Lab 3: MapReduce Programming

Aditi Desai ; Gev Manekshaw

1. Objectives

- Understand the MapReduce concept.
- Get familiar with the Hadoop framework.
- Experience working with VMs created by KVM and CloudStack in a small cloud.

2. Equipment Needs

- Computers
- Internet

3. Experiments

3.1 Basics

a) Go through the Apache Hadoop introduction to get the general idea about Hadoop: http://hadoop.apache.org/

b) Go through the Apache Hadoop release notes to understand the evolution of Hadoop: http://hadoop.apache.org/releases.html

3.2 Hadoop Single Node Mode.

a) Follow the instructions on http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html to set up Hadoop environment on your own Linux machine or use QuickStart VM from Cloudera http://www.cloudera.com/content/cloudera/en/documentation/core/v5-2-x/topics/cloudera_quickstart_vm.html

b) Follow the instructions/tutorials from the above links to run simple word count program using MapReduce.

3.3 Hadoop "cluster"

Your TA will provide you with login information for connecting to a small cloud. Connect to the cloud, create VMs and run a simple Word Count program in Hadoop.

a) SSH to the CloudStack.

b) Two VMs are already created for each group, one is the master node and the other one is the slave node. Please use the corresponding IP addresses to SSH the two VMs your groups is assigned.

c) Configure Master and Slave nodes to create a small Hadoop cluster.
d) Run the WordCount Hadoop job on the file the TA provides to you.

4. Reports

(a) What are the differences between Hadoop 0.X, 1.X, 2.X?

Hadoop 1.x uses MRv1 where in Master does the job of JobTracker and Slave does the job of TaskTracker. Hadoop 2.x uses MRv2 also known as YARN (Yet Another Resource Negotiator) wherein Master does the job of Resource Manager, Slave does the job of Node Manager, in addition to this there is Application Specific Application Master.

Hadoop 1.x uses map/reduce slots for running tasks and Hadoop 2.x uses containers which is bunch of resources like memory, cores for running tasks.

Hadoop 1.x does not scale as good as Hadoop 2.x does.

(b) What is YARN? Why do we need YARN?

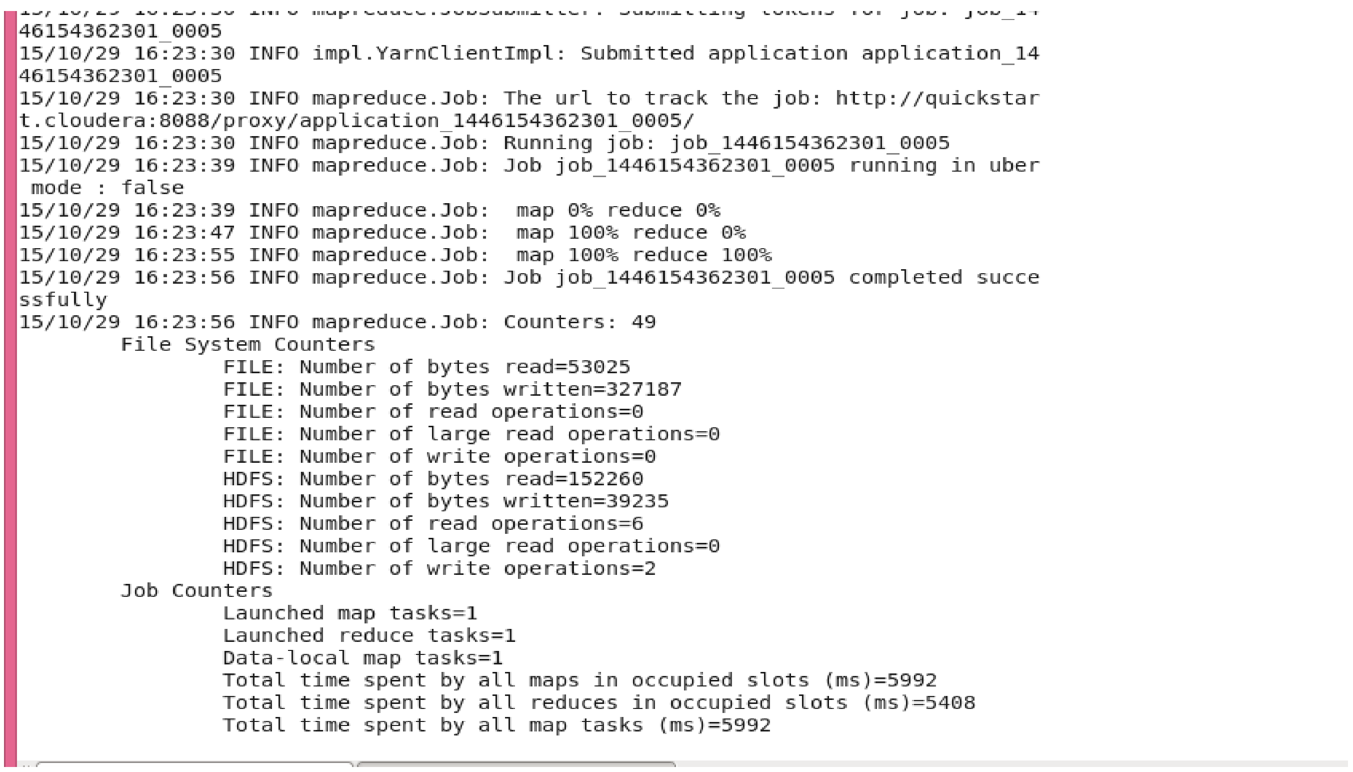
YARN stands for Yet Another Resource Negotiator. It is a key feature of Hadoop 2.x . It is responsible for managing the computing resources in a cluster. It is an operating system for Big Data Applications.

YARN adds the feature of multi-tenancy to Hadoop by allowing multiple engines to used Hadoop to access the same data. Unlike Apache Hadoop MapReduce, with YARN multiple applications, all sharing a common resource management can be run in Hadoop. Due to dynamic allocation of cluster resources YARN improves utilization of Hadoop. YARN helps in improving scalability of Hadoop for increasing processing powers of datacenters. YARN focusses on scheduling and maintains the speed of Hadoop as the cluster expands.

(c) What is Hadoop streaming?

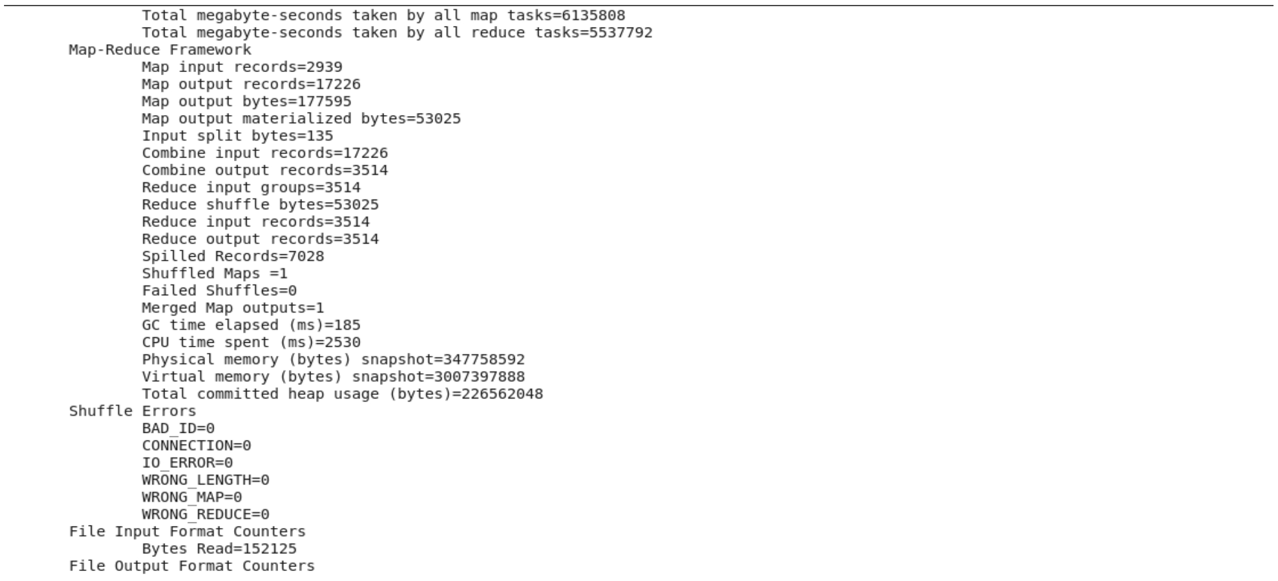
Hadoop streaming allows Hadoop to generate and run Map/Reduce jobs using any form of script. Mappers and Reducers receive input on stdin and emit output on stout. Input and output are in form of (key, value) pairs. It is useful for text processing. Key and value are separated using “Tab”. The Hadoop streaming uses this tab to split line into pair of key and value. The output is emitted in similar format.

(d) Screenshots of the practice (word count) of single node Hadoop on your own computer. The screenshot should show the output and result of Hadoop execution as well as the files in HDFS.

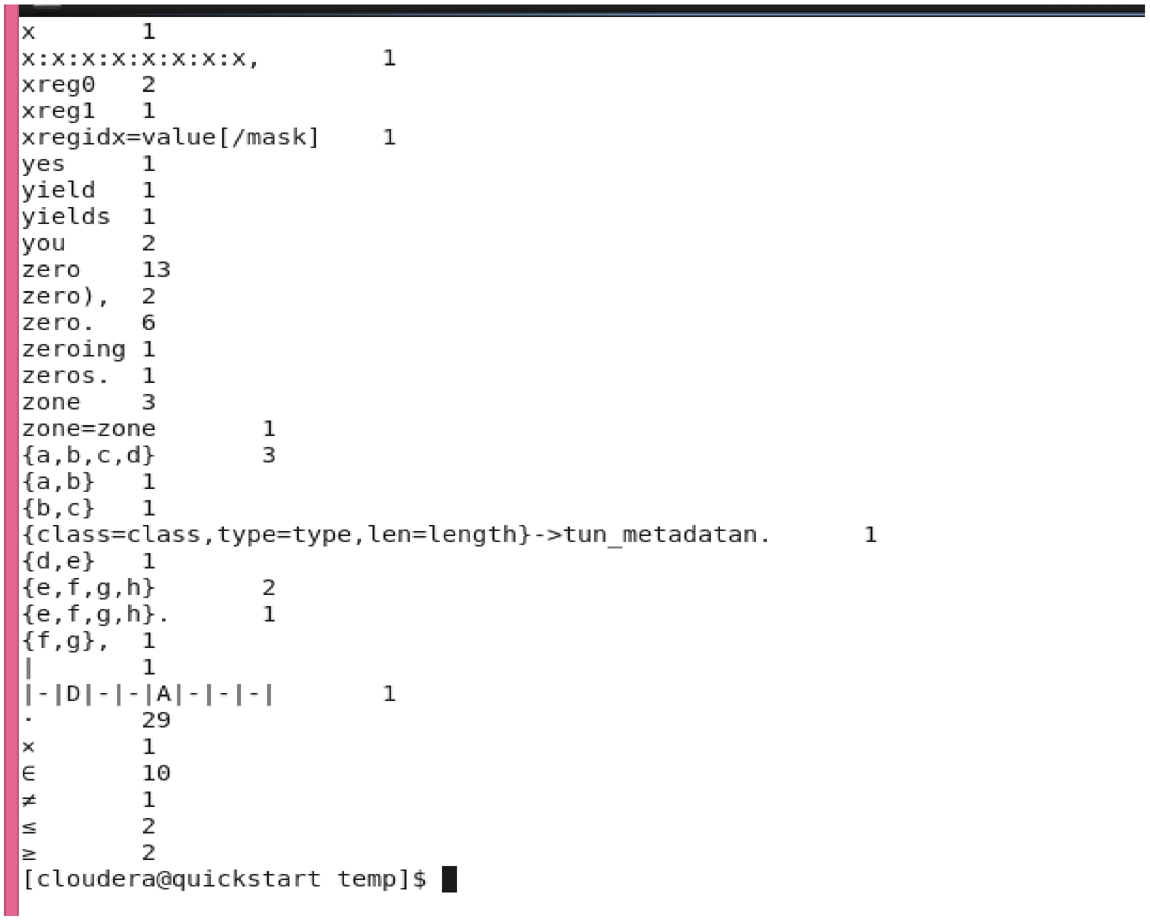


```
15/10/29 16:23:30 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_14
46154362301_0005
15/10/29 16:23:30 INFO impl.YarnClientImpl: Submitted application application_14
46154362301_0005
15/10/29 16:23:30 INFO mapreduce.Job: The url to track the job: http://quickstar
t.cloudera:8088/proxy/application_1446154362301_0005/
15/10/29 16:23:30 INFO mapreduce.Job: Running job: job_1446154362301_0005
15/10/29 16:23:39 INFO mapreduce.Job: Job job_1446154362301_0005 running in uber
 mode : false
15/10/29 16:23:39 INFO mapreduce.Job:  map 0% reduce 0%
15/10/29 16:23:47 INFO mapreduce.Job:  map 100% reduce 0%
15/10/29 16:23:55 INFO mapreduce.Job:  map 100% reduce 100%
15/10/29 16:23:56 INFO mapreduce.Job: Job job_1446154362301_0005 completed succe
ssfully
15/10/29 16:23:56 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=53025
                FILE: Number of bytes written=327187
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=152260
                HDFS: Number of bytes written=39235
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=5992
                Total time spent by all reduces in occupied slots (ms)=5408
                Total time spent by all map tasks (ms)=5992
```

The above screen shot shows that mapping has been completed and reduction has also been completed.

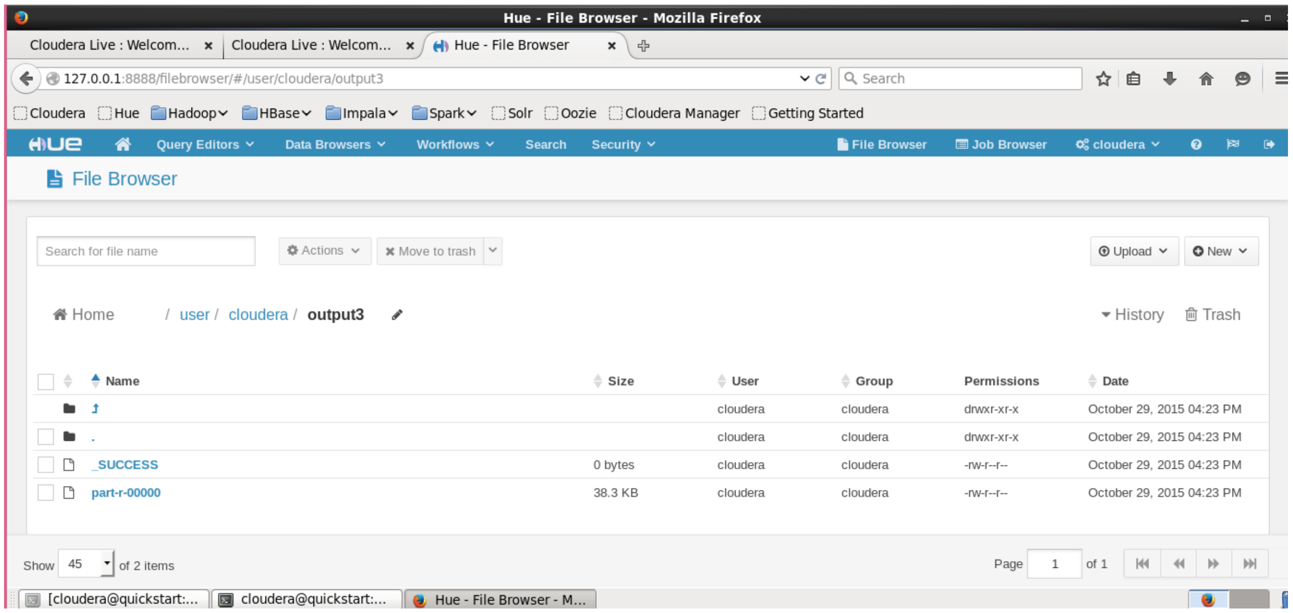


```
                Total megabyte-seconds taken by all map tasks=6135808
                Total megabyte-seconds taken by all reduce tasks=5537792
        Map-Reduce Framework
                Map input records=2939
                Map output records=17226
                Map output bytes=177595
                Map output materialized bytes=53025
                Input split bytes=135
                Combine input records=17226
                Combine output records=3514
                Reduce input groups=3514
                Reduce shuffle bytes=53025
                Reduce input records=3514
                Reduce output records=3514
                Spilled Records=7028
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=185
                CPU time spent (ms)=2530
                Physical memory (bytes) snapshot=347758592
                Virtual memory (bytes) snapshot=3007397888
                Total committed heap usage (bytes)=226562048
        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=152125
        File Output Format Counters
```



```
x       1
x:x:x:x:x:x:x:x,        1
xreg0   2
xreg1   1
xregidx=value[/mask]    1
yes     1
yield   1
yields  1
you     2
zero    13
zero),  2
zero.   6
zeroing 1
zeros.  1
zone    3
zone=zone       1
{a,b,c,d}       3
{a,b}   1
{b,c}   1
{class=class,type=type,len=length}->tun_metadatan.      1
{d,e}   1
{e,f,g,h}       2
{e,f,g,h}.      1
{f,g},  1
|       1
|-|D|-|-|A|-|-|-|       1
·       29
×       1
∈       10
≠       1
≤       2
≥       2
[cloudera@quickstart temp]$
```

The above screen shot shows the frequency of words in the text file.



Hue - File Browser - Mozilla Firefox

127.0.0.1:8888/filebrowser/#/user/cloudera/output3

File Browser

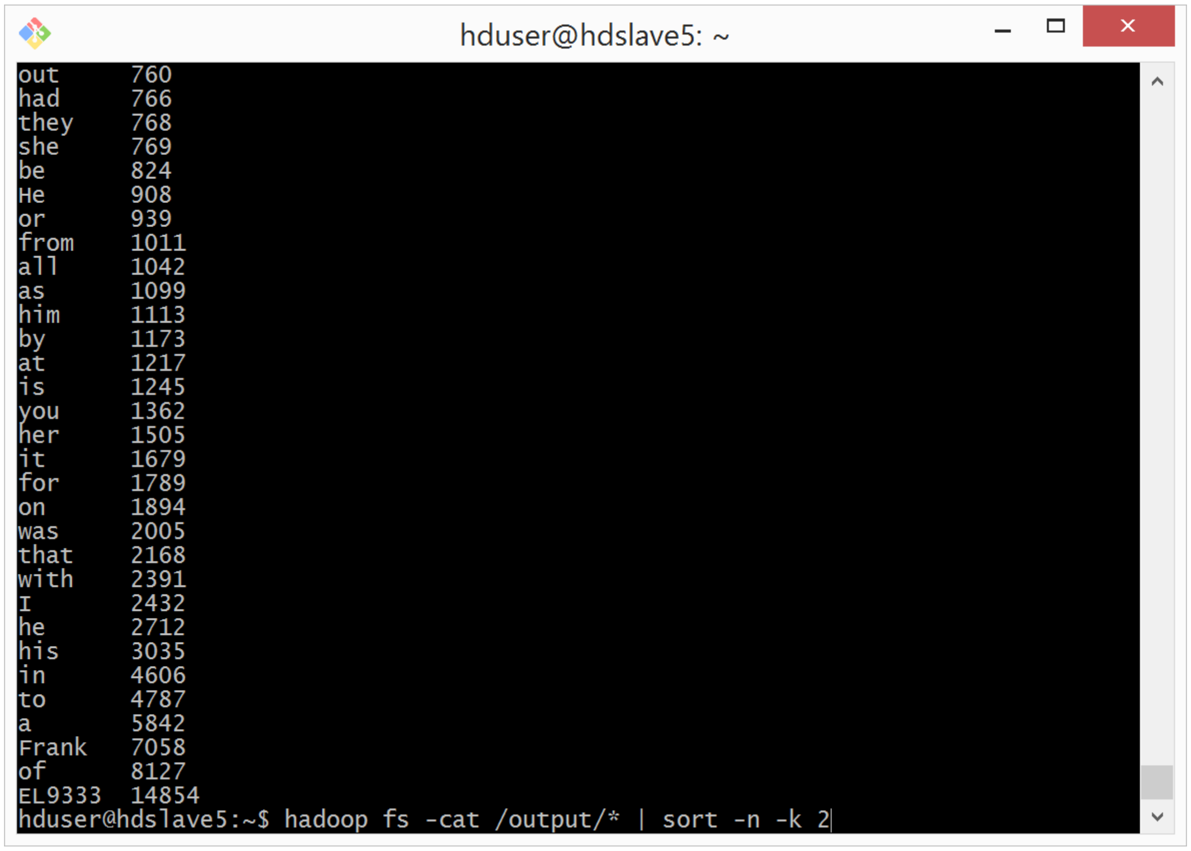
Home / user / cloudera / output3

Name	Size	User	Group	Permissions	Date
↥		cloudera	cloudera	drwxr-xr-x	October 29, 2015 04:23 PM
.		cloudera	cloudera	drwxr-xr-x	October 29, 2015 04:23 PM
_SUCCESS	0 bytes	cloudera	cloudera	-rw-r--r--	October 29, 2015 04:23 PM
part-r-00000	38.3 KB	cloudera	cloudera	-rw-r--r--	October 29, 2015 04:23 PM

Show 45 of 2 items   Page 1 of 1

The above screen shot shows files in hdfs .

(e) What are the top 5 most frequent word in the provided txt file?



```
out     760
had     766
they    768
she     769
be      824
He      908
or      939
from    1011
all     1042
as      1099
him     1113
by      1173
at      1217
is      1245
you     1362
her     1505
it      1679
for     1789
on      1894
was     2005
that    2168
with    2391
I       2432
he      2712
his     3035
in      4606
to      4787
a       5842
Frank   7058
of      8127
EL9333  14854
hduser@hdslave5:~$ hadoop fs -cat /output/* | sort -n -k 2
```

EL9333 – 14854

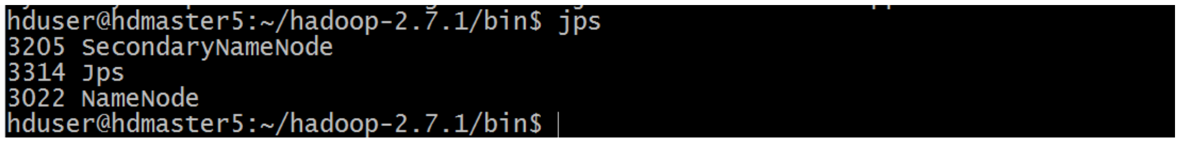
Of – 8127

Frank – 7058

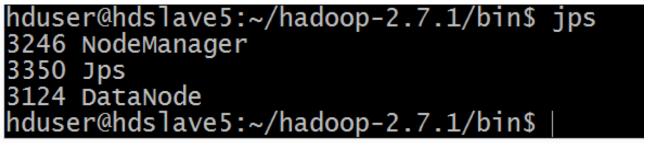
A – 5842

To - 4787

(f) Use jps commands on both VMs to show running Hadoop daemons and provide and screenshots.



```
hduser@hdmaster5:~/hadoop-2.7.1/bin$ jps
3205 SecondaryNameNode
3314 Jps
3022 NameNode
hduser@hdmaster5:~/hadoop-2.7.1/bin$
```

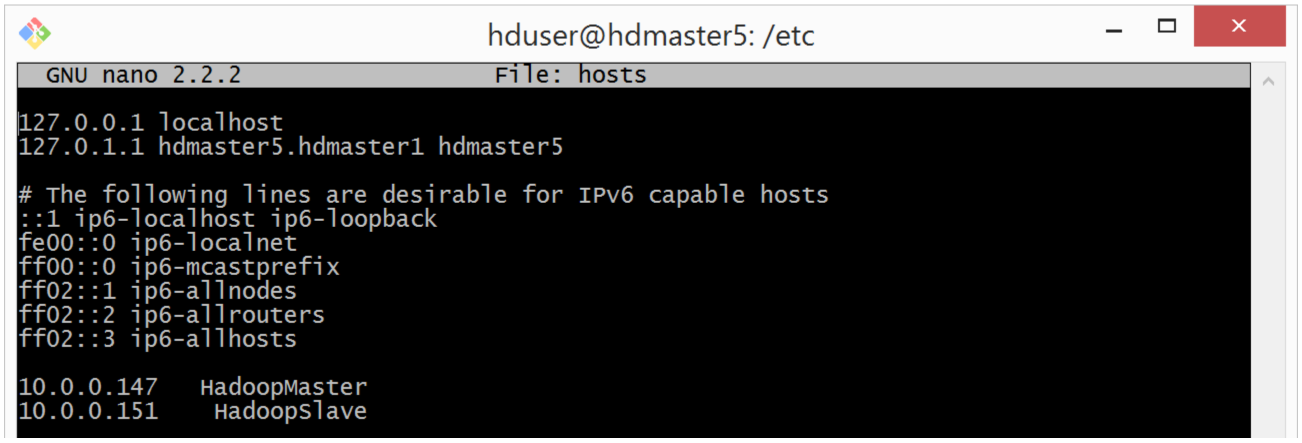


```
hduser@hdslave5:~/hadoop-2.7.1/bin$ jps
3246 NodeManager
3350 Jps
3124 DataNode
hduser@hdslave5:~/hadoop-2.7.1/bin$
```

(g) Screenshots of configuration files and IP addresses for Master node and Slave node on the small cloud as well as the MapReduce execution result. For each configuration file, please also briefly explain what it does.

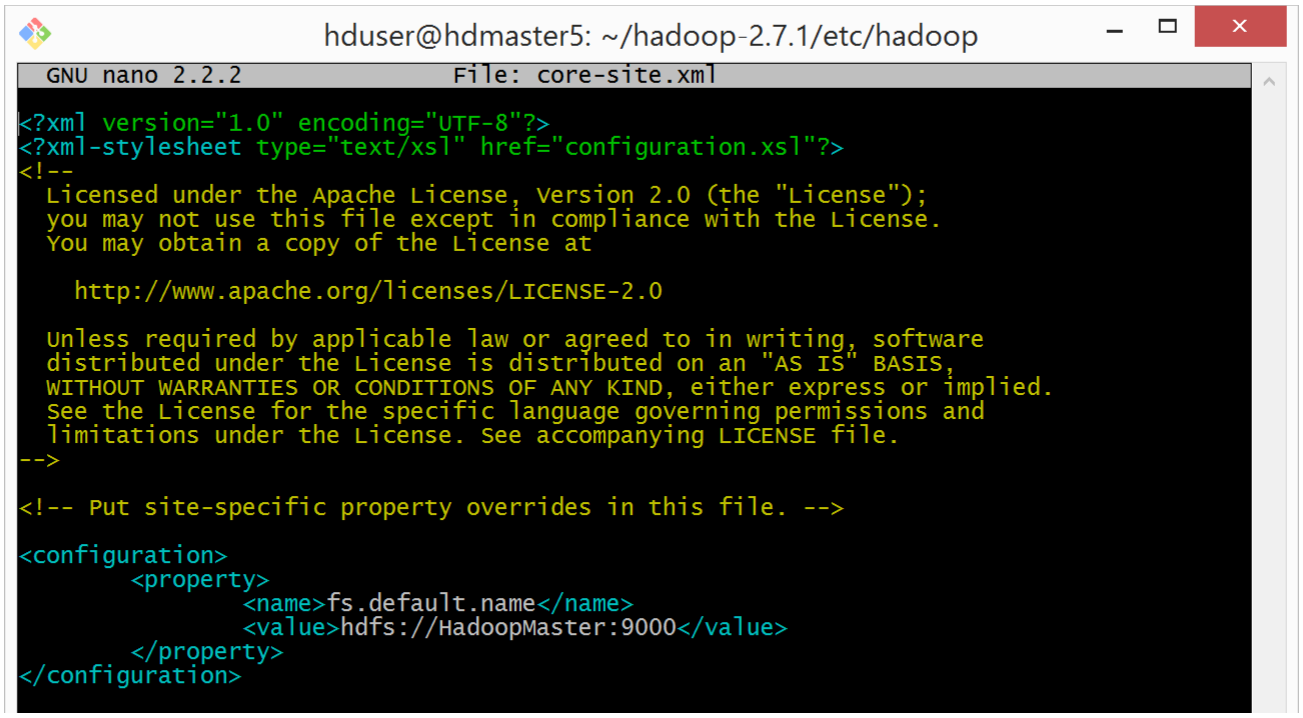
The config files for both master and slave are the same:

The /etc/hosts files defines the IP address of the Master and Slave nodes .



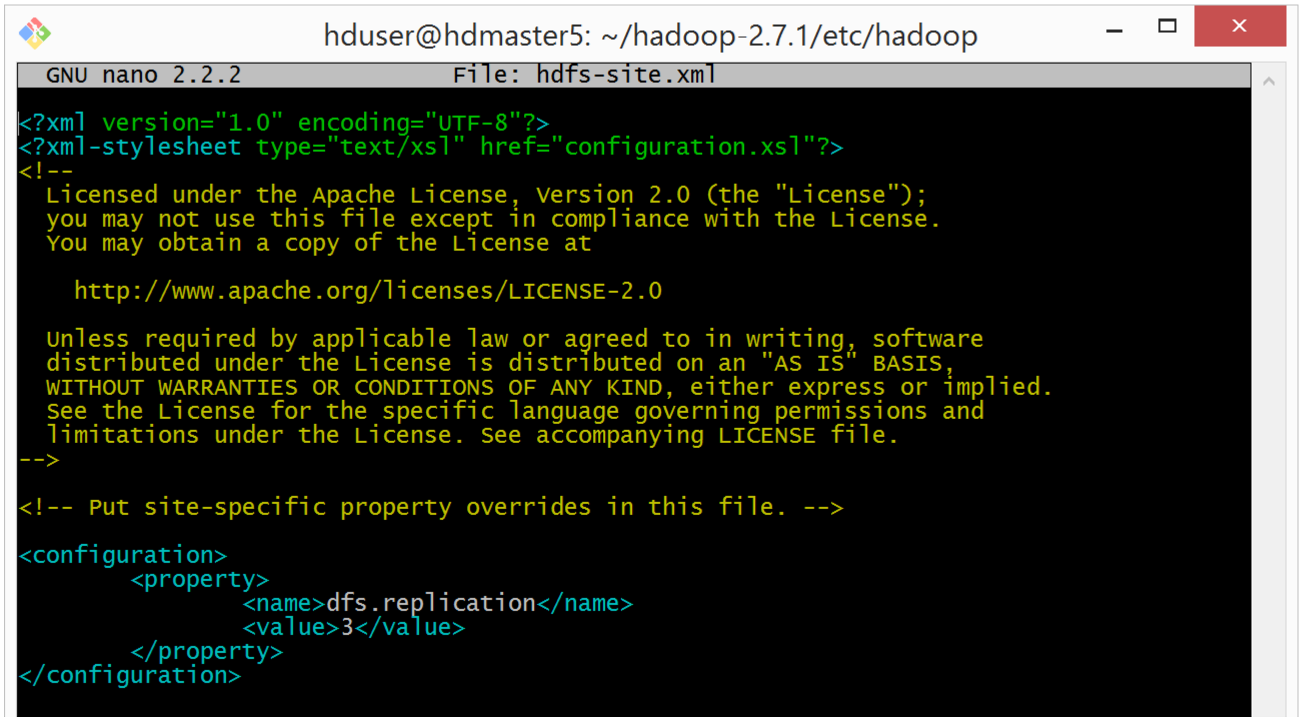
```
hduser@hdmaster5: /etc
  GNU nano 2.2.2                        File: hosts

127.0.0.1 localhost
127.0.1.1 hdmaster5.hdmaster1 hdmaster5

# 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
ff02::3 ip6-allhosts

10.0.0.147   HadoopMaster
10.0.0.151    HadoopSlave
```

The core-site.xml file informs Hadoop daemon where NameNode runs in the cluster. It contains the configuration settings for Hadoop Core such as I/O settings that are common to HDFS and MapReduce.



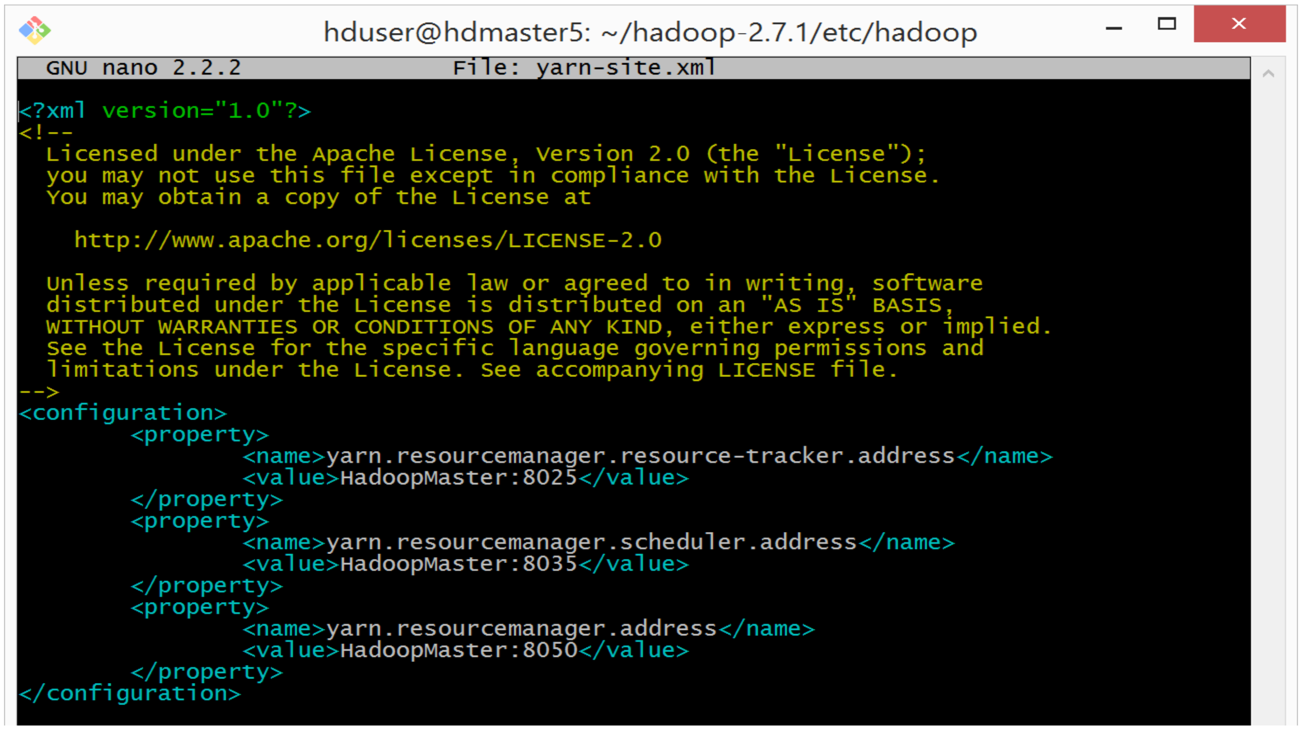
```
hduser@hdmaster5: ~/hadoop-2.7.1/etc/hadoop
  GNU nano 2.2.2                     File: core-site.xml

<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

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

<configuration>
        <property>
                <name>fs.default.name</name>
                <value>hdfs://HadoopMaster:9000</value>
        </property>
</configuration>
```

The hdfs-site.xml file contains the configuration settings for HDFS daemons; the NameNode, the Secondary NameNode, and the DataNodes. Here, we can configure hdfs-site.xml to specify default block replication and permission checking on HDFS.



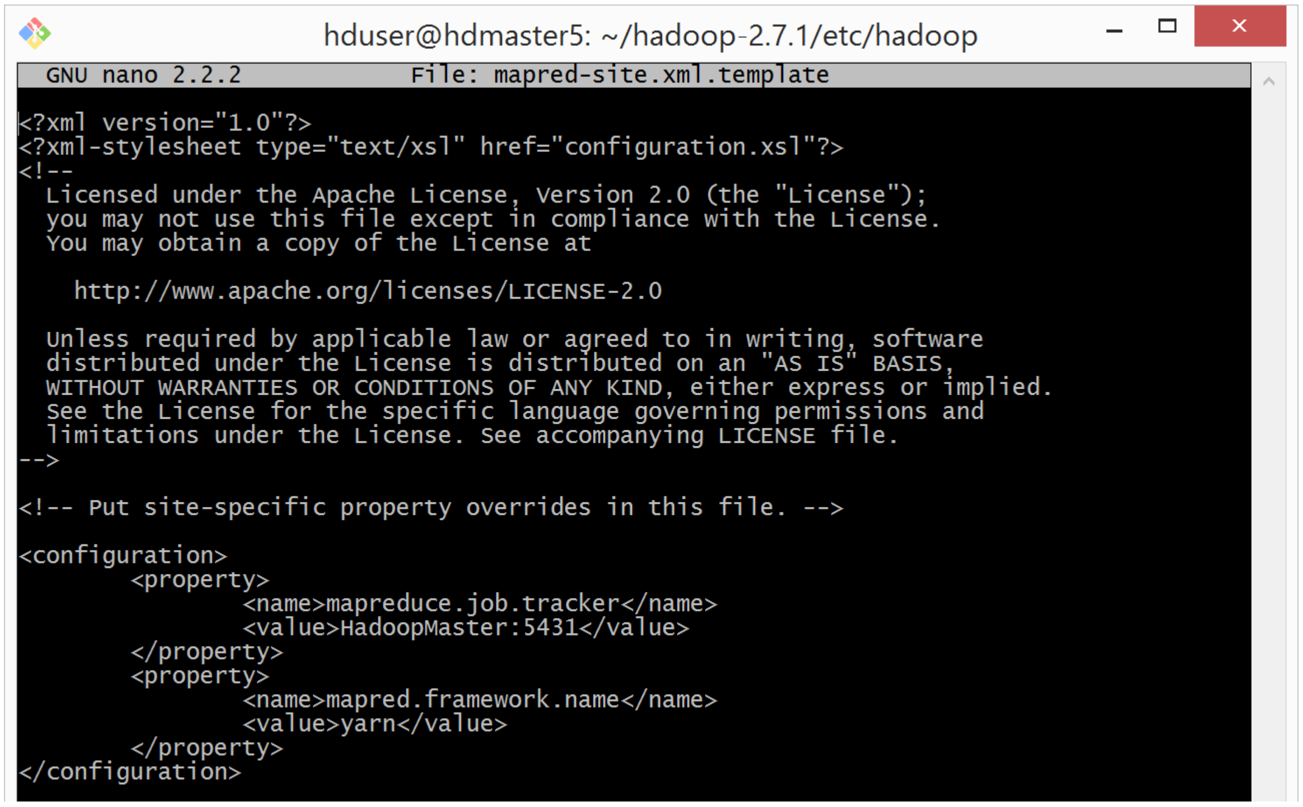
```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

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

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

The yarn-site.xml file contains configuration information that overrides the default values for YARN parameters. Overrides of the default values for core configuration properties are stored in the default YARN parameter file.



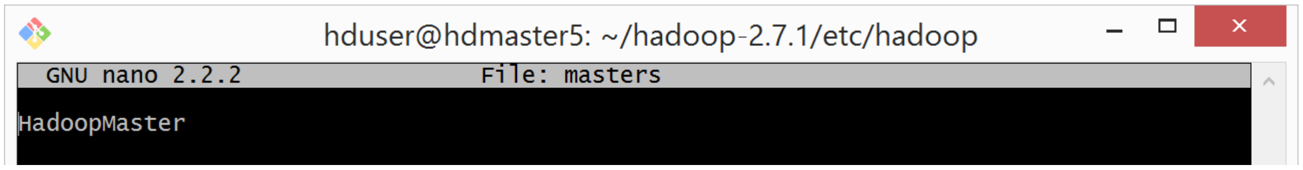
```
hduser@hdmaster5: ~/hadoop-2.7.1/etc/hadoop
  GNU nano 2.2.2                File: yarn-site.xml

<?xml version="1.0"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->
<configuration>
        <property>
                <name>yarn.resourcemanager.resource-tracker.address</name>
                <value>HadoopMaster:8025</value>
        </property>
        <property>
                <name>yarn.resourcemanager.scheduler.address</name>
                <value>HadoopMaster:8035</value>
        </property>
        <property>
                <name>yarn.resourcemanager.address</name>
                <value>HadoopMaster:8050</value>
        </property>
</configuration>
```

The mapred-site.xml file contains the configuration settings for MapReduce daemons; the job tracker and the task-trackers.



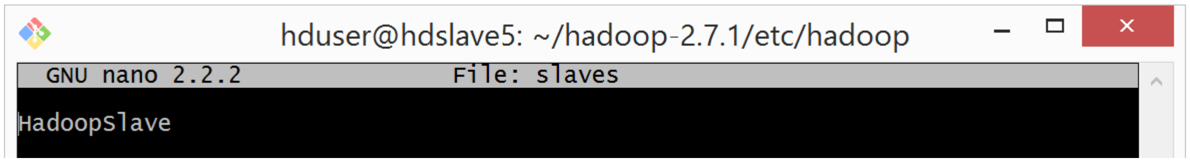
```
hduser@hdmaster5: ~/hadoop-2.7.1/etc/hadoop
  GNU nano 2.2.2             File: mapred-site.xml.template

<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

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

<configuration>
        <property>
                <name>mapreduce.job.tracker</name>
                <value>HadoopMaster:5431</value>
        </property>
        <property>
                <name>mapred.framework.name</name>
                <value>yarn</value>
        </property>
</configuration>
```

The masters file specifies the host name of the Master node



```
hduser@hdmaster5: ~/hadoop-2.7.1/etc/hadoop
  GNU nano 2.2.2                    File: masters

HadoopMaster
```

The salves file specifies the host name of the Slave node



```
hduser@hdslave5: ~/hadoop-2.7.1/etc/hadoop
  GNU nano 2.2.2                  File: slaves

HadoopSlave
```

(h) What are the differences between Hadoop master and slave nodes? Also name which functionalities are performed on each node.

The three types of machine roles in a Hadoop are Client machines, Masters nodes, and Slave nodes.

The Master nodes is responsible for storing lots of data (HDFS) and running parallel computations on all that data (Map Reduce).

Most of the machines in Hadoop architecture are Slave nodes which are responsible for storing the data and running the computations. Every slave runs both Data Node (slave to Name node) and Task Tracer(slave to Job Tracker) daemon that communicates with master nodes.

(i) Write a pseudo code to multiply large matrices using Hadoop. Also explain the function of your Mappers and Reducers.

$C_{IxJ} = A_{IxK} \times B_{KxJ}$

$c(i,j) = \sum a(i,k) * b(k,j)$, from $0 <= k < K$

Splitting A and B into blocks (sub-matrices),

IB = Number of rows per A block and C block.
KB = Number of columns per A block and rows per B block.
JB = Number of columns per B block and C block.

NIB = number of A row and C row partitions = (I-1)/IB+1
NKB = number of A column and B row partitions = (K-1)/KB+1
NJB = number of B column and C column partitions = (J-1)/JB+1

Defining:

IB = Number of rows per A block and C block.
KB = Number of columns per A block and rows per B block.
JB = Number of columns per B block and C block.

NIB = number of A row and C row partitions = (I-1)/IB+1
NKB = number of A column and B row partitions = (K-1)/KB+1
NJB = number of B column and C column partitions = (J-1)/JB+1

Variables:

0 <= ib < NIB
0 <= kb < NKB
0 <= jb < NJB

A[ib,kb] = The block of A consisting of
rows IB*ib through min(IB*(ib+1),I)-1
columns KB*kb through min(KB*(kb+1),K)-1

B[kb,jb] = The block of B consisting of
rows KB*kb through min(KB*(kb+1),K)-1
columns JB*jb through min(JB*(jb+1),J)-1

C[ib,jb] = The block of C consisting of
rows IB*ib through min(IB*(ib+1),I)-1
columns JB*jb through min(JB*(jb+1),J)-1

C[ib,kb,jb] = A[ib,kb] * B[kb,jb]

Function of Mapper:

The mappers are responsible for distributing the block data to the reducers, with the help of a carefully chosen intermediate key structure and key comparator and partitioning functions. They route a copy of each A[ib,kb] block to all of the R[ib,kb,jb] reducers, for each 0 <= jb < NJB. This is NJB copies of A and a total of NJB*I*K key/value pairs, for the worst case where A is dense with no zero elements. Similarly, the mappers must route a copy of each B[kb,jb] block to all the R[ib,kb,jb] reducers, for each 0 <= ib < NIB. This is NIB copies of B for a worst-case total of NIB*K*J key/value pairs. Thus with our first strategy, we must transfer a worst-case grand total of K*(NJB*I + NIB*J) key/value pairs over the network during the sort & shuffle phase of job 1.

Function of Reducer:

The reducers do the block multiplications. This strategy uses lots of reducers, which makes good use of parallelization, but it also requires a large amount of network traffic. Reducer R[ib,kb,jb] is responsible for multiplying A[ib,kb] times B[kb,jb] to produce C[ib,kb,jb]. We have a maximum of NIB*NKB*NJB reducers multiplying blocks in parallel.

Pseudocode:

```
setup ()
   var NIB = (I-1)/IB+1
   var NKB = (K-1)/KB+1
   var NJB = (J-1)/JB+1

map (key, value)
   if from matrix A with key=(i,k) and value=a(i,k)
      for 0 <= jb < NJB
         emit (i/IB, k/KB, jb, 0), (i mod IB, k mod KB, a(i,k))
   if from matrix B with key=(k,j) and value=b(k,j)
      for 0 <= ib < NIB
         emit (ib, k/KB, j/JB, 1), (k mod KB, j mod JB, b(k,j))
   r = ((ib*JB + jb)*KB + kb) mod R

   var A = new matrix of dimension IBxKB
   var B = new matrix of dimension KBxJB
   var sib = -1
   var skb = -1

reduce (key, valueList)
   if key is (ib, kb, jb, 0)
      // Save the A block.
      sib = ib
      skb = kb
      Zero matrix A
      for each value = (i, k, v) in valueList A(i,k) = v
   if key is (ib, kb, jb, 1)
      if ib != sib or kb != skb return   // A[ib,kb] must be zero!
      // Build the B block.
      Zero matrix B
      for each value = (k, j, v) in valueList B(k,j) = v
      // Multiply the blocks and emit the result.
      ibase = ib*IB
      jbase = jb*JB
      for 0 <= i < row dimension of A
         for 0 <= j < column dimension of B
            sum = 0
            for 0 <= k < column dimension of A = row dimension of B
               sum += A(i,k)*B(k,j)
            if sum != 0 emit (ibase+i, jbase+j), sum
```

(j) What is a combiner? Add a combiner to the last question and explain its function.
Outputs of Map Operation are available in memory. Combiner is used to perform reduce type function. When a combiner is used the map key- value pairs, instead of being immediately outputted it will be

collected in lists per value of the key. After certain number of key value pairs are generated, the buffer is emptied by sending all the values of every key to reduce method of the combiner and the key-value pairs generated by the combiners are outputted which seems to be generated by original map operation.

Each job 1 reducer R[ib,kb,jb] emits key/value pairs of the form ((i,j), v). A quick analysis of the algorithm reveals that within this output all the (i,j) pairs are unique. But reducer tasks serve double duty if R < NIB*NKB*NJB, and in this case it is possible that the output file generated by a reducer task can contain more than one key/value pair for the same key pair (i,j).

References:

http://www.norstad.org/matrix-multiply/

http://stratapps.net/hadoop-multinode-installation-guide.php