Hanjiao Qiu
CSCIE63 Big Data Analytics
Assignment 02

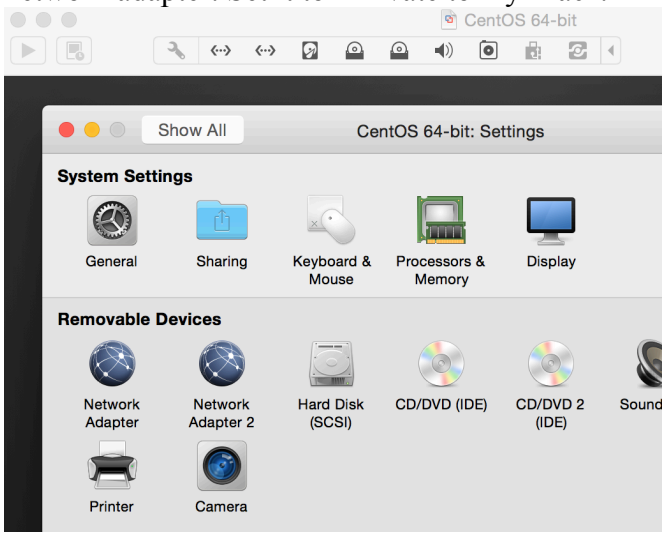
Assignment 02 Solution

Problem 1) Please, download and install VMware Workstation 11 on your 64 bit Windows PC or VMWare Fusion 7, if you are on a MAC. Please download 64 bit CentOS6.7 and create a 64 bit VM. If you know what you are doing and you work with another flavor of Linux supported by CDH5.5.1, please be free to create a virtual machine based on your favorite Linux flavor. Provide your virtual machine with some 40GB of disk space, if you can spare it. For whatever reasons, Hadoop installation appears to prefer to have more than 20 GB of available space. Name the main user of your VM `cloudera`. Do not use name `hadoop`. "`hadoop`" is a bad name for a user, since Hadoop framework has an executable called `hadoop` and it creates many directories with that same name and those would not necessarily be owned by the VM user called `hadoop`. On that VM create yet another user called `joe`. Make both users `sudo` users. Once your CentOS is fully installed, please shut the VM down and make a copy of the entire directory containing that VM. Name the folder containing that copy differently. Two VMs are identical and you could even run them simultaneously if your machine has enough memory. In the folder of each VM add a text file describing OS on your VM, `usernames` and passwords of important users. This little file will make your VMs useful long into the future. The reason you are creating the backup VM is to save time, if you damage the one VM on which you are installing your software.

Please do not capture installation of Workstation 12 or Fusion. Please do not capture every step in creation of VM. Show addition of the second network adapter. Show steps in creation of user `joe`. Demonstrate that `joe` is a `sudo` user. Show results of your `ifconfig` command.

Solution:

1. Install VMWare Fusion 8 and create a 64 bit CentOS6.7 VM. Add an additional network adapter. Set it to "Private to my Mac".



Show the results of “ifconfig”, it gives two network interfaces now.

```
[cloudera@localhost .ssh]$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:19:F2:B9
          inet addr:192.168.80.132  Bcast:192.168.80.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe19:f2b9/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:42511 errors:0 dropped:0 overruns:0 frame:0
          TX packets:22803 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:60217921 (57.4 MiB)  TX bytes:1388104 (1.3 MiB)

eth1      Link encap:Ethernet  HWaddr 00:0C:29:19:F2:C3
          inet6 addr: fe80::20c:29ff:fe19:f2c3/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:220 (220.0 b)  TX bytes:258 (258.0 b)
```

2. Create a user “joe” and add it to the “sudo” user. First change to “root” user.

```
[cloudera@localhost ~]$ su -
Password:
```

Add a user by command “adduser <username>”.

```
[root@localhost ~]# adduser joe
[root@localhost ~]# passwd joe
Changing password for user joe.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

Make user “joe” and “cloudera” as “sudo” user. Modify file “/etc/sudoers” using “visudo”, add two lines below. Remember to change the permission of the file “/etc/sudoers” back after modification.

```
[root@localhost ~]# chmod a+w /etc/sudoers
[root@localhost ~]# visudo -f /etc/sudoers
## Same thing without a password
# %wheel        ALL=(ALL)       NOPASSWD: ALL
cloudera        ALL=(ALL)       NOPASSWD: ALL
joe             ALL=(ALL)       NOPASSWD: ALL
[root@localhost ~]# chmod 400 /etc/sudoers
[root@localhost ~]# ls -lart /etc/sudoers
-r--------. 1 root root 4099 Feb 11 06:51 /etc/sudoers
```

Both “joe” and “cloudera” can do “sudo yum update” without entering password.

```
[joe@localhost ~]$ sudo yum update
Loaded plugins: fastestmirror, refresh-packagekit, security
Setting up Update Process
Loading mirror speeds from cached hostfile
 * base: repos.dfw.quadranet.com
 * extras: repos.dfw.quadranet.com
 * updates: mirror.unl.edu
No Packages marked for Update


[cloudera@localhost ~]$ sudo yum update
Loaded plugins: fastestmirror, refresh-packagekit, security
Setting up Update Process
Loading mirror speeds from cached hostfile
 * base: repos.dfw.quadranet.com
 * extras: repos.dfw.quadranet.com
 * updates: mirror.unl.edu
No Packages marked for Update
```

Problem 2) Use one of above VMs and follow closely steps in the CDH5.5.1 Quick Start Guide, or my notes. PDF and PPT formats and characters on PC do not always map well into Unix (Linux) characters. If you want to copy commands from the guide you are better off doing it from the HTML version of the CDH Quick Start Guide, which you could find at:
http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cm_qs_quick_start.html and open from with your VM.
Both my notes and the Quick Start Guide will lead you through a "semi-automated" process of installing Hadoop. Please install YARN version of Hadoop. My notes add a few explanations beyond what you can see in the Cloudera's guide. Read the notes and the guide very carefully. Do not execute commands for flavors of Linux other than RedHat (CentOS) unless you are working with another flavor purposefully. You will know that you have successfully installed Hadoop if all of tests described in the guide work properly.

Solution:

1. Download Java SE 8 and install Jave SE Development Kit 8u60. Set JAVA_HOME for user root.

2. Install CDH 5.5.1 on a Single Pseudo-Distributed Node.
   Download file https://archive.cloudera.com/cdh5/one-click-install/redhat/6/x86_64/cloudera-cdh-5-0.x86_64.rpm

   Login into the VM from my laptop. Install the RPM file. Accept all defaults when asked. We will get "Complete!" in the end.

```
hqiu@bos-mp9cx>> ssh -i id_rsa cloudera@192.168.80.134
The authenticity of host '192.168.80.134 (192.168.80.134)' can't be established.
RSA key fingerprint is 42:26:84:c1:63:73:8f:46:bc:46:34:38:43:25:a7:60.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.80.134' (RSA) to the list of known hosts.
cloudera@192.168.80.134's password:
Last login: Thu Feb 11 06:43:25 2016 from 192.168.80.1
```

```
[cloudera@localhost ~]$ cd ~/Downloads/
[cloudera@localhost Downloads]$ sudo yum --nogpgcheck localinstall cloudera-cdh-5-0.x86_64.rpm
Installed:
  cloudera-cdh.x86_64 0:5-0

Complete!
```

   Add the Cloudera Public GPG Key to the local repository.

```
[cloudera@localhost Downloads]$ sudo rpm --import https://archive.cloudera.com/cdh5/redhat/6/x86_64/cdh/RPM-GPG-KEY-cloudera
```

3. Install Hadoop with YARN in pseudo-distributed mode. Before proceeding, clean cached packages to ensure the repos are up to date.

```
[cloudera@localhost Downloads]$ sudo yum clean all
Loaded plugins: fastestmirror, refresh-packagekit, security
Cleaning repos: base cloudera-cdh5 extras updates
Cleaning up Everything
Cleaning up list of fastest mirrors
```

```
[cloudera@localhost Downloads]$ sudo yum install hadoop-conf-pseudo
Loaded plugins: fastestmirror, refresh-packagekit, security
Setting up Install Process
Determining fastest mirrors
 * base: repos.mia.quadranet.com
 * extras: repos.mia.quadranet.com
 * updates: mirrors.tripadvisor.com
base                                                      | 3.7 kB     00:00
base/primary_db                                           | 4.6 MB     00:00
cloudera-cdh5                                             |  951 B     00:00
cloudera-cdh5/primary                                     |  43 kB     00:00
cloudera-cdh5                                                          146/146
extras                                                    | 3.4 kB     00:00
extras/primary_db                                         |  34 kB     00:00
updates                                                   | 3.4 kB     00:00
updates/primary_db                                        | 3.9 MB     00:00
Resolving Dependencies
--> Running transaction check
```

View configuration files for YARN on CentOS.

```
[cloudera@localhost Downloads]$ rpm -ql hadoop-conf-pseudo
/etc/hadoop/conf.pseudo
/etc/hadoop/conf.pseudo/README
/etc/hadoop/conf.pseudo/core-site.xml
/etc/hadoop/conf.pseudo/hadoop-env.sh
/etc/hadoop/conf.pseudo/hadoop-metrics.properties
/etc/hadoop/conf.pseudo/hdfs-site.xml
/etc/hadoop/conf.pseudo/log4j.properties
/etc/hadoop/conf.pseudo/mapred-site.xml
/etc/hadoop/conf.pseudo/yarn-site.xml
```

4. Format the HDFS file system.

```
[cloudera@localhost conf.pseudo]$ sudo -u hdfs hdfs namenode -format
16/02/11 09:11:42 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = localhost/127.0.0.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.6.0-cdh5.5.1
STARTUP_MSG:   classpath = /etc/hadoop/conf:/usr/lib/hadoop/lib/commons-beanutil
16/02/11 09:11:44 INFO util.ExitUtil: Exiting with status 0
16/02/11 09:11:44 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at localhost/127.0.0.1
************************************************************/
```

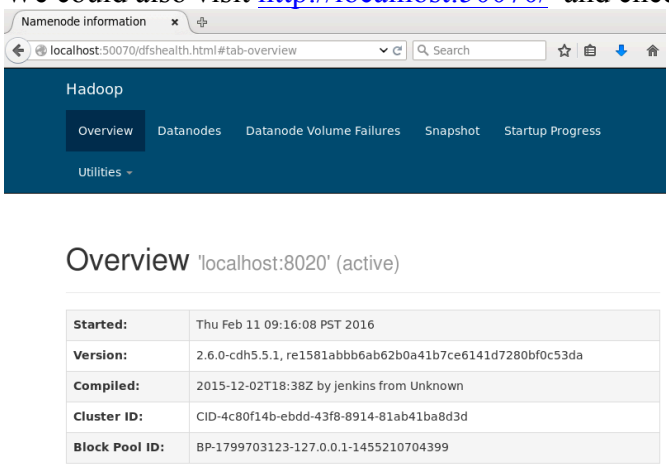
Start HDFS services.

```
[cloudera@localhost ~]$ for x in `cd /etc/init.d; ls hadoop-hdfs-*`; do sudo service $x start; done
starting datanode, logging to /var/log/hadoop-hdfs/hadoop-hdfs-datanode-localhost.localdomain.out
Started Hadoop datanode (hadoop-hdfs-datanode):            [  OK  ]
starting namenode, logging to /var/log/hadoop-hdfs/hadoop-hdfs-namenode-localhost.localdomain.out
Started Hadoop namenode:                                   [  OK  ]
starting secondarynamenode, logging to /var/log/hadoop-hdfs/hadoop-hdfs-secondarynamenode-localhost.
localdomain.out
Started Hadoop secondarynamenode:                          [  OK  ]
```

To verify the services have started. Check HDFS report.

```
[cloudera@localhost ~]$ sudo -u hdfs hdfs dfsadmin -report
Configured Capacity: 37668720640 (35.08 GB)
Present Capacity: 31805779968 (29.62 GB)
DFS Remaining: 31805755392 (29.62 GB)
DFS Used: 24576 (24 KB)
DFS Used%: 0.00%
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0
Missing blocks (with replication factor 1): 0
```

We could also visit http://localhost:50070/ and check DFS status page.



5. Create the /tmp, /var and /var/log HDFS directories.

```
[cloudera@localhost ~]$ sudo /usr/lib/hadoop/libexec/init-hdfs.sh
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -mkdir -p /tmp'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -chmod -R 1777 /tmp'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -mkdir -p /var'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -mkdir -p /var/log'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -chmod -R 1775 /var/log'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -chown yarn:mapred /var/log'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -mkdir -p /tmp/hadoop-yarn'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -chown -R mapred:mapred /tmp/hadoop-yarn'
+ su -s /bin/bash hdfs -c '/usr/bin/hadoop fs -mkdir -p /tmp/hadoop-yarn/staging/history/done_interm
ediate'
```

View HDFS file structure.

```
[cloudera@localhost ~]$ sudo -u hdfs hadoop fs -ls -R /
drwxrwxrwx   - hdfs  supergroup          0 2016-02-11 09:25 /benchmarks
drwxr-xr-x   - hbase supergroup          0 2016-02-11 09:25 /hbase
drwxrwxrwt   - hdfs  supergroup          0 2016-02-11 09:25 /tmp
drwxrwxrwt   - mapred mapred              0 2016-02-11 09:25 /tmp/hadoop-yarn
drwxrwxrwt   - mapred mapred              0 2016-02-11 09:25 /tmp/hadoop-yarn/staging
drwxrwxrwt   - mapred mapred              0 2016-02-11 09:25 /tmp/hadoop-yarn/staging/history
drwxrwxrwt   - mapred mapred              0 2016-02-11 09:25 /tmp/hadoop-yarn/staging/history/done_i
ntermediate
```

Start YARN and map-reduce.

```
[cloudera@localhost ~]$ cd /etc/init.d/
[cloudera@localhost init.d]$ sudo service hadoop-yarn-resourcemanager start
starting resourcemanager, logging to /var/log/hadoop-yarn/yarn-yarn-resourcemanager-localhost.localdomain.out
Started Hadoop resourcemanager:                            [  OK  ]
[cloudera@localhost init.d]$ sudo service hadoop-yarn-nodemanager start
starting nodemanager, logging to /var/log/hadoop-yarn/yarn-yarn-nodemanager-localhost.localdomain.out
Started Hadoop nodemanager:                                [  OK  ]
[cloudera@localhost init.d]$ sudo service hadoop-mapreduce-historyserver start
starting historyserver, logging to /var/log/hadoop-mapreduce/mapred-mapred-historyserver-localhost.localdomain.
out
16/02/11 10:28:42 INFO hs.JobHistoryServer: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting JobHistoryServer
STARTUP_MSG:   host = localhost/127.0.0.1
STARTUP_MSG:   args = []
STARTUP_MSG:   version = 2.6.0-cdh5.5.1
STARTUP_MSG:   classpath = /etc/hadoop/conf:/usr/lib/hadoop/lib/commons-beanutils-1.7.0.jar:/usr/lib/hadoop/lib
/activation-1.1.jar:/usr/lib/hadoop/lib/junit-4.11.jar:/usr/lib/hadoop/lib/jackson-xc-1.8.8.jar:/usr/lib/hadoop
```

Problem 3) As your new Linux user joe fetch the .txt version of James Joyce's Ulysses by issuing the following command on the command prompt:

`wget http://www.gutenberg.org/files/4300/4300.zip`

Unzip the file. Open the resulting txt file with Vi and convince yourself that the life of Buck Mulligan is in front of you. Create a HDFS directory called `ulysses` and copy the .txt file into that HDFS directory. Do not create another HDFS directory called `counted`. The Map Reduce job you will run will create that directory for its output. Actually, if the directory preexists the job will raise an error. That same `hadoop-mapreduce-examples.jar` file mentioned in class notes and you used as the final proof that MapReduce works contains another program called `wordcount`. `wordcount` will tell you how many times a word appears in a provided text. Invoke `wordcount` by the following command:

```
$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar wordcount ulysses counted
```

Once the job is finished visit site http://localhost:19888. You will see some statistics on MapReduce jobs executed on your cluster. There will not be much for your short job. In general that is a very useful site.

Copy results of word count analysis to the local file system. Write a small program in any language (or scripting tool) of your choice and order the counting results by the decreasing count. Present the portion of your final result which does not contain so called stop words (the, a, and, or, …) in your report. Submit top 200 words in separate .txt file with your report.

Solution:

1. Fetch the text file using "joe". Here I use user "cloudera", it should be the same.

```
[cloudera@localhost Downloads]$ wget http://www.gutenberg.org/files/4300/4300.zip
--2016-02-11 10:31:13--  http://www.gutenberg.org/files/4300/4300.zip
Resolving www.gutenberg.org... 152.19.134.47
Connecting to www.gutenberg.org|152.19.134.47|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 661646 (646K) [application/zip]
Saving to: "4300.zip"

100%[======================================================>] 661,646      506K/s   in 1.3s

2016-02-11 10:31:15 (506 KB/s) - "4300.zip" saved [661646/661646]

[cloudera@localhost Downloads]$ unzip 4300.zip
Archive:  4300.zip
  inflating: 4300.txt
[cloudera@localhost Downloads]$ ls
4300.txt  4300.zip  cloudera-cdh-5-0.x86_64.rpm  jdk-8u60-linux-x64.rpm
```

2. Create a directory in HDFS called "Ulysses" and copy the file into the folder. First create a new user directory for user "cloudera" on HDFS.

```
[cloudera@localhost Downloads]$ sudo -u hdfs hadoop fs -mkdir /user/cloudera
[cloudera@localhost Downloads]$ sudo -u hdfs hadoop fs -chown cloudera /user/cloudera
[cloudera@localhost Downloads]$ hadoop fs -ls /user
Found 8 items
drwxr-xr-x   - cloudera supergroup          0 2016-02-11 12:17 /user/cloudera
drwxr-xr-x   - mapred   supergroup          0 2016-02-11 09:25 /user/history
drwxrwxrwx   - hive     supergroup          0 2016-02-11 09:26 /user/hive
drwxrwxrwx   - hue      supergroup          0 2016-02-11 09:26 /user/hue
drwxrwxrwx   - jenkins  supergroup          0 2016-02-11 09:25 /user/jenkins
drwxrwxrwx   - oozie    supergroup          0 2016-02-11 09:26 /user/oozie
drwxrwxrwx   - root     supergroup          0 2016-02-11 09:26 /user/root
drwxr-xr-x   - hdfs     supergroup          0 2016-02-11 09:27 /user/spark
```

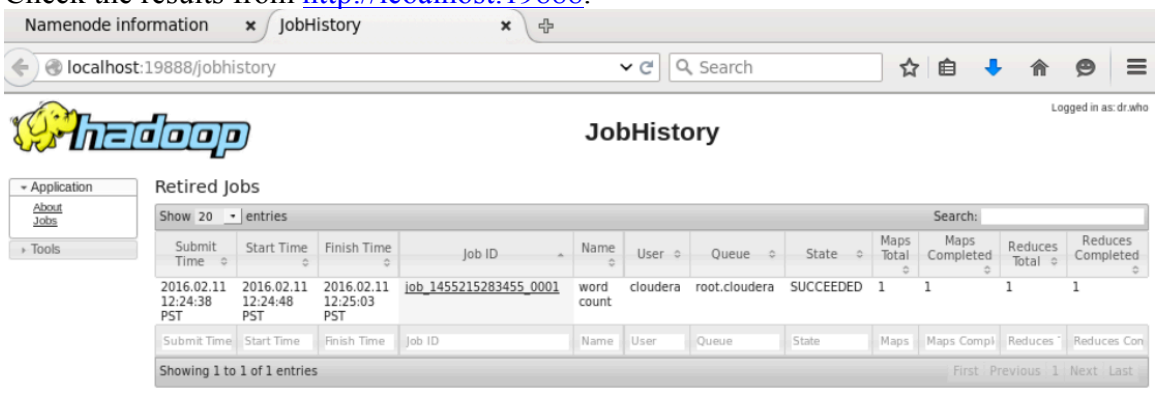
Then create “ulysses” on “cloudera’s” HDFS.

```
[cloudera@localhost ~]$ hadoop fs -mkdir ulysses
[cloudera@localhost ~]$ hadoop fs -put ~/Downloads/4300.txt ulysses
[cloudera@localhost ~]$ hadoop fs -ls
Found 1 items
drwxr-xr-x   - cloudera supergroup          0 2016-02-11 12:21 ulysses
[cloudera@localhost ~]$ hadoop fs -ls ulysses
Found 1 items
-rw-r--r--   1 cloudera supergroup    1573079 2016-02-11 12:21 ulysses/4300.txt
```

3. Invoke the Hadoop Map-Reduce job and invoke “wordcount”.

```
[cloudera@localhost ~]$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar wordcount
 ulysses counted
16/02/11 12:24:36 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
16/02/11 12:24:38 INFO input.FileInputFormat: Total input paths to process : 1
16/02/11 12:24:38 INFO mapreduce.JobSubmitter: number of splits:1
16/02/11 12:24:38 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1455215283455_0001
16/02/11 12:24:38 INFO impl.YarnClientImpl: Submitted application application_1455215283455_0001
16/02/11 12:24:38 INFO mapreduce.Job: The url to track the job: http://localhost:8088/proxy/applicat
ion_1455215283455_0001/
16/02/11 12:24:38 INFO mapreduce.Job: Running job: job_1455215283455_0001
16/02/11 12:24:50 INFO mapreduce.Job: Job job_1455215283455_0001 running in uber mode : false
16/02/11 12:24:50 INFO mapreduce.Job:  map 0% reduce 0%
16/02/11 12:24:58 INFO mapreduce.Job:  map 100% reduce 0%
16/02/11 12:25:04 INFO mapreduce.Job:  map 100% reduce 100%
16/02/11 12:25:05 INFO mapreduce.Job: Job job_1455215283455_0001 completed successfully
16/02/11 12:25:05 INFO mapreduce.Job: Counters: 49
```

Check the results from http://lcoalhost:19888.



```
[cloudera@localhost ~]$ hadoop fs -ls
Found 2 items
drwxr-xr-x   - cloudera supergroup          0 2016-02-11 12:25 counted
drwxr-xr-x   - cloudera supergroup          0 2016-02-11 12:21 ulysses
[cloudera@localhost ~]$ hadoop fs -ls counted
Found 2 items
-rw-r--r--   1 cloudera supergroup          0 2016-02-11 12:25 counted/_SUCCESS
-rw-r--r--   1 cloudera supergroup     527547 2016-02-11 12:25 counted/part-r-00000
```

Redo the above procedure using user “joe”, everything is the same.

```
[joe@localhost Downloads]$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar wordcount ulysses
counted
16/02/12 10:38:02 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
16/02/12 10:38:03 INFO input.FileInputFormat: Total input paths to process : 1
16/02/12 10:38:03 INFO mapreduce.JobSubmitter: number of splits:1
16/02/12 10:38:03 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1455215283455_0003
16/02/12 10:38:03 INFO impl.YarnClientImpl: Submitted application application_1455215283455_0003
16/02/12 10:38:03 INFO mapreduce.Job: The url to track the job: http://localhost:8088/proxy/application_14552152
83455_0003/
16/02/12 10:38:03 INFO mapreduce.Job: Running job: job_1455215283455_0003
16/02/12 10:38:12 INFO mapreduce.Job: Job job_1455215283455_0003 running in uber mode : false
16/02/12 10:38:12 INFO mapreduce.Job:  map 0% reduce 0%
16/02/12 10:38:19 INFO mapreduce.Job:  map 100% reduce 0%
16/02/12 10:38:26 INFO mapreduce.Job:  map 100% reduce 100%
16/02/12 10:38:27 INFO mapreduce.Job: Job job_1455215283455_0003 completed successfully
16/02/12 10:38:27 INFO mapreduce.Job: Counters: 49
```

```
[joe@localhost Downloads]$ hadoop fs -ls counted
Found 2 items
-rw-r--r--   1 joe supergroup          0 2016-02-12 10:38 counted/_SUCCESS
-rw-r--r--   1 joe supergroup     527547 2016-02-12 10:38 counted/part-r-00000
```

4. Copy results of word count analysis to the local file system.

```
[cloudera@localhost ~]$ hadoop fs -copyToLocal counted/part-r-00000 ~/Documents
[cloudera@localhost ~]$ sudo cp ~/Documents/part-r-00000 /mnt/hgfs/VM_shared/
```

5. Sort the results by the decreasing count. I solve it using both Java and Python.

   The Java file is called “WordCounter.java”. It will first re-process the words since there are still some special characters in the words. Then the “reducer()” will merge the results and sort them. The result file is attached as “output.txt”.

```
public static void main (String[] args) throws IOException {
    String filename = "/Users/hqiu/Documents/Virtual Machines.localized/VM_shared/part-r-00000";

    WordCounter wordCounter = new WordCounter();

    List<HashMap<String, Integer>> wordCount = wordCounter.parseFile(filename);

    List<Map.Entry<String, Integer>> sortedEntries = wordCounter.reducer(wordCount);
    wordCounter.printHashEntryList(sortedEntries);
}
```

   Define the stop words and the special characters to be excluded.

```
HashSet<String> stopwords = new HashSet<String>(Arrays.asList("the", "a", "an", "and", "or", "of", "to",
        "about", "above", "after", "all", "are", "be", "but", "he", "she", "by", "can't", "for", "do", "i",
        "has", "don't", "her", "is", "in", "our", "his", "with", "that", "you", "it", "was", "on", "him"));

String delimiters = "[.,;:\\*\\-\\?!\\\"\\[\\]\\(\\)\\{\\}_]+";
```

The “parseFile()” will parse the file and store them into a list. Each element in the list is a (word, count) pair. The word is processed through “processWord()”. After processing, the word listed in the stop word list will be removed.

```
private List<HashMap<String, Integer>> parseFile(String filename) throws IOException {
    File file = new File(filename);

    String line = null;
    List<HashMap<String, Integer>> wordCount = new LinkedList<HashMap<String, Integer>>();


    // Read the input file.
    BufferedReader br = new BufferedReader(new FileReader(file));

    // Divide each line into (word, count) pairs.
    while ((line = br.readLine()) != null) {
        String[] pairs = line.split("\t");
        String word = pairs[0];
        word = processWord(word);
        if (word == null) {
            continue;
        }
        int num = Integer.parseInt(pairs[1]);

        HashMap<String, Integer> pair = new HashMap<String, Integer>();
        pair.put(word, num);

        if (! stopwords.contains(word)) {
            wordCount.add(pair);
        }
    }

    br.close();


    return wordCount;
}
```

This is the processWord() will remove the special characters in the word and transfer them to lower cases.

```
private String processWord(String str) {
    String[] words = str.split(delimiters);

    String word = null;

    for (int i = 0; i < words.length; i++) {
        if (!words[i].trim().isEmpty()) {
            word = words[i];
            break;
        }
    }

    return (word == null? null : word.toLowerCase());
}
```

The sortHashMap() is the real function to do the sort() job. It will first compare the counts of each pair. The count is sorted in the decreasing order. For the same count number, the word will be sorted alphabetically.

```
// Sort the name value pairs.
private List<Map.Entry<String, Integer>> sortHashMap(HashMap<String, Integer> hashmap) {
    List<Map.Entry<String, Integer>> sortedEntries = new LinkedList<Map.Entry<String, Integer>>(hashmap.entrySet());

    Collections.sort(sortedEntries, new Comparator<Map.Entry<String, Integer>>() {
        public int compare(Map.Entry<String, Integer> left, Map.Entry<String, Integer> right) {
            if (left.getValue() != right.getValue()) {
                return right.getValue() - left.getValue();
            }

            return left.getKey().compareTo(right.getKey());
        }
    });

    return sortedEntries;
}
```

The reducer() will merge the results. There may be some duplicated words again after the processing.

```
private List<Map.Entry<String, Integer>> reducer(List<HashMap<String, Integer>> wordCount) {
    HashMap<String, Integer> reducedPairs = new HashMap<String, Integer>();

    for (int i = 0; i < wordCount.size(); i++) {
        HashMap<String, Integer> pair = wordCount.get(i);
        // Each HashMap only have one entry here. Get the key value pair.
        Entry<String, Integer> entry = pair.entrySet().iterator().next();

        String word = entry.getKey();
        int val = entry.getValue();

        // Aggregate to (word, count) pairs.
        if (reducedPairs.containsKey(word)) {
            reducedPairs.put(word, reducedPairs.get(word) + val);
        } else {
            reducedPairs.put(word, val);
        }
    }

    // Sort all the entries based on its occurrence. If the words have the same occurrence,
    // sort them alphabetically.
    List<Map.Entry<String, Integer>> sortedEntries = sortHashMap(reducedPairs);

    return sortedEntries;
}
```

Some results:

```
at      1303    no      691
said    1208    them    672
as      1197    so      618
from    1102    then    579     high    135     heard   127
they    1022    if      564     place   135     once    127
me      942     when    555     miss    134     great   126
bloom   933     which   525     till    134     heart   125
not     914     were    510     turned  134     dead    124
out     899     stephen 505     wife    134     gave    124
what    896     your    496     better  133     mouth   124
my      837     this    493     between 133     another 123
up      829     old     491     upon    133     while   123
had     814     who     488     s       130     big     122
like    731     says    473     against 129     coming  122
their   720     down    452     years   129     hair    122
mr      719     over    443     hear    129     half    122
there   706     now     441     best    127     mother  122
one     705     too     441     dark    127     read    121
have    699     see     435     even    127     water   121
```

The python file is called “WordFreqCounter.py”. It has the same idea as the Java method. We process the words, merge them and put the pairs into a dictionary. The “Counter()” will automatically sort the words and list the most common 200 pairs. The results are quite similar to the Java results. The words with high frequency are the same, only the count numbers are slightly different. It may because the way to process the word is different. Python uses regex here.

WordFreqCounter.py

```
from collections import Counter
from sets import Set
import re

filename = '/Users/hqiu/Documents/Virtual Machines.localized/VM_shared/part-r-00000'

with open(filename) as f:
    lines = f.readlines()

stopwords = Set(['the', 'a', 'an', 'and', 'or', 'of', 'to',
                 'about', 'above', 'after', 'all', 'are', 'be', 'but', 'he', 'she', 'by', 'can\'t', 'for', 'do', 'i',
                 'has', 'don\'t', 'her', 'is', 'in', 'our', 'his', 'with', 'that', 'you', 'it', 'was', 'on', 'him'])

cnt = Counter()

for line in lines:
    pairs = line.split('\t')
    words = re.match(r"[a-zA-Z]+", pairs[0])
    if words is None:
        continue

    word = words.group(0).lower()
    if word in stopwords:
        continue
    num = pairs[1]

    cnt[word] += int(num)

print cnt.most_common(200)
```

```
hqiu@bos-mp9cx>> python WordFreqCounter.py
[('at', 1296), ('said', 1207), ('as', 1176), ('from', 1088), ('they', 1023), ('bloom', 967), ('me', 940), ('not'
, 902), ('out', 898), ('what', 884), ('up', 828), ('my', 822), ('had', 813), ('there', 751), ('like', 728), ('th
eir', 714), ('mr', 703), ('one', 698), ('have', 691), ('them', 671), ('no', 658), ('so', 613), ('then', 568), ('
stephen', 558), ('when', 547), ('if', 544), ('o', 514), ('which', 510), ('were', 508), ('your', 491), ('old', 48
7), ('this', 487), ('who', 481), ('says', 473), ('down', 448), ('man', 448), ('we', 442), ('too', 441), ('over',
 438), ('now', 435), ('see', 429), ('did', 389), ('would', 382), ('time', 379), ('two', 378), ('off', 365), ('ba
ck', 359), ('will', 349), ('other', 334), ('into', 330), ('eyes', 329), ('know', 327), ('where', 318), ('more',
315), ('those', 312), ('some', 311), ('could', 310), ('hand', 306), ('its', 305), ('good', 302), ('father', 298)
, ('street', 293), ('little', 289), ('here', 287), ('yes', 284), ('way', 278), ('first', 274), ('can', 273), ('h
ow', 263), ('say', 262), ('get', 256), ('us', 253), ('only', 253), ('day', 252), ('through', 248), ('never', 247
), ('again', 244), ('come', 244), ('just', 242), ('well', 242), ('long', 240), ('round', 239), ('night', 235), (
'face', 235), ('right', 233), ('god', 232), ('under', 230), ('must', 230), ('go', 229), ('any', 227), ('himself'
, 221), ('head', 221), ('before', 220), ('very', 218), ('name', 218), ('sir', 218), ('woman', 216), ('because',
214), ('put', 208), ('been', 205), ('thing', 200), ('mrs', 198), ('life', 198), ('going', 198), ('came', 197), (
'j', 195), ('john', 194), ('let', 194), ('young', 192), ('still', 190), ('don', 190), ('look', 189), ('asked', 1
87), ('last', 183), ('away', 182), ('made', 182), ('went', 182), ('poor', 181), ('got', 181), ('voice', 177), ('
something', 177), ('why', 176), ('might', 176), ('make', 176), ('being', 174), ('always', 174), ('dedalus', 171)
, ('house', 171), ('than', 171), ('hat', 168), ('love', 167), ('give', 167), ('same', 167), ('course', 167), ('t
ell', 166), ('three', 166), ('much', 165), ('left', 165), ('though', 164), ('mulligan', 162), ('white', 162), ('
ever', 161), ('saw', 160), ('world', 155), ('am', 155), ('new', 154), ('own', 154), ('hands', 153), ('men', 151)
, ('most', 151), ('want', 148), ('lord', 147), ('without', 144), ('behind', 144), ('told', 144), ('think', 142),
 ('took', 141), ('fellow', 141), ('bit', 141), ('black', 140), ('every', 140), ('mother', 140), ('door', 139), (
'wife', 138), ('take', 138), ('joe', 137), ('place', 135), ('such', 135), ('turned', 134), ('high', 134), ('till
', 134), ('years', 132), ('between', 132), ('upon', 132), ('miss', 132), ('best', 131), ('s', 130), ('better', 1
30), ('against', 129), ('hear', 128), ('heard', 127), ('mouth', 126), ('another', 126), ('once', 126), ('great',
 126), ('even', 126), ('dark', 125), ('heart', 125), ('dublin', 125), ('gave', 123), ('m', 123), ('coming', 122)
, ('hair', 122), ('dead', 122), ('while', 122), ('big', 122), ('water', 121), ('read', 120), ('towards', 120), (
'eye', 119)]
```

Problem 4). Consider a symmetric matrix

$$A = \begin{bmatrix} 3 & 2 & 4 \\ 2 & 0 & 2 \\ 4 & 2 & 3 \end{bmatrix}$$

Using R demonstrate that all three eigenvectors of that matrix are mutually orthogonal. Let Λ be the matrix of eigenvectors of matrix A. Calculate product of tree matrices:

$\Lambda^T A \Lambda$

Symbol T indicates the transpose matrix. Google around for properties of eigenvectors and eigenvalues of real symmetric matrices. What is the general statement you can make about the observation on the value of the above product. Include copies of your R commands in your MS Word report.

Solution:

1. Calculate the product of three matrices $\Lambda^T A \Lambda$

```
> A <- matrix(c(3, 2, 4, 2, 0, 2, 4, 2, 3), nrow = 3, ncol = 3, byrow = TRUE)
> A
     [,1] [,2] [,3]
[1,]    3    2    4
[2,]    2    0    2
[3,]    4    2    3
> eigenA = eigen(A)
> eigenA
$values
[1]  8 -1 -1

$vectors
          [,1]       [,2]       [,3]
[1,] 0.6666667  0.7453560  0.0000000
[2,] 0.3333333 -0.2981424 -0.8944272
[3,] 0.6666667 -0.5962848  0.4472136
> eigenVector = eigenA$vectors
> t(eigenVector) %*% A %*% eigenVector
              [,1]          [,2]          [,3]
[1,] 8.000000e+00  2.664535e-15  1.332268e-15
[2,] 3.441691e-15 -1.000000e+00  2.220446e-16
[3,] 1.221245e-15  1.665335e-16 -1.000000e+00
> zapsmall(t(eigenVector) %*% A %*% eigenVector)
     [,1] [,2] [,3]
[1,]    8    0    0
[2,]    0   -1    0
[3,]    0    0   -1
> zapsmall(t(eigenVector) %*% eigenVector)
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
```

If all the eigenvalues of a symmetric matrix A are distinct, the matrix Λ, which has as its columns the corresponding eigenvectors, has the property that $\Lambda^T \Lambda = I$,

i.e., Λ is an orthogonal matrix (Eigenvectors corresponding to distinct eigenvalues are orthogonal.) The product of Λ^T A Λ is a diagonal matrix. The diagonal values of the product are the eigenvalues of A.