose files or directories you want to copy to or from your local filesystem.  cluster target --name *cluster\_name* |
| 3 | Run the command cfg fs --namenode *namenode\_address*.  You must run this command before using fs put or fs get to identify the namenode of the HDFS. |
| 4 | You can copy (upload) a file from the local filesystem to a specific HDFS using the fs putcommand.  fs put --from *source\_path\_and\_file* --to *dest\_path\_and\_file*  The specified file or directory is copied from your local filesystem to the HDFS. |
| 5 | You can copy (download) a file from the a specific HDFS to your local filesystem using the fs get command.  fs get --from *source\_path\_and\_file* to *dest\_path\_and\_file*  The specified file or directory is copied from the HDFS to your local filesystem. |

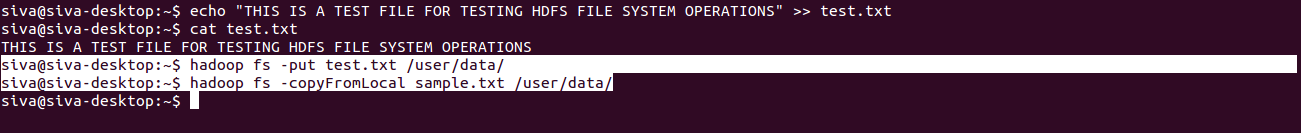
**OUTPUT:**

**Copies files from local file system to HDFS. This is similar to -copyFromLocal command:**

### **1. put:**

**Syntax:**

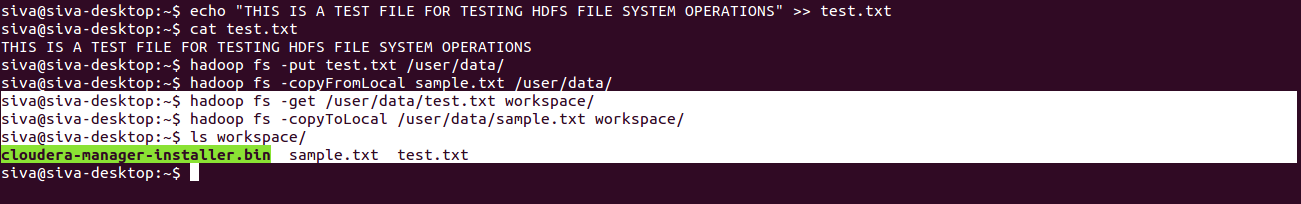
$ hadoop fs -put [-f] [-p] <localsrc>

[](http://hadooptutorial.info/wp-content/uploads/2014/04/FS-put.png)

2. get:

**Copies files from HDFS to local file system. This is similar to -copyToLocal  command:**

$ hadoop fs -get /user/data/sample.txt workspace/

[](http://hadooptutorial.info/wp-content/uploads/2014/04/FS-get.png)

EXPERIMENT 4

**AIM:**

Develop a Mapper class for the word count on text document

**Steps:**

Step 1.  Open Eclipse> File > New > Java Project >( Name it – MRProgramsDemo) > Finish

Step 2.  Right Click > New > Package ( Name it - PackageDemo) > Finish

Step 3. Right Click on Package > New > Class (Name it - WordCount)

Step 4. Add Following Reference Libraries –

Right Click on Project > Build Path> Add External Archivals

* */usr/lib/hadoop-0.20/***hadoop-core.jar**
* *Usr/lib/hadoop-0.20/lib/***Commons-cli-1.2.jar**

Step 5. Type following Program :

**Source Code:**

package PackageDemo;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

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;

import org.apache.hadoop.util.GenericOptionsParser;

public static class MapForWordCount extends Mapper<LongWritable, Text, Text, IntWritable>{

public void map(LongWritable key, Text value, Context con) throws IOException, InterruptedException

{

String line = value.toString();

String[] words=line.split(",");

for(String word: words )

{

Text outputKey = new Text(word.toUpperCase().trim());

IntWritable outputValue = new IntWritable(1);

con.write(outputKey, outputValue);

}

}

}

EXPERIMENT 5

**AIM:**

Develop a Reducer class for the word count on text document

**Steps:**

Step 1.  Open Eclipse> File > New > Java Project >( Name it – MRProgramsDemo) > Finish

Step 2.  Right Click > New > Package ( Name it - PackageDemo) > Finish

Step 3. Right Click on Package > New > Class (Name it - WordCount)

Step 4. Add Following Reference Libraries –

Right Click on Project > Build Path> Add External Archivals

* */usr/lib/hadoop-0.20/***hadoop-core.jar**
* *Usr/lib/hadoop-0.20/lib/***Commons-cli-1.2.jar**

Step 5. Type following Program :

**Source Code:**

package PackageDemo;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

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;

import org.apache.hadoop.util.GenericOptionsParser;

public static class ReduceForWordCount extends Reducer<Text, IntWritable, Text, IntWritable>

{

public void reduce(Text word, Iterable<IntWritable> values, Context con) throws IOException, InterruptedException

{

int sum = 0;

for(IntWritable value : values)

{

sum += value.get();

}

con.write(word, new IntWritable(sum));

}

}

EXPERIMENT 5

**AIM:**

Develop an application to find the word count on text document

**Steps:**

Step 1.  Open Eclipse> File > New > Java Project >( Name it – MRProgramsDemo) > Finish

Step 2.  Right Click > New > Package ( Name it - PackageDemo) > Finish

Step 3. Right Click on Package > New > Class (Name it - WordCount)

Step 4. Add Following Reference Libraries –

Right Click on Project > Build Path> Add External Archivals

* */usr/lib/hadoop-0.20/***hadoop-core.jar**
* *Usr/lib/hadoop-0.20/lib/***Commons-cli-1.2.jar**

Step 5. Type following Program :

**Source Code:**

package PackageDemo;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

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;

import org.apache.hadoop.util.GenericOptionsParser;

public class WordCount {

public static void main(String [] args) throws Exception

{

Configuration c=new Configuration();

String[] files=new GenericOptionsParser(c,args).getRemainingArgs();

Path input=new Path(files[0]);

Path output=new Path(files[1]);

Job j=new Job(c,"wordcount");

j.setJarByClass(WordCount.class);

j.setMapperClass(MapForWordCount.class);

j.setReducerClass(ReduceForWordCount.class);

j.setOutputKeyClass(Text.class);

j.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(j, input);

FileOutputFormat.setOutputPath(j, output);

System.exit(j.waitForCompletion(true)?0:1);

}

public static class MapForWordCount extends Mapper<LongWritable, Text, Text, IntWritable>{

public void map(LongWritable key, Text value, Context con) throws IOException, InterruptedException

{

String line = value.toString();

String[] words=line.split(",");

for(String word: words )

{

Text outputKey = new Text(word.toUpperCase().trim());

IntWritable outputValue = new IntWritable(1);

con.write(outputKey, outputValue);

}

}

}

public static class ReduceForWordCount extends Reducer<Text, IntWritable, Text, IntWritable>

{

public void reduce(Text word, Iterable<IntWritable> values, Context con) throws IOException, InterruptedException

{

int sum = 0;

for(IntWritable value : values)

{

sum += value.get();

}

con.write(word, new IntWritable(sum));

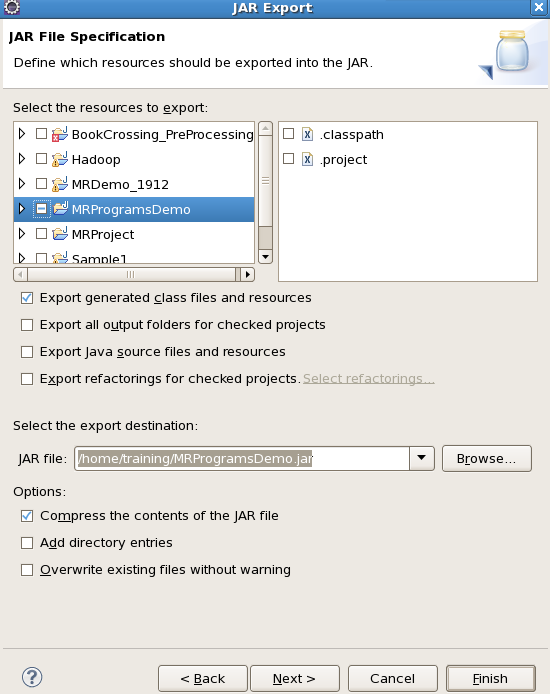
}

}

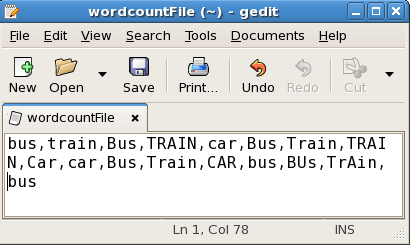
}

**Step 6**. **Make Jar File**

Right Click on Project> Export> Select export destination as **Jar File** > next> Finish



**Step 7: Take a text file and move it in HDFS**



To Move this into Hadoop directly, open the terminal and enter the following commands:

[training@localhost ~]$ hadoop fs -put wordcountFile wordCountFile

**OUTPUT:**

*(hadoop jar jarfilename.jar packageName.ClassName  PathToInputTextFile PathToOutputDirectry)*

[training@localhost ~]$ hadoop jar MRProgramsDemo.jar PackageDemo.WordCount wordCountFile MRDir1

Step 9. Open Result

[training@localhost ~]$ hadoop fs -ls MRDir1

Found 3 items

-rw-r--r-- 1 training supergroup 0 2016-02-23 03:36 /user/training/MRDir1/\_SUCCESS

drwxr-xr-x - training supergroup 0 2016-02-23 03:36 /user/training/MRDir1/\_logs

-rw-r--r-- 1 training supergroup 20 2016-02-23 03:36 /user/training/MRDir1/part-r-00000

[training@localhost ~]$ hadoop fs -cat MRDir1/part-r-00000

BUS 7

CAR 4

TRAIN 6

EXPERIMENT 5

**AIM:**

Develop a Mapper class for the Maximum temperature calculation in the weather dataset

**Steps:**

Step 1.  Open Eclipse> File > New > Java Project >( Name it – MRProgramsDemo) > Finish

Step 2.  Right Click > New > Package ( Name it - PackageDemo) > Finish

Step 3. Right Click on Package > New > Class (Name it - WordCount)

Step 4. Add Following Reference Libraries –

Right Click on Project > Build Path> Add External Archivals

*/usr/lib/hadoop-0.20/*hadoop-core.jar

*Usr/lib/hadoop-0.20/lib/*Commons-cli-1.2.jar

Step 5. Type following Program :

**Source Code:**

**import java.io.IOException;**

**import org.apache.hadoop.io.IntWritable;**

**import org.apache.hadoop.io.LongWritable;**

**import org.apache.hadoop.io.Text;**

**import org.apache.hadoop.mapreduce.Mapper;**

**public class MaxTemperatureMapper**

**extends Mapper<LongWritable, Text, Text, IntWritable> {**

**private static final int MISSING = 9999;**

**@Override**

**public void map(LongWritable key, Text value, Context context)**

**throws IOException, InterruptedException {**

**String line = value.toString();**

**String year = line.substring(15, 19);**

**int airTemperature;**

**if (line.charAt(87) == '+') { // parseInt doesn't like leading plus signs**

**airTemperature = Integer.parseInt(line.substring(88, 92));**

**} else {**

**airTemperature = Integer.parseInt(line.substring(87, 92));**

**}**

**String quality = line.substring(92, 93);**

**if (airTemperature != MISSING && quality.matches("[01459]")) {**

**context.write(new Text(year), new IntWritable(airTemperature));**

**}**

**}**

**}**

EXPERIMENT 5

**AIM:**

Develop a Mapper class for the Maximum temperature calculation in the weather dataset

**Steps:**

Step 1.  Open Eclipse> File > New > Java Project >( Name it – MRProgramsDemo) > Finish

Step 2.  Right Click > New > Package ( Name it - PackageDemo) > Finish

Step 3. Right Click on Package > New > Class (Name it - WordCount)

Step 4. Add Following Reference Libraries –

Right Click on Project > Build Path> Add External Archivals

*/usr/lib/hadoop-0.20/*hadoop-core.jar

*Usr/lib/hadoop-0.20/lib/*Commons-cli-1.2.jar

Step 5. Type following Program :

**Source Code:**

**import java.io.IOException;**

**import org.apache.hadoop.io.IntWritable;**

**import org.apache.hadoop.io.Text;**

**import org.apache.hadoop.mapreduce.Reducer;**

**public class MaxTemperatureReducer**

**extends Reducer<Text, IntWritable, Text, IntWritable> {**

**@Override**

**public void reduce(Text key, Iterable<IntWritable> values,**

**Context context)**

**throws IOException, InterruptedException {**

**int maxValue = Integer.MIN\_VALUE;**

**for (IntWritable value : values) {**

**maxValue = Math.max(maxValue, value.get());**

**}**

**context.write(key, new IntWritable(maxValue));**

**}**

**}**

EXPERIMENT 5

**AIM:**

Develop a Mapper class for the Maximum temperature calculation in the weather dataset

**Steps:**

Step 1.  Open Eclipse> File > New > Java Project >( Name it – MRProgramsDemo) > Finish

Step 2.  Right Click > New > Package ( Name it - PackageDemo) > Finish

Step 3. Right Click on Package > New > Class (Name it - WordCount)

Step 4. Add Following Reference Libraries –

Right Click on Project > Build Path> Add External Archivals

*/usr/lib/hadoop-0.20/*hadoop-core.jar

*Usr/lib/hadoop-0.20/lib/*Commons-cli-1.2.jar

Step 5. Type following Program :

**Source Code:**

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.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class MaxTemperature {

public static void main(String[] args) throws Exception {

if (args.length != 2) {

System.err.println("Usage: MaxTemperature <input path> <output path>");

System.exit(-1);

}

Job job = new Job();

job.setJarByClass(MaxTemperature.class);

job.setJobName("Max temperature");

FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

job.setMapperClass(MaxTemperatureMapper.class);

job.setReducerClass(MaxTemperatureReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}

**OUTPUT:**

**[ STEP0. Make needed java files ]**

- MaxTemperature.java, MaxTemperatureMapper.java, MaxTemperatureReducer.java

   (attached in this document)  
- You can see how these program work from the book.

  ("Analyzing the data with hadoop section in chapter 2")

**[ STEP1. compile classes ]**

[hadoop@server1 ch2]$ **mkdir classes**[hadoop@server1 ch2]$**javac -classpath /home/hadoop/hadoop-1.0.4/hadoop-core-1.0.4.jar -d classes MaxTemperature\*.java**[hadoop@server1 ch2]$ **ls**classes  MaxTemperature.class  MaxTemperatureMapper.class  MaxTemperatureReducer.class  max\_temperature.sh               NewMaxTemperature.java  
data     MaxTemperature.java   MaxTemperatureMapper.java   MaxTemperatureReducer.java   MaxTemperatureWithCombiner.java  sample.txt  
[hadoop@server1 ch2]$ **cd classes**[hadoop@server1 classes]$**ls**MaxTemperature.class  MaxTemperatureMapper.class  MaxTemperatureReducer.class  MaxTemperatureWithCombiner.class

**[ STEP2. make a jar file ]**

[hadoop@server1 classes]$ **jar -cvf MaxTemperature.jar \***added manifest  
adding: MaxTemperature.class(in = 1440) (out= 772)(deflated 46%)  
adding: MaxTemperatureMapper.class(in = 1916) (out= 804)(deflated 58%)  
adding: MaxTemperatureReducer.class(in = 1605) (out= 645)(deflated 59%)  
adding: MaxTemperatureWithCombiner.class(in = 1516) (out= 799)(deflated 47%)  
[hadoop@server1 classes]$**ls**MaxTemperature.class  MaxTemperature.jar  MaxTemperatureMapper.class  MaxTemperatureReducer.class  MaxTemperatureWithCombiner.class  
[hadoop@server1 classes]$ **mv \*.jar ..**[hadoop@server1 classes]$ **ls**MaxTemperature.class  MaxTemperatureMapper.class  MaxTemperatureReducer.class  MaxTemperatureWithCombiner.class  
[hadoop@server1 classes]$**cd ..**[hadoop@server1 ch2]$ **ls**classes  MaxTemperature.class  MaxTemperature.java         MaxTemperatureMapper.java    MaxTemperatureReducer.java  MaxTemperatureWithCombiner.java  sample.txt  
data    **MaxTemperature.jar**    MaxTemperatureMapper.class  MaxTemperatureReducer.class  max\_temperature.sh          NewMaxTemperature.java

**[ STEP3. upload an input file ]**

Now, it's time to upload an input file from your Linux machine to HDFS

which is the distributed file system for Hadoop.

[hadoop@server1 ch2]$ **cat sample.txt**006701199099999**1950**051507004+68750+023550FM-12+038299999V0203301N00671220001CN9999999N9**+00001**+99999999999  
0043011990999991950051512004+68750+023550FM-12+038299999V0203201N00671220001CN9999999N9+00221+99999999999  
0043011990999991950051518004+68750+023550FM-12+038299999V0203201N00261220001CN9999999N9-00111+99999999999  
0043012650999991949032412004+62300+010750FM-12+048599999V0202701N00461220001CN0500001N9+01111+99999999999  
004301265099999**1949**032418004+62300+010750FM-12+048599999V0202701N00461220001CN0500001N9**+00781**+99999999999

**Year                                                                                                        Sensed temperature**

[hadoop@server1 ch2]$ **hadoop fs -mkdir input**[hadoop@server1 ch2]$**hadoop fs -lsr**drwxr-xr-x   - hadoop supergroup          0 2012-08-19 00:08 /user/hadoop/input  
[hadoop@server1 ch2]$ **hadoop fs -put sample.txt input**[hadoop@server1 ch2]$ **hadoop fs -lsr**drwxr-xr-x   - hadoop supergroup          0 2012-08-19 00:12 /user/hadoop/input  
-rw-r--r--   1 hadoop supergroup        529 2012-08-19 00:12 /user/hadoop/input/sample.txt

**[ STEP4. Run MaxTemperature MapReduce program ]**

[hadoop@server1 ch2]$ **hadoop jar MaxTemperature.jar MaxTemperature input output**12/08/19 00:29:24 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.  
12/08/19 00:29:24 INFO util.NativeCodeLoader: Loaded the native-hadoop library  
12/08/19 00:29:24 WARN snappy.LoadSnappy: Snappy native library not loaded  
12/08/19 00:29:24 INFO mapred.FileInputFormat: Total input paths to process : 1  
12/08/19 00:29:24 INFO mapred.JobClient: Running job: job\_201207191718\_0005  
12/08/19 00:29:25 INFO mapred.JobClient:  map 0% reduce 0%  
12/08/19 00:29:40 INFO mapred.JobClient:  map 100% reduce 0%  
12/08/19 00:29:52 INFO mapred.JobClient:  map 100% reduce 100%  
12/08/19 00:29:57 INFO mapred.JobClient: Job complete: job\_201207191718\_0005  
12/08/19 00:29:57 INFO mapred.JobClient: Counters: 30  
12/08/19 00:29:57 INFO mapred.JobClient:   Job Counters   
12/08/19 00:29:57 INFO mapred.JobClient:     Launched reduce tasks=1  
12/08/19 00:29:57 INFO mapred.JobClient:     SLOTS\_MILLIS\_MAPS=19553  
12/08/19 00:29:57 INFO mapred.JobClient:     Total time spent by all reduces waiting after reserving slots (ms)=0  
12/08/19 00:29:57 INFO mapred.JobClient:     Total time spent by all maps waiting after reserving slots (ms)=0  
12/08/19 00:29:57 INFO mapred.JobClient:     Launched map tasks=2  
12/08/19 00:29:57 INFO mapred.JobClient:     Data-local map tasks=2  
12/08/19 00:29:57 INFO mapred.JobClient:     SLOTS\_MILLIS\_REDUCES=10471  
12/08/19 00:29:57 INFO mapred.JobClient:   File Input Format Counters   
12/08/19 00:29:57 INFO mapred.JobClient:     Bytes Read=795  
12/08/19 00:29:57 INFO mapred.JobClient:   File Output Format Counters   
12/08/19 00:29:57 INFO mapred.JobClient:     Bytes Written=17  
12/08/19 00:29:57 INFO mapred.JobClient:   FileSystemCounters  
12/08/19 00:29:57 INFO mapred.JobClient:     FILE\_BYTES\_READ=61  
12/08/19 00:29:57 INFO mapred.JobClient:     HDFS\_BYTES\_READ=1001  
12/08/19 00:29:57 INFO mapred.JobClient:     FILE\_BYTES\_WRITTEN=63781  
12/08/19 00:29:57 INFO mapred.JobClient:     HDFS\_BYTES\_WRITTEN=17  
12/08/19 00:29:57 INFO mapred.JobClient:   Map-Reduce Framework  
12/08/19 00:29:57 INFO mapred.JobClient:     Map output materialized bytes=67  
12/08/19 00:29:57 INFO mapred.JobClient:     Map input records=5  
12/08/19 00:29:57 INFO mapred.JobClient:     Reduce shuffle bytes=67  
12/08/19 00:29:57 INFO mapred.JobClient:     Spilled Records=10  
12/08/19 00:29:57 INFO mapred.JobClient:     Map output bytes=45  
12/08/19 00:29:57 INFO mapred.JobClient:     Total committed heap usage (bytes)=408944640  
12/08/19 00:29:57 INFO mapred.JobClient:     CPU time spent (ms)=2060  
12/08/19 00:29:57 INFO mapred.JobClient:     Map input bytes=529  
12/08/19 00:29:57 INFO mapred.JobClient:     SPLIT\_RAW\_BYTES=206  
12/08/19 00:29:57 INFO mapred.JobClient:     Combine input records=0  
12/08/19 00:29:57 INFO mapred.JobClient:     Reduce input records=5  
12/08/19 00:29:57 INFO mapred.JobClient:     Reduce input groups=2  
12/08/19 00:29:57 INFO mapred.JobClient:     Combine output records=0  
12/08/19 00:29:57 INFO mapred.JobClient:     Physical memory (bytes) snapshot=397139968  
12/08/19 00:29:57 INFO mapred.JobClient:     Reduce output records=2  
12/08/19 00:29:57 INFO mapred.JobClient:     Virtual memory (bytes) snapshot=1174450176  
12/08/19 00:29:57 INFO mapred.JobClient:     Map output records=5

**[ STEP5. Check the running result ]**

[hadoop@server1 ch2]$ **hadoop fs -lsr**drwxr-xr-x   - hadoop supergroup          0 2012-08-19 00:12 /user/hadoop/input  
-rw-r--r--   1 hadoop supergroup        529 2012-08-19 00:12 /user/hadoop/input/sample.txt  
drwxr-xr-x   - hadoop supergroup          0 2012-08-19 00:29 /user/hadoop/output  
-rw-r--r--   1 hadoop supergroup          0 2012-08-19 00:29 /user/hadoop/output/\_SUCCESS  
drwxr-xr-x   - hadoop supergroup          0 2012-08-19 00:29 /user/hadoop/output/\_logs  
drwxr-xr-x   - hadoop supergroup          0 2012-08-19 00:29 /user/hadoop/output/\_logs/history  
-rw-r--r--   1 hadoop supergroup      16521 2012-08-19 00:29 /user/hadoop/output/\_logs/history/job\_201207191718\_0005\_1345354164616\_hadoop\_Max+temperature  
-rw-r--r--   1 hadoop supergroup      20000 2012-08-19 00:29 /user/hadoop/output/\_logs/history/job\_201207191718\_0005\_conf.xml  
-rw-r--r--   1 hadoop supergroup         17 2012-08-19 00:29 /user/hadoop/output/part-00000

[hadoop@server1 ch2]$ **hadoop fs -cat output/part-00000**1949 111  
1950 22