CS426: Hadoop Experiment Report

Zuoming Zhang
5120309626
237632270@qq.com
Department of Computer Science and Engineering
School of Electronic Information and Electrical Engineering
Shanghai Jiao Tong University

Weichen Li
5120309662
eizo.lee@sjtu.edu.cn
Department of Computer Science and Engineering
School of Electronic Information and Electrical Engineering
Shanghai Jiao Tong University

Peiyuan Liu
5120309394
ms_green@sjtu.edu.cn
Department of Computer Science and Engineering
School of Electronic Information and Electrical Engineering
Shanghai Jiao Tong University

Abstract

This report describes steps to construct Hadoop both in **Single Mode** and in **Cluster Mode**. Our report & hadoop installation guide is suitable for both real computers and virtual machines, and the operating system is based on **Ubuntu** (both Ubuntu Desktop and Ubuntu Server are OK, we highly recommand you to use Ubuntu Desktop for Master and Server for Slaves). You can follow our report step by step and build your own hadoop in single/cluster mode. We also run *WordCount* program, in both **Single Mode** and **Cluster Mode**, to test the performance of our hadoop system. Then we further discussed our understanding of hadoop and some interesting findings in the conclusion part.

CONTENTS

I. Division of Labor

Our group is composed of three members: Zuoming Zhang, Weichen Li, Peiyuan Liu, and Zuoming Zhang is the team leader.

In the first project, all the three of us members have implemented the single mode separatedly. To construct the cluster mode and implement this report & hadoop installation guide, we divide the labor in the following pattern according to each one's ability and experience:

When first trying to establish the cluster mode, Weichen Li gathered relevant information and guidance from the Internet. Then his computer worked as the Master node to control the procedure of construction. Most of the configuration and settings in the cluster mode are implemented by him.

After the first success, Zuoming Zhang reimplemented the whole process in standard steps without mistakes, get the screenshots and write a rough Chinese version draft. He is also resposible for solving technique problems based on his hadoop experience in pseudo-cluster mode.

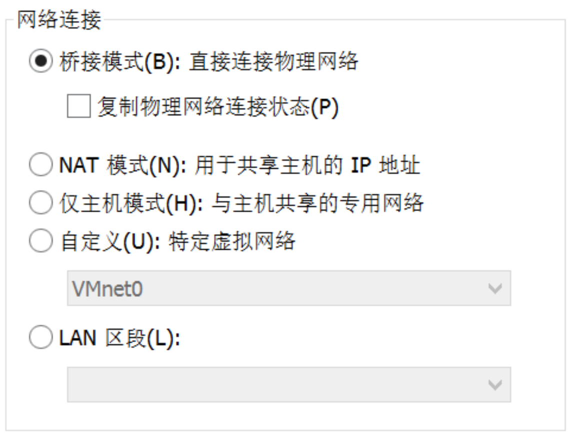
Then Peiyuan Liu further implemented the project report in English in detail based on the Chinese draft, and she is reposible for setting up the type of the whole report.

II. Experiment Environment

A. Network Environment

In our experiment, we use **real Ethernet** connecting three different laptop computers to build our Hadoop cluster. Each operating system is running on a virtual machine on separate labtops.

We use **VMWARE** to run our operating systems, before we try to connect to the real Ethernet, we must modify the **Network Adapter** of the virtual machine first: Changing **NAT Mode** to **Bridge Mode** (Fig.1).



网络连接

[✓ 桥接模式(B): 直接连接物理网络]

☐ 复制物理网络连接状态(P)

NAT 模式(N): 用于共享主机的 IP 地址

仅主机模式(H): 与主机共享的专用网络

自定义(U): 特定虚拟网络

VMnet0

LAN 区段(L):

Fig. 1. Modify Network Adapter

B. Operating System Environment

The operating systems and there kernel versions are listed in Table.I:

TABLE I
Operating System Environment

NodeName	Operating System	Kernel Version
Master	Ubuntu Desktop 14.04 64bit	3.13.0-39
Slave1	Ubuntu Kylin 14.10 64bit	3.16.0-23
Slave2	Ubuntu Kylin 14.10 64bit	3.16.0-23

III. Single Mode Construction

A. Add the Hadoop user

First we add the **hadoop** user group:

```
sudo addgroup hadoop
```

Then we add the **hadoop** user:

```
sudo adduser -ingroup hadoop hadoop
```

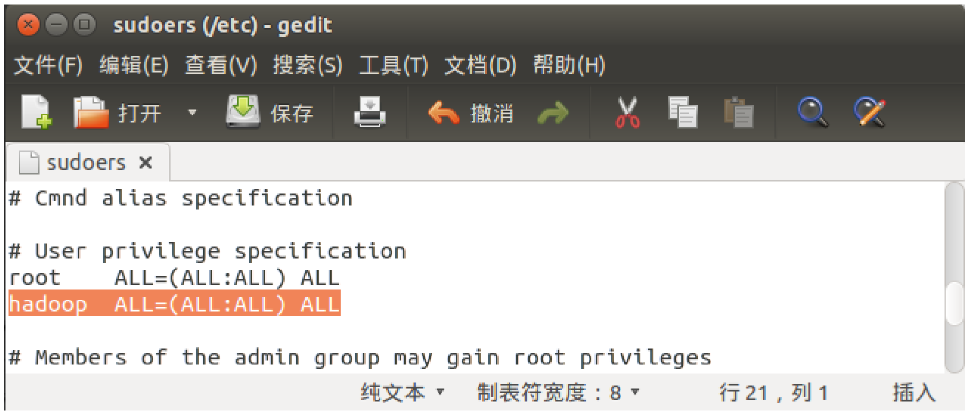
When you type in the "Enter" key, it will ask you to type in the new UNIX password and your personal information. The password is compulsory while you can leave personal information blank.

B. Add authority to the Hadoop user

```
sudo gedit /etc/sudoers
```

This command give user **hadoop** the same authority as **root**.

In gedit, add such a line as in Figure 2.



```
# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
hadoop  ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
```

Fig. 2. Add authority to user hadoop

C. Login Ubuntu using the new Hadoop user

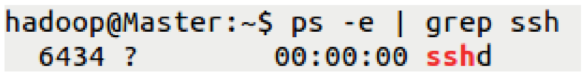
D. Installation of ssh

```
sudo apt-get install openssh-server
```

After the installation, check whether the service is established.

```
ps -e | grep ssh
```

A correct response should look like the codes in Figure 3.



```
hadoop@Master:~$ ps -e | grep ssh
 6434 ?        00:00:00 sshd
```

Fig. 3. SSH Service Establishment Check

Then set the password-free login, we will get the public key and the private key.

```
ssh-keygen -t rsa -P ""
```

Corresponding output should looks like Figure 4.

At the same time, there will be two new files in directory ∼ /.ssh, as you can see in Figure 5.

Add the public key to a new file *authorized_keys*:

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

Then we will get files in Figure 6.

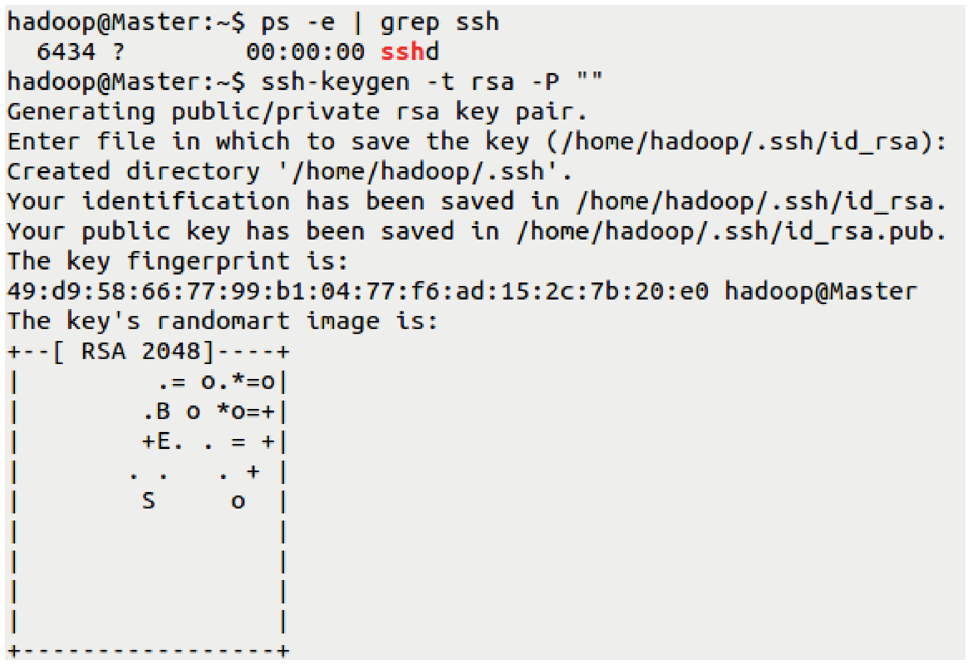
Next, login ssh:

```
ssh localhost
```

Result is in Figure 7.

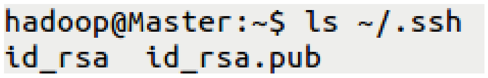
Then exit ssh.

```
exit
```



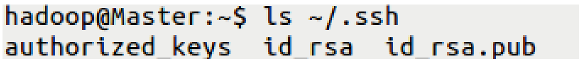
```
hadoop@Master:~$ ps -e | grep ssh
  6434 ?        00:00:00 sshd
hadoop@Master:~$ ssh-keygen -t rsa -P ""
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hadoop/.ssh/id_rsa):
Created directory '/home/hadoop/.ssh'.
Your identification has been saved in /home/hadoop/.ssh/id_rsa.
Your public key has been saved in /home/hadoop/.ssh/id_rsa.pub.
The key fingerprint is:
49:d9:58:66:77:99:b1:04:77:f6:ad:15:2c:7b:20:e0 hadoop@Master
The key's randomart image is:
+--[ RSA 2048]----+
|         .= o.*=o|
|        .B o *o=+|
|        +E. . = +|
|       . .   . + |
|        S     o  |
|                 |
|                 |
|                 |
|                 |
+-----------------+
```

Fig. 4. Generating SSH Keys



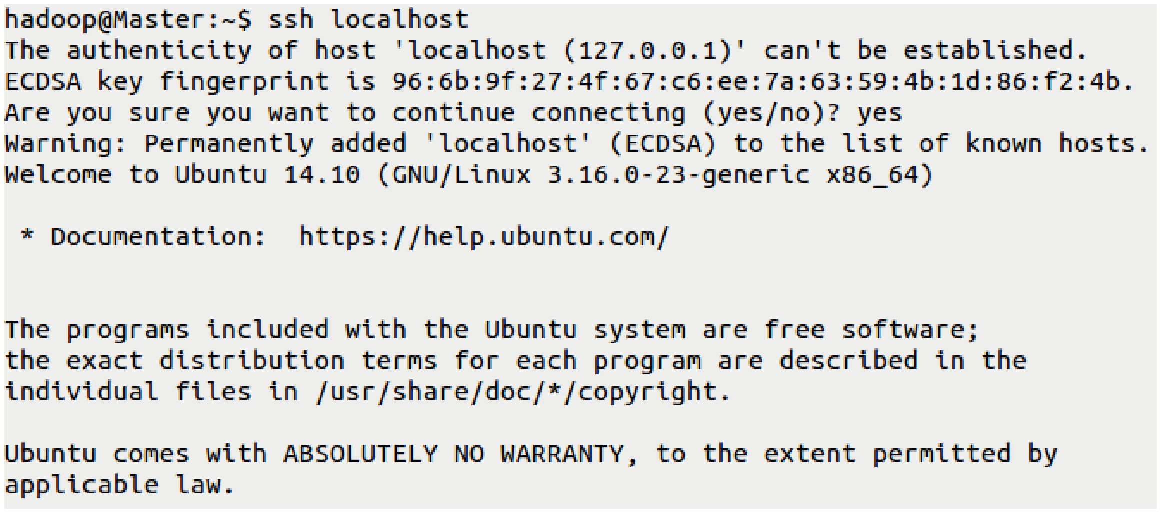
```
hadoop@Master:~$ ls ~/.ssh
id_rsa  id_rsa.pub
```

Fig. 5. SSH files after generating keys



```
hadoop@Master:~$ ls ~/.ssh
authorized_keys  id_rsa  id_rsa.pub
```

Fig. 6. SSH files after adding *authorized_keys*



```
hadoop@Master:~$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is 96:6b:9f:27:4f:67:c6:ee:7a:63:59:4b:1d:86:f2:4b.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 14.10 (GNU/Linux 3.16.0-23-generic x86_64)

 * Documentation:  https://help.ubuntu.com/


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.
```

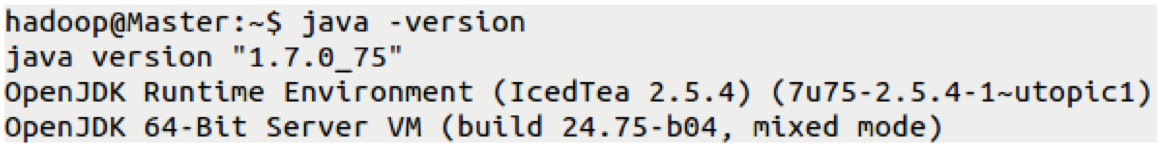
Fig. 7. Result of ssh to localhost

E. Installation of Java environment

```
sudo apt-get install openjdk-7-jdk
```

Check the java environment installation(Fig.8):

```
java -version
```



```
hadoop@Master:~$ java -version
java version "1.7.0_75"
OpenJDK Runtime Environment (IcedTea 2.5.4) (7u75-2.5.4-1~utopic1)
OpenJDK 64-Bit Server VM (build 24.75-b04, mixed mode)
```

Fig. 8. Check the installation of java

F. Installation of Hadoop 2.6.0

Download Hadoop 2.6.0 from the website *http://mirror.bit.edu.cn/apache/hadoop/common/hadoop-2.6.0/hadoop-2.6.0.tar.gz*
Unzip and move the hadoop file:

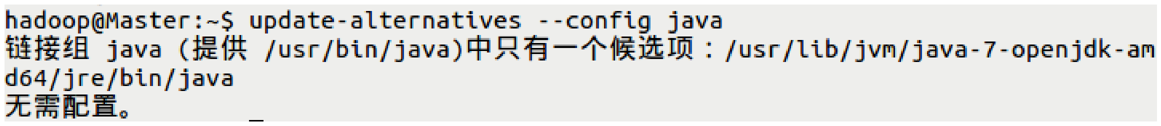
```
sudo tar xzf hadoop-2.6.0.tar.gz
sudo mv hadoop-2.6.0 /usr/local/hadoop
```

Change the authority of the hadoop folder:

```
sudo chmod 777 /usr/local/hadoop
```

Check the location of java(see in Figure 9):

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



```
hadoop@Master:~$ update-alternatives --config java
链接组 java (提供 /usr/bin/java)中只有一个候选项：/usr/lib/jvm/java-7-openjdk-am
d64/jre/bin/java
无需配置。
```

Fig. 9. Check the location of java

Here, the java location is: */usr/lib/jvm/java-7-openjdk-amd64*.
Then set ∼/.bashrc:

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

Add these content to the end of the file:

```
#HADOOP VARIABLES START
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64 #Same to the java location
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
```

Enable the environment variable:

```
source ~/.bashrc
```

Modify *hadoop-env.sh*:

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

Find **JAVA_HOME**, modify it as:

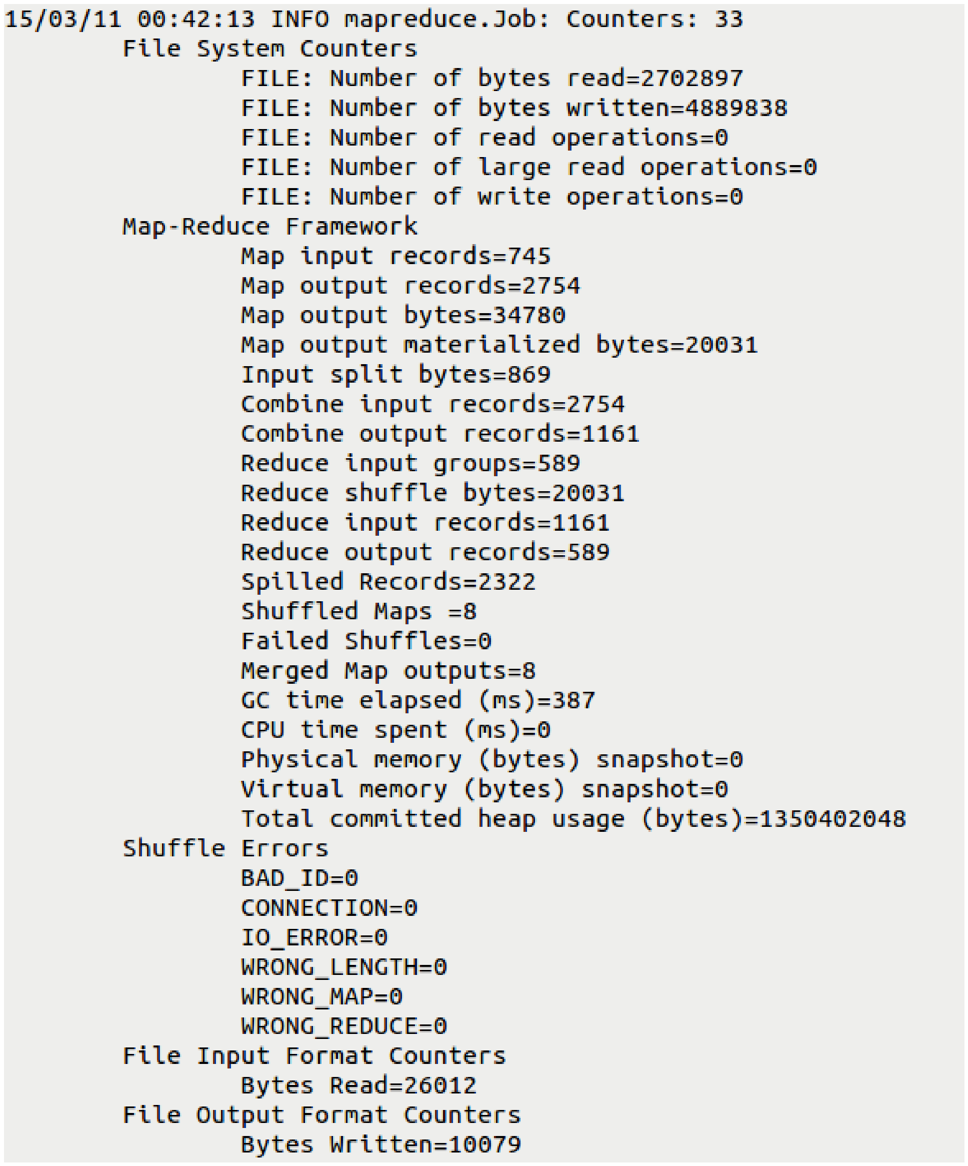
```
# The java implementation to use.
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
```

Now, we can run hadoop in **Single Mode**.

IV. WordCount Test in Single Mode

```
cd /usr/local/hadoop/
mkdir input
cp etc/hadoop/*.xml input
bin/hadoop jar share/hadoop/mapreduce/sources/hadoop-mapreduce-examples-2.6.0-sources.jar org.apache.hadoop.
examples.WordCount input output
```

The execution result is in Figure 10



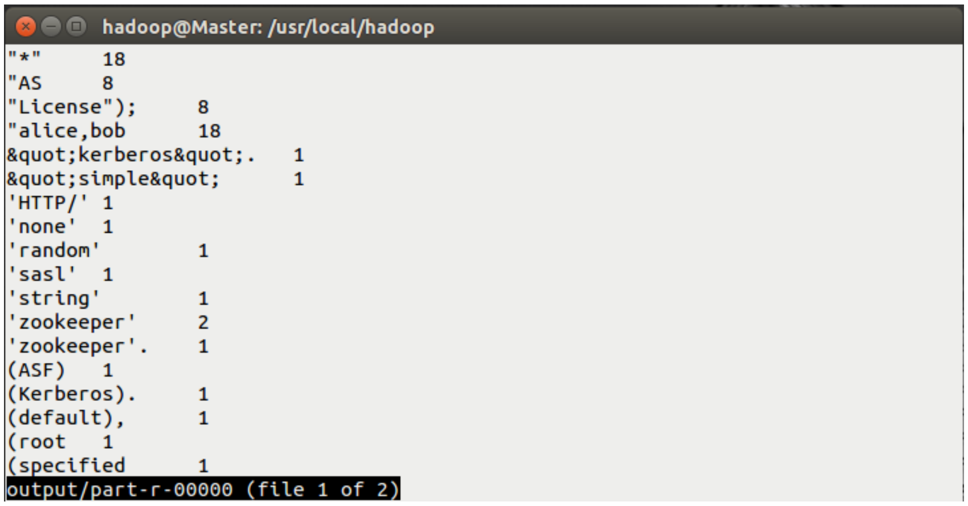
```
15/03/11 00:42:13 INFO mapreduce.Job: Counters: 33
        File System Counters
                FILE: Number of bytes read=2702897
                FILE: Number of bytes written=4889838
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
        Map-Reduce Framework
                Map input records=745
                Map output records=2754
                Map output bytes=34780
                Map output materialized bytes=20031
                Input split bytes=869
                Combine input records=2754
                Combine output records=1161
                Reduce input groups=589
                Reduce shuffle bytes=20031
                Reduce input records=1161
                Reduce output records=589
                Spilled Records=2322
                Shuffled Maps =8
                Failed Shuffles=0
                Merged Map outputs=8
                GC time elapsed (ms)=387
                CPU time spent (ms)=0
                Physical memory (bytes) snapshot=0
                Virtual memory (bytes) snapshot=0
                Total committed heap usage (bytes)=1350402048
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=26012
        File Output Format Counters
                Bytes Written=10079
```

Fig. 10. WordCount result

Check the output(see in Figure 11), press q to quit:

```
less output/*_
```

We have finished building **Single Mode**, and we will start to build **Cluster Mode** of hadoop.



```
"*"     18
"AS     8
"License");     8
"alice,bob      18
&quot;kerberos&quot;.    1
&quot;simple&quot;      1
'HTTP/' 1
'none'  1
'random'        1
'sasl'  1
'string'        1
'zookeeper'     2
'zookeeper'.    1
(ASF)   1
(Kerberos).     1
(default),      1
(root   1
(specified      1
output/part-r-00000 (file 1 of 2)
```

Fig. 11. WordCount output

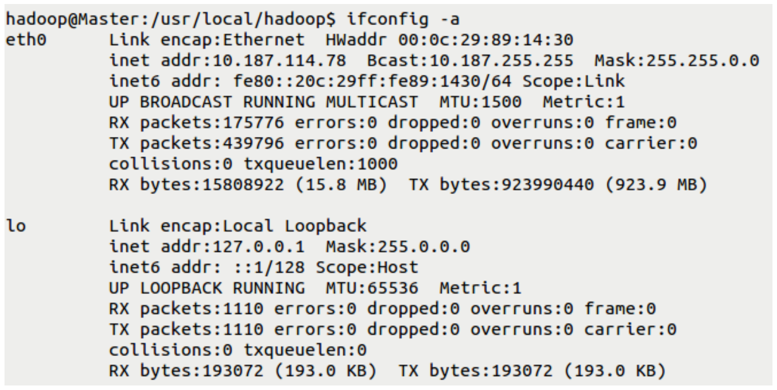
V. Cluster Mode Construction

A. Preparations for constructing cluster mode.

According to the configuration of Hadoop on a single machine, install Hadoop on the other computers in the cluster, including SSH and java environment.

We can check the IP address using(see in Figure 12, the IP address is the first *inet address*. Thus, the IP address of the machine is 10.187.114.78):

```
ifconfig -a
```



```
hadoop@Master:/usr/local/hadoop$ ifconfig -a
eth0      Link encap:Ethernet  HWaddr 00:0c:29:89:14:30
          inet addr:10.187.114.78  Bcast:10.187.255.255  Mask:255.255.0.0
          inet6 addr: fe80::20c:29ff:fe89:1430/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:175776 errors:0 dropped:0 overruns:0 frame:0
          TX packets:439796 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:15808922 (15.8 MB)  TX bytes:923990440 (923.9 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1110 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1110 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:193072 (193.0 KB)  TX bytes:193072 (193.0 KB)
```

Fig. 12. Check IP Address

The addresses of the three machines in the cluster are:

```
10.187.114.78  Master
10.187.114.76  Slave1
10.187.244.157 Slave2
```

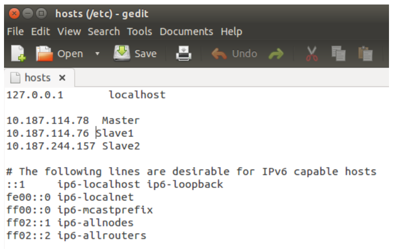
Names of your computers may not look like above, then how to modify the names of the machines? Execute the following command on each machine:

```
sudo gedit /etc/hostname
```

Modify the name as *Master*, *Slave1* and *Slave2*, respectively. (One name for one machine)

Then modify the *hosts* file. Correct result is in Figure 13:

```
sudo vim /etc/hosts
```

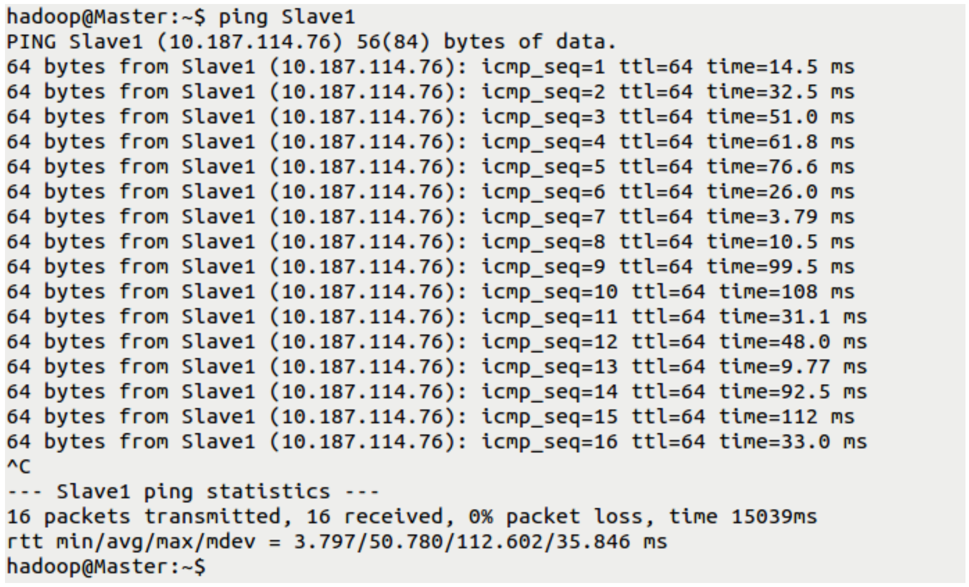


```
127.0.0.1       localhost

10.187.114.78  Master
10.187.114.76 Slave1
10.187.244.157 Slave2

# 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
```

Fig. 13. Modify the *hosts* file

All the machines should modify the *hosts* file, to ensure that all the *hosts* files are the same.

After the modification, we can use *ping* command to connect to the other two nodes, if the results are similar to Figure 14, then the configuration is done.



```
hadoop@Master:~$ ping Slave1
PING Slave1 (10.187.114.76) 56(84) bytes of data.
64 bytes from Slave1 (10.187.114.76): icmp_seq=1 ttl=64 time=14.5 ms
64 bytes from Slave1 (10.187.114.76): icmp_seq=2 ttl=64 time=32.5 ms
64 bytes from Slave1 (10.187.114.76): icmp_seq=3 ttl=64 time=51.0 ms
64 bytes from Slave1 (10.187.114.76): icmp_seq=4 ttl=64 time=61.8 ms
64 bytes from Slave1 (10.187.114.76): icmp_seq=5 ttl=64 time=76.6 ms
64 bytes from Slave1 (10.187.114.76): icmp_seq=6 ttl=64 time=26.0 ms
64 bytes from Slave1 (10.187.114.76): icmp_seq=7 ttl=64 time=3.79 ms
64 bytes from Slave1 (10.187.114.76): icmp_seq=8 ttl=64 time=10.5 ms
64 bytes from Slave1 (10.187.114.76): icmp_seq=9 ttl=64 time=99.5 ms
64 bytes from Slave1 (10.187.114.76): icmp_seq=10 ttl=64 time=108 ms
64 bytes from Slave1 (10.187.114.76): icmp_seq=11 ttl=64 time=31.1 ms
64 bytes from Slave1 (10.187.114.76): icmp_seq=12 ttl=64 time=48.0 ms
64 bytes from Slave1 (10.187.114.76): icmp_seq=13 ttl=64 time=9.77 ms
64 bytes from Slave1 (10.187.114.76): icmp_seq=14 ttl=64 time=92.5 ms
64 bytes from Slave1 (10.187.114.76): icmp_seq=15 ttl=64 time=112 ms
64 bytes from Slave1 (10.187.114.76): icmp_seq=16 ttl=64 time=33.0 ms
^C
--- Slave1 ping statistics ---
16 packets transmitted, 16 received, 0% packet loss, time 15039ms
rtt min/avg/max/mdev = 3.797/50.780/112.602/35.846 ms
hadoop@Master:~$
```

Fig. 14. Ping command test

B. *SSH password-free login on slaves*

Enable Master node to login on a Slave node without password.

Generate the public key of Master:

```
cd ~/.ssh
ssh-keygen -t rsa
```

Master should be able to *ssh* itself without password, execute:

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

Use *ssh* to check the result.(result is in Figure 15):

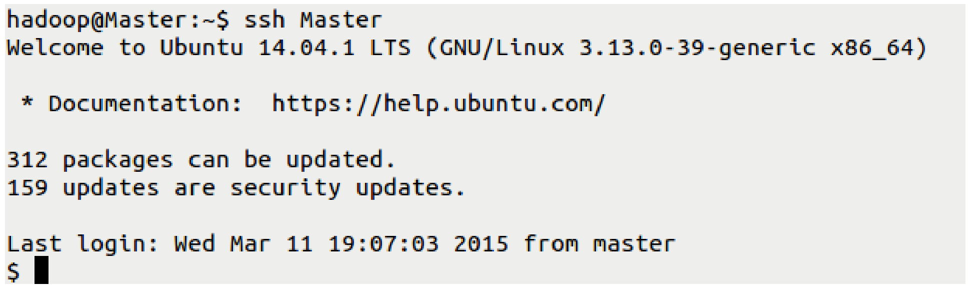
```
ssh Master
```

Transmit the public key to the two slaves, and password is required in the first time connection:

```
scp /home/hadoop/.ssh/id_rsa.pub hadoop@Slave1: /home/hadoop/
scp /home/hadoop/.ssh/id_rsa.pub hadoop@Slave2: /home/hadoop/
```

On Slave1 and Slave2, save the public key to the target location:

```
cat ~/id_rsa.pub >> ~/.ssh/authorized_keys
```



```
hadoop@Master:~$ ssh Master
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-39-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

312 packages can be updated.
159 updates are security updates.

Last login: Wed Mar 11 19:07:03 2015 from master
$ 
```

Fig. 15. SSH command test

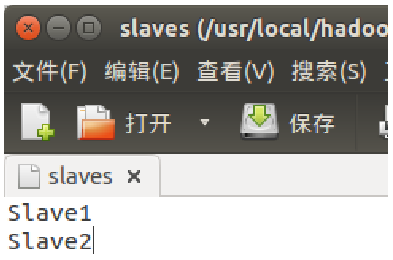
Then we can use Master to login on Slave1 and Slave2 without password:

```
ssh Slave1
ssh Slave2
```

C. *Cluster environment configuration*

On Master, modify *slaves* file as Figure 16:

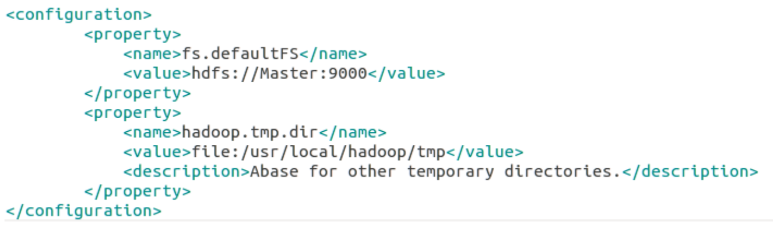
```
cd /usr/local/hadoop/etc/hadoop/
sudo gedit slaves
```



```
Slave1
Slave2
```

Fig. 16. *slaves* file modification

On Master, modify *core-site.xml* file as Figure 17:



```
<configuration>
        <property>
            <name>fs.defaultFS</name>
            <value>hdfs://Master:9000</value>
        </property>
        <property>
            <name>hadoop.tmp.dir</name>
            <value>file:/usr/local/hadoop/tmp</value>
            <description>Abase for other temporary directories.</description>
        </property>
</configuration>
```

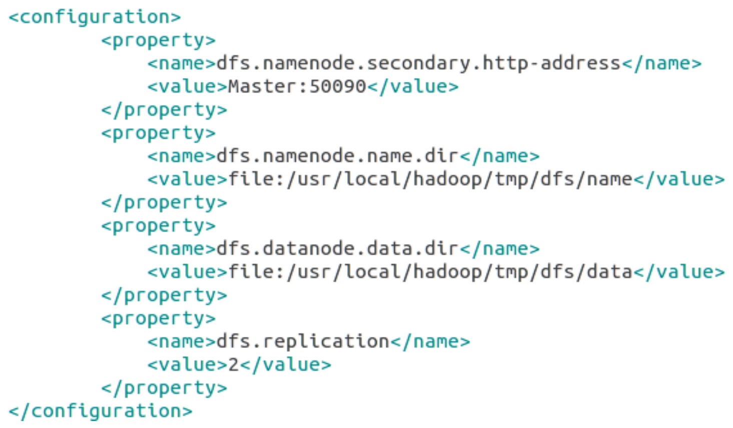
Fig. 17. *core-site.xml* file modification

On Master, modify *hdfs-site.xml* file as Figure 18:
The value of *dfs.replication* is the number of slaves.
On Master, copy a template first:

```
cp mapred-site.xml.template mapred-site.xml
```

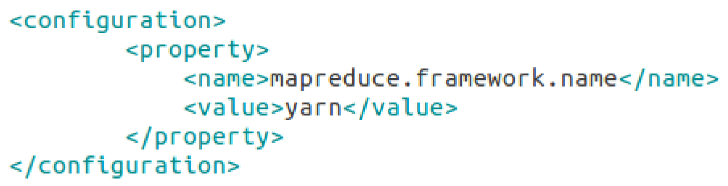
Then on Master modify *hdfs-site.xml* file as Figure 19:
Finally, on Master, modify *yarn-site.xml* file as Figure 20:
After the configuration of these files, compress the Hadoop file, then send it to slaves.

```
cd /usr/local/
sudo tar -zcf ./hadoop.tar.gz ./hadoop
scp ./hadoop.tar.gz Slave1:/home/hadoop/
scp ./hadoop.tar.gz Slave2:/home/hadoop/
```



```
<configuration>
        <property>
            <name>dfs.namenode.secondary.http-address</name>
            <value>Master:50090</value>
        </property>
        <property>
            <name>dfs.namenode.name.dir</name>
            <value>file:/usr/local/hadoop/tmp/dfs/name</value>
        </property>
        <property>
            <name>dfs.datanode.data.dir</name>
            <value>file:/usr/local/hadoop/tmp/dfs/data</value>
        </property>
        <property>
            <name>dfs.replication</name>
            <value>2</value>
        </property>
</configuration>
```

Fig. 18. *hdfs-site.xml* file modification



```
<configuration>
        <property>
            <name>mapreduce.framework.name</name>
            <value>yarn</value>
        </property>
</configuration>
```

Fig. 19. *mapred-site.xml* file modification

On Slave1 and Slave2, execute:

```
sudo tar -zxf ~/hadoop.tar.gz -C /usr/local/
sudo chown -R hadoop:hadoop /usr/local/hadoop
```

Now, we can set on to run *WordCount* in cluster mode.

VI. WordCount Test in Cluster Mode

From now on, all the steps are implemented only on Master

A. Format HDFS

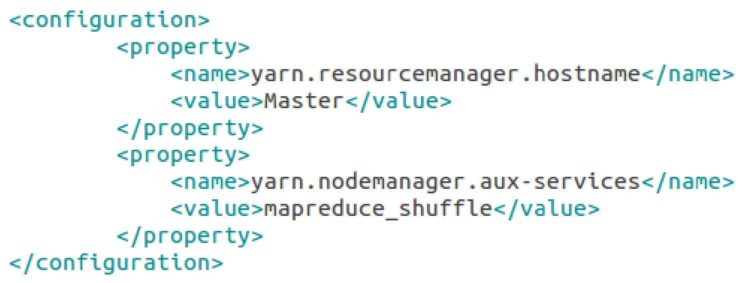
At the first time of Hadoop operation, we need to format:

```
cd /usr/local/hadoop/
bin/hdfs namenode -format
```

B. Start HDFS

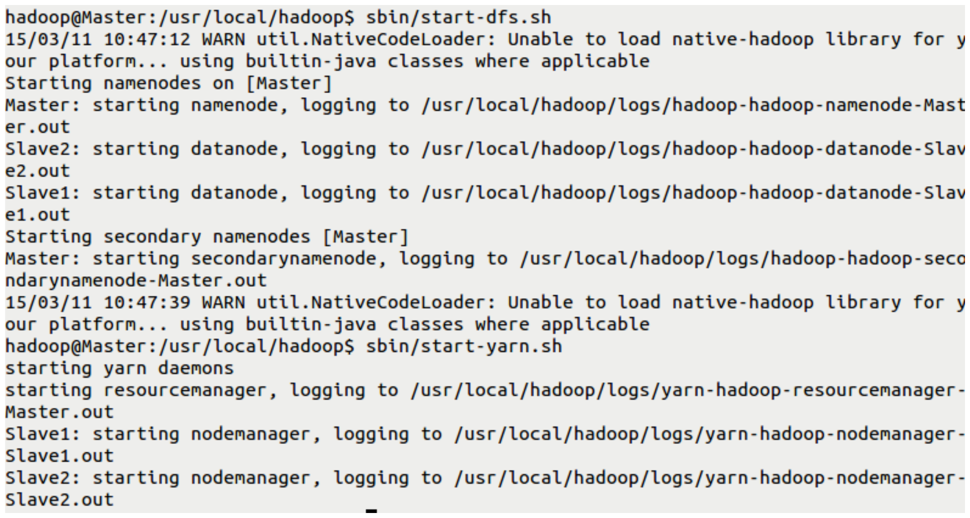
Then we start hadoop, like in Figure 21:

```
sbin/start-dfs.sh
sbin/start-yarn.sh
```



```
<configuration>
        <property>
            <name>yarn.resourcemanager.hostname</name>
            <value>Master</value>
        </property>
        <property>
            <name>yarn.nodemanager.aux-services</name>
            <value>mapreduce_shuffle</value>
        </property>
</configuration>
```

Fig. 20. *yarn-site.xml* file modification



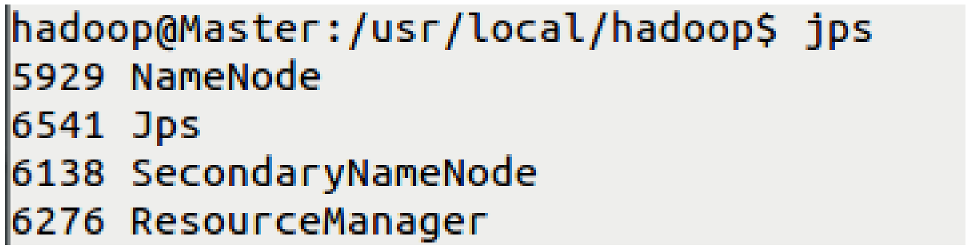
```
hadoop@Master:/usr/local/hadoop$ sbin/start-dfs.sh
15/03/11 10:47:12 WARN util.NativeCodeLoader: Unable to load native-hadoop library for y
our platform... using builtin-java classes where applicable
Starting namenodes on [Master]
Master: starting namenode, logging to /usr/local/hadoop/logs/hadoop-hadoop-namenode-Mast
er.out
Slave2: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hadoop-datanode-Slav
e2.out
Slave1: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hadoop-datanode-Slav
e1.out
Starting secondary namenodes [Master]
Master: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-hadoop-seco
ndarynamenode-Master.out
15/03/11 10:47:39 WARN util.NativeCodeLoader: Unable to load native-hadoop library for y
our platform... using builtin-java classes where applicable
hadoop@Master:/usr/local/hadoop$ sbin/start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hadoop-resourcemanager-
Master.out
Slave1: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hadoop-nodemanager-
Slave1.out
Slave2: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hadoop-nodemanager-
Slave2.out
```

Fig. 21. Start hadoop in cluster mode

To check processes on each node, we can use(see in Figure 22):

```
jps
```

On *Master*, we should see **NameNode**, **SecondrryNameNode** and **ResourceManager**. On two *Slaves*, we should see **DataNode** and **NodeManager**.



```
hadoop@Master:/usr/local/hadoop$ jps
5929 NameNode
6541 Jps
6138 SecondaryNameNode
6276 ResourceManager
```

Fig. 22. Using jps to check running hadoop processes

To check the status of nodes, we can use(see in Figure 23):

```
bin/hdfs dfsadmin -report
```

We can also check online: *http://master:50070/*(see in Figure 24)

C. Execute WordCount

Build a new folder in hdfs, then copy everything in etc/hadoop into *input*:

```
bin/hdfs dfs -mkdir /user
bin/hdfs dfs -mkdir /user/hadoop
bin/hdfs dfs -put etc/hadoop input
```

Then Run WordCount:

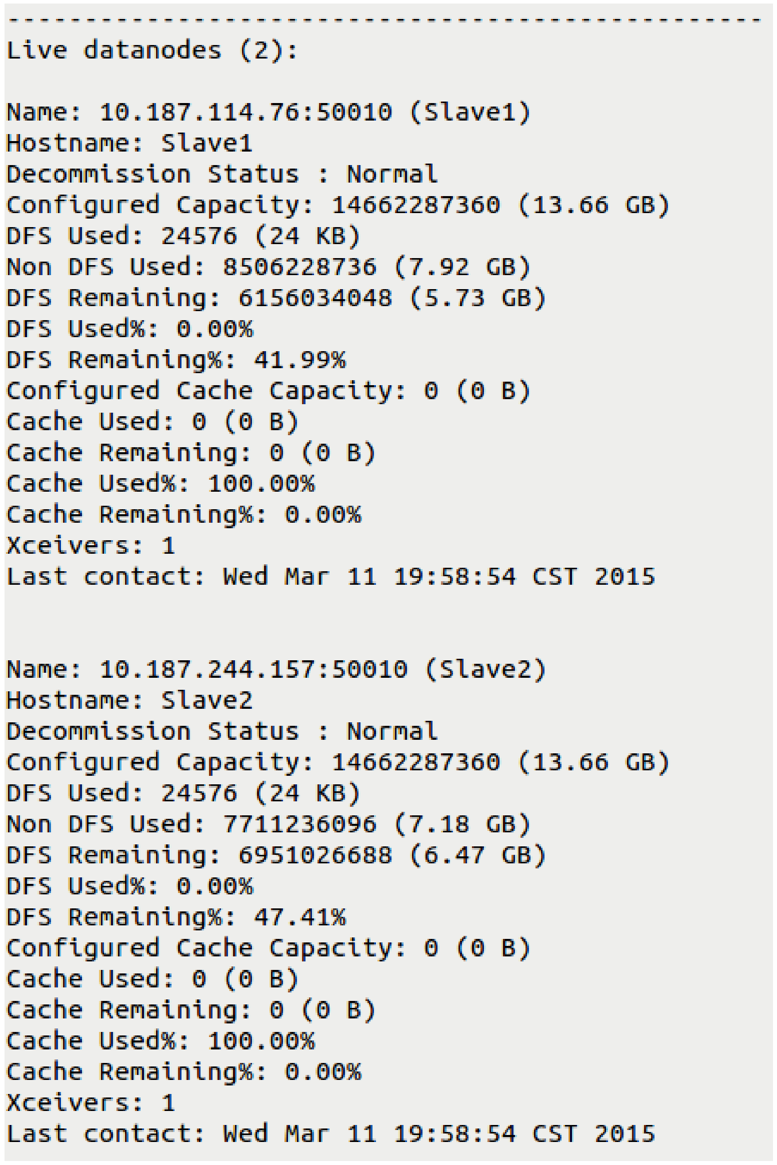
```
bin/hadoop jar share/hadoop/mapreduce/sources/hadoop-mapreduce-examples-2.6.0-sources.jar org.apache.hadoop.
examples.WordCount input output
```

We can both check the process using command line and using a webpage: *http://master:8088/cluster* (Figure 25, Figure 26, Figure 27 indicates the beginning , middle and finishing of the process correspondingly). The process result observed using command line is shown in Figure.28. After the execution of *WordCount*, we may observe in Figure 29 that the used storaged has changed.

We can check the output result(see in Figure 30):

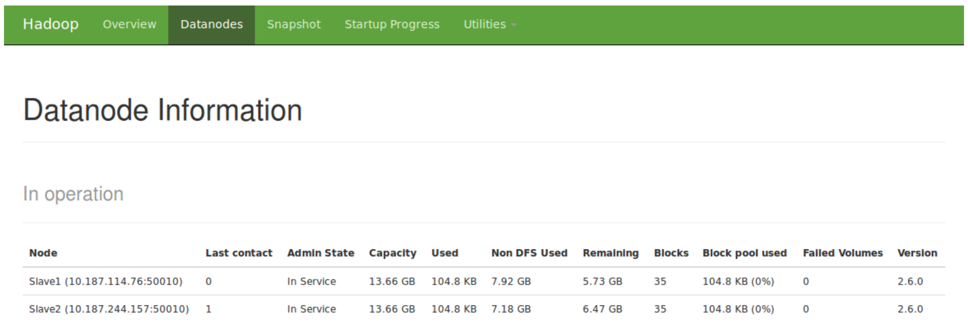
```
bin/hdfs dfs -cat output/* | less
```

Press q to quit.



```
-------------------------------------------------
Live datanodes (2):

Name: 10.187.114.76:50010 (Slave1)
Hostname: Slave1
Decommission Status : Normal
Configured Capacity: 14662287360 (13.66 GB)
DFS Used: 24576 (24 KB)
Non DFS Used: 8506228736 (7.92 GB)
DFS Remaining: 6156034048 (5.73 GB)
DFS Used%: 0.00%
DFS Remaining%: 41.99%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Wed Mar 11 19:58:54 CST 2015


Name: 10.187.244.157:50010 (Slave2)
Hostname: Slave2
Decommission Status : Normal
Configured Capacity: 14662287360 (13.66 GB)
DFS Used: 24576 (24 KB)
Non DFS Used: 7711236096 (7.18 GB)
DFS Remaining: 6951026688 (6.47 GB)
DFS Used%: 0.00%
DFS Remaining%: 47.41%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Wed Mar 11 19:58:54 CST 2015
```

Fig. 23. Check the status of Datanodes

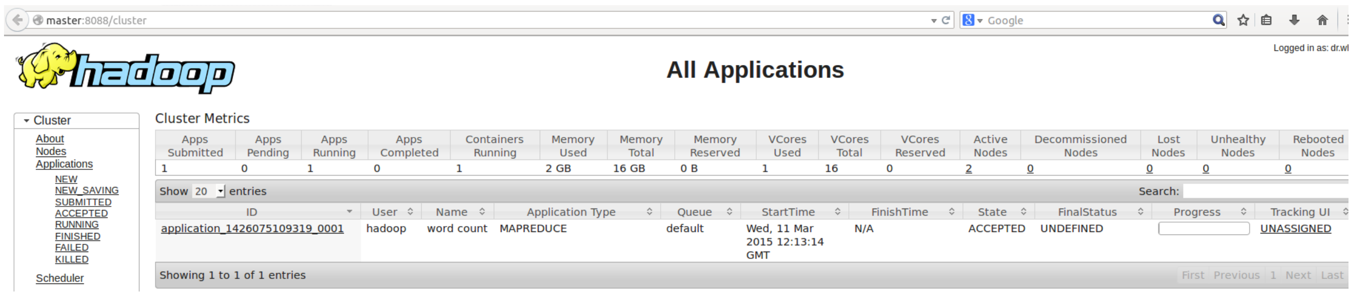


Datanode Information

In operation

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
Slave1 (10.187.114.76:50010)	0	In Service	13.66 GB	104.8 KB	7.92 GB	5.73 GB	35	104.8 KB (0%)	0	2.6.0
Slave2 (10.187.244.157:50010)	1	In Service	13.66 GB	104.8 KB	7.18 GB	6.47 GB	35	104.8 KB (0%)	0	2.6.0

Fig. 24. Check Datanodes online before executing *WordCount*



All Applications

Cluster Metrics

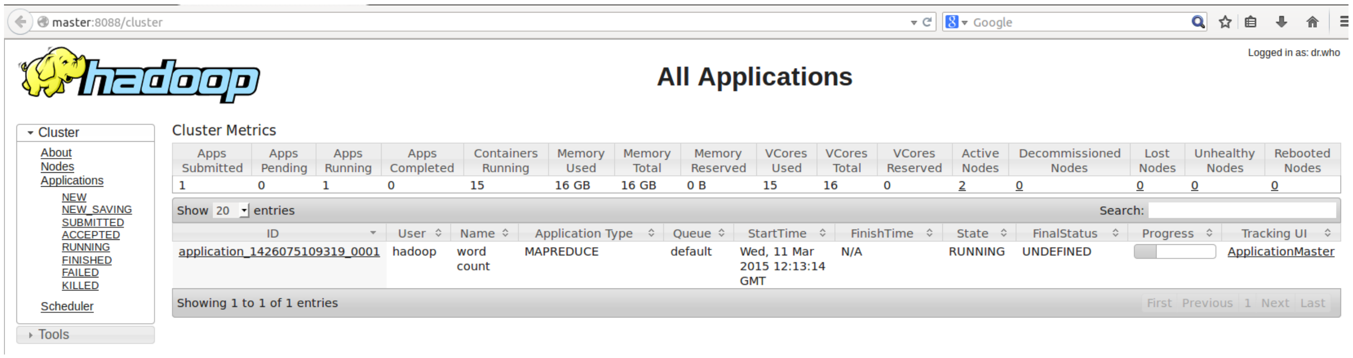
Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
1	0	1	0	1	2 GB	16 GB	0 B	1	16	0	2	0	0	0	0

Show 20 entries Search:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_1426075109319_0001	hadoop	word count	MAPREDUCE	default	Wed, 11 Mar 2015 12:13:14 GMT	N/A	ACCEPTED	UNDEFINED		UNASSIGNED

Showing 1 to 1 of 1 entries First Previous 1 Next Last

Fig. 25. The beginning of *WordCount*



All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
1	0	1	0	15	16 GB	16 GB	0 B	15	16	0	2	0	0	0	0

Show 20 entries Search:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_1426075109319_0001	hadoop	word count	MAPREDUCE	default	Wed, 11 Mar 2015 12:13:14 GMT	N/A	RUNNING	UNDEFINED		ApplicationMaster

Showing 1 to 1 of 1 entries First Previous 1 Next Last

Fig. 26. During the process of *WordCount*

D. Stop HDFS

Finally, do not forget to stop hadoop, or strange troubles may occur.(see in Figure 31)

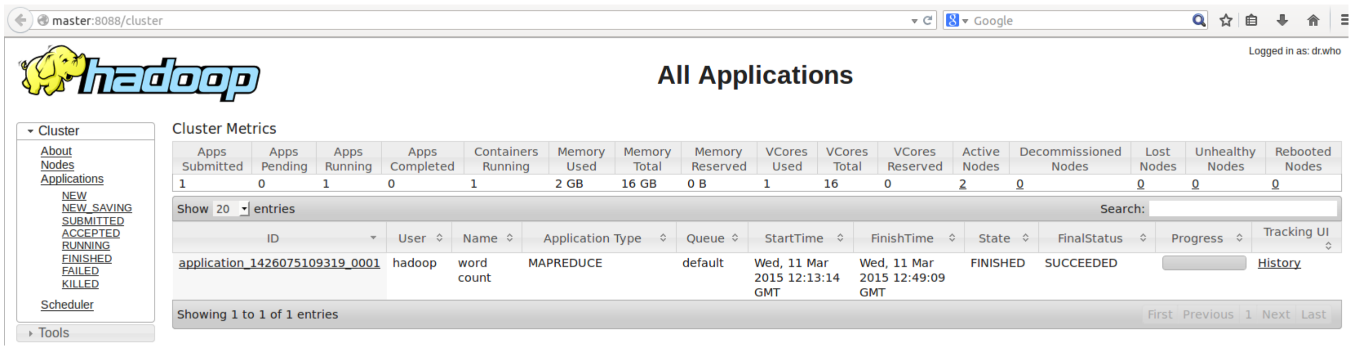
```
sbin/stop-dfs.sh
sbin/stop-yarn.sh
```

OK, that's all the steps to build and run hadoop in a cluster.

VII. Conclusion & Additional Finding

A. Conclusion

From this project, we have known the basic steps to construct hadoop in both single mode and cluster mode. Furthermore, we have learned from the project that Hadoop Distributed File System(HDFS) is in a master/slave structure. There is only a single NameNode on one HDFS, which runs on Master. Master is reponsible for managing the namespaces of HDFS and the visit from clients. Also, HDFS has many DataNodes, usually one DataNode per slave machine, which is used for storaging the data.



All Applications

Cluster Metrics

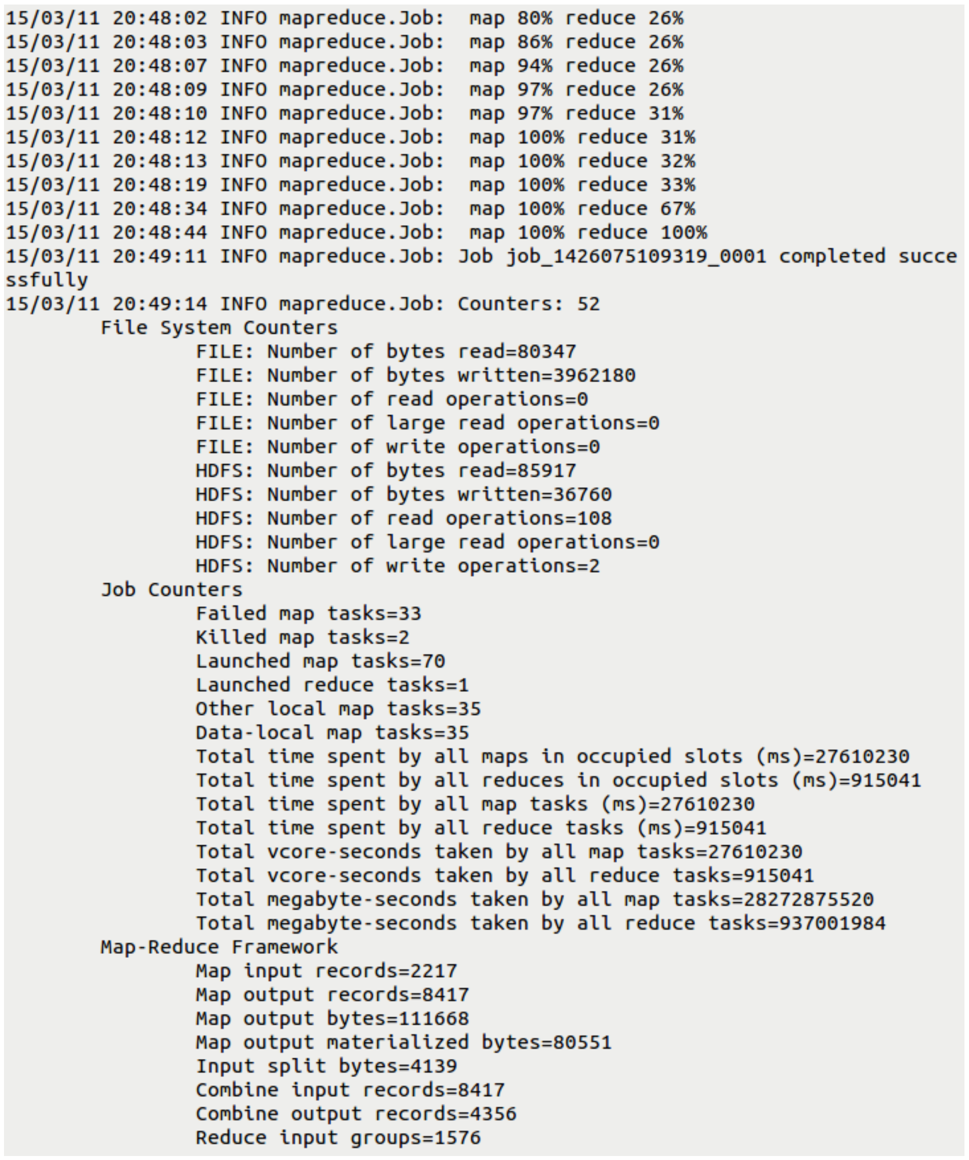
Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
1	0	1	0	1	2 GB	16 GB	0 B	1	16	0	2	0	0	0	0

Show 20 entries Search:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_1426075109319_0001	hadoop	word count	MAPREDUCE	default	Wed, 11 Mar 2015 12:13:14 GMT	Wed, 11 Mar 2015 12:49:09 GMT	FINISHED	SUCCEEDED		History

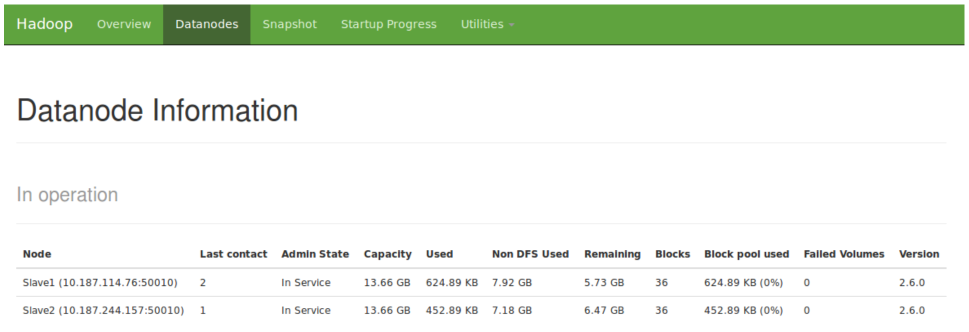
Showing 1 to 1 of 1 entries First Previous 1 Next Last

Fig. 27. Finishing the task *WordCount*



```
15/03/11 20:48:02 INFO mapreduce.Job:  map 80% reduce 26%
15/03/11 20:48:03 INFO mapreduce.Job:  map 86% reduce 26%
15/03/11 20:48:07 INFO mapreduce.Job:  map 94% reduce 26%
15/03/11 20:48:09 INFO mapreduce.Job:  map 97% reduce 26%
15/03/11 20:48:10 INFO mapreduce.Job:  map 97% reduce 31%
15/03/11 20:48:12 INFO mapreduce.Job:  map 100% reduce 31%
15/03/11 20:48:13 INFO mapreduce.Job:  map 100% reduce 32%
15/03/11 20:48:19 INFO mapreduce.Job:  map 100% reduce 33%
15/03/11 20:48:34 INFO mapreduce.Job:  map 100% reduce 67%
15/03/11 20:48:44 INFO mapreduce.Job:  map 100% reduce 100%
15/03/11 20:49:11 INFO mapreduce.Job: Job job_1426075109319_0001 completed succe
ssfully
15/03/11 20:49:14 INFO mapreduce.Job: Counters: 52
        File System Counters
                FILE: Number of bytes read=80347
                FILE: Number of bytes written=3962180
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=85917
                HDFS: Number of bytes written=36760
                HDFS: Number of read operations=108
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Failed map tasks=33
                Killed map tasks=2
                Launched map tasks=70
                Launched reduce tasks=1
                Other local map tasks=35
                Data-local map tasks=35
                Total time spent by all maps in occupied slots (ms)=27610230
                Total time spent by all reduces in occupied slots (ms)=915041
                Total time spent by all map tasks (ms)=27610230
                Total time spent by all reduce tasks (ms)=915041
                Total vcore-seconds taken by all map tasks=27610230
                Total vcore-seconds taken by all reduce tasks=915041
                Total megabyte-seconds taken by all map tasks=28272875520
                Total megabyte-seconds taken by all reduce tasks=937001984
        Map-Reduce Framework
                Map input records=2217
                Map output records=8417
                Map output bytes=111668
                Map output materialized bytes=80551
                Input split bytes=4139
                Combine input records=8417
                Combine output records=4356
                Reduce input groups=1576
```

Fig. 28. Check process of *WordCount* using command line

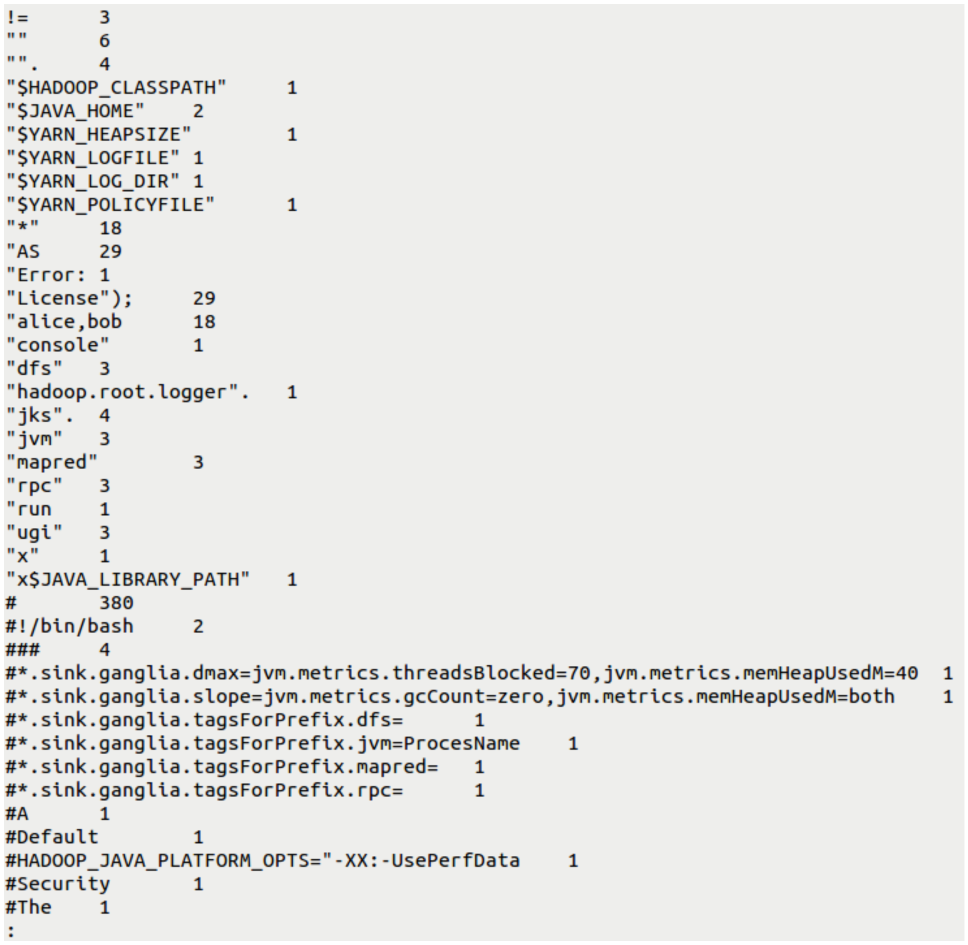


Datanode Information

In operation

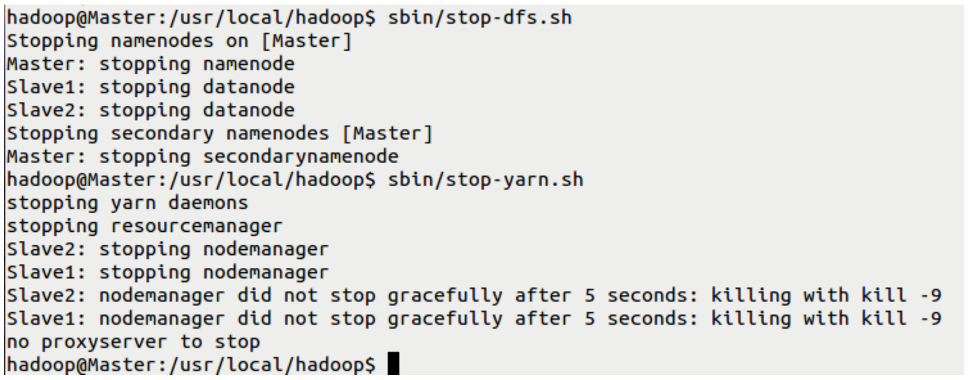
Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
Slave1 (10.187.114.76:50010)	2	In Service	13.66 GB	624.89 KB	7.92 GB	5.73 GB	36	624.89 KB (0%)	0	2.6.0
Slave2 (10.187.244.157:50010)	1	In Service	13.66 GB	452.89 KB	7.18 GB	6.47 GB	36	452.89 KB (0%)	0	2.6.0

Fig. 29. Check Datanodes online after executing *WordCount*



```
!=      3
""      6
"".     4
"$HADOOP_CLASSPATH"     1
"$JAVA_HOME"    2
"$YARN_HEAPSIZE"        1
"$YARN_LOGFILE" 1
"$YARN_LOG_DIR" 1
"$YARN_POLICYFILE"      1
"*"     18
"AS     29
"Error: 1
"License");     29
"alice,bob      18
"console"       1
"dfs"   3
"hadoop.root.logger".   1
"jks".  4
"jvm"   3
"mapred"        3
"rpc"   3
"run    1
"ugi"   3
"x"     1
"x$JAVA_LIBRARY_PATH"   1
#       380
#!/bin/bash     2
###     4
#*.sink.ganglia.dmax=jvm.metrics.threadsBlocked=70,jvm.metrics.memHeapUsedM=40  1
#*.sink.ganglia.slope=jvm.metrics.gcCount=zero,jvm.metrics.memHeapUsedM=both    1
#*.sink.ganglia.tagsForPrefix.dfs=      1
#*.sink.ganglia.tagsForPrefix.jvm=ProcesName    1
#*.sink.ganglia.tagsForPrefix.mapred=   1
#*.sink.ganglia.tagsForPrefix.rpc=      1
#A      1
#Default        1
#HADOOP_JAVA_PLATFORM_OPTS="-XX:-UsePerfData    1
#Security       1
#The    1
:
```

Fig. 30. *WordCount* output



```
hadoop@Master:/usr/local/hadoop$ sbin/stop-dfs.sh
Stopping namenodes on [Master]
Master: stopping namenode
Slave1: stopping datanode
Slave2: stopping datanode
Stopping secondary namenodes [Master]
Master: stopping secondarynamenode
hadoop@Master:/usr/local/hadoop$ sbin/stop-yarn.sh
stopping yarn daemons
stopping resourcemanager
Slave2: stopping nodemanager
Slave1: stopping nodemanager
Slave2: nodemanager did not stop gracefully after 5 seconds: killing with kill -9
Slave1: nodemanager did not stop gracefully after 5 seconds: killing with kill -9
no proxyserver to stop
hadoop@Master:/usr/local/hadoop$ 
```

Fig. 31. Stop HDFS

Files are divided into blocks on DataNodes, usually 64MB per block. Each file may have 2-3 copies separated on different DataNodes to provide storage reliability.

NameNode maintains the namespace, any changes to the category structure of the file system are stored in NameNode. The file system organization is very similar to that of a linux file system. You can create, delete, move and rename files and categories via NameNode. Pay attention that NameNode is only responsible for metadata and there are no data flow from NameNode to clients.

A job is more like an aim we would like to implement using hadoop, while tasks are the subparts of a job. Usually, hadoop divides job into two main kinds of tasks: map tasks and reduce tasks. JobTracker and TaskTracker are two kinds of nodes that control the running of job.

B. Additional Finding

When running *WordCount* in both single mode and cluster mode, we find an interesting phenomenon: it cost several seconds in single mode while cost several minutes in cluster mode. So why the job run much slower using more machines?

This may be caused by the communication cost between seperate machines. Remember that our input data is less than 1MB, so the communication cost far exceeds the computation cost and it turns out that our job runs much faster on a single machine than a distributed cluster.

Hadoop cluster is only suitable for very large number of data(usually TB or PB). When you have to operate on small datasets, using other programming languages and run it on a single machine would be a much better choice.

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

[2] *Install Hadoop2.4.0 On Ubuntu14.04 (Single Mode)*, http://www.cnblogs.com/kinglau/p/3794433.html

[3] *Guidance of Installation and Configuration of Hadoop in a cluster: Hadoop2.4.1/Ubuntu 14.04*, http://www.powerxing.com/install-hadoop-cluster-2-4-1/