EE 360P: Concurrent and Distributed Systems **Vijay K. Garg**
Spring 2018 **TA: Tushar Chouhan , Asad Malik, Xiong Zheng, Eyenunwana Nwoko**

Assignment 4
Deadline: Tuesday, 27th March, 11 am

This assignment should be done in teams of two. The Jar and Java files have to be zipped into a file named `EID1_EID2.zip` and submitted through Canvas.

In this assignment, you will implement a text analyzer using Hadoop, which is a programming model and software framework for developing applications that concurrently process large scale data (Big Data) on distributed systems. The analyzer has to build a table that shows the occurrences of the words that appear together in the given text. The data set for this problem is the novel Pride and Prejudice. A collection of files (one per chapter) is provided in the attached zip file for the assignment. The occurrences are calculated in individual lines; your program does not need to intelligently separate sentences because the Mapper function is invoked for each newline by default.

Requirement:

- **Text:** Mr. Bingley was good-looking and gentlemanlike; he had a pleasant countenance, and easy, unaffected manners."
- **Occurrences:** For the word Bingley, "and" appears twice and "gentlemanlike" appears once. In this calculation, we say Bingley is a 'contextword'; "and" and "gentlemanlike" are 'querywords'. The other example: for the contextword and, "pleasant" appears once, "and" appears once (not zero times or twice)

For simplicity of this assignment and obtaining the same results, you should 1) convert every character into lower-case, 2) mask non-word characters by white-space; i.e., any character other than a-z, A-Z, or 0-9 should be replaced by a single-space character. For instance, Bingley's book" is converted into bingley s book". If the text of the novel is fully analyzed, we should get these results:

- 1. Darcy occurs 112 times with Bingley; Bingley is the contextword and Darcy is the queryword.
- 2. Elizabeth occurs 135 times with Jane; Jane is the contextword, and Elizabeth is the queryword.

The output file should contain the queryword and the count of its occurences for each context word in the below given format with each context word having the query word with maximum count appear first followed by other query words.

contextword1 max_count
<queryword1, occurrence>
<queryword2, occurrence>
contextword2 max_count
<queryword1, occurrence>
<queryword2, occurrence>

Quick Guide for Setting Up a Pseudo-Distributed Hadoop Cluster

Hadoop has two main modules: HDFS (Hadoop Distributed File System) and Yarn. The HDFS modules provides the storage for MapReduce tasks and the Yarn module manages the tasks and the resources for processing data in MapReduce framework. In the logical view, the HDFS module has two kinds of node: name node and data node. The name node accepts the requests for accessing/changing the data in the file system and data node performs the operations to the data. In Yarn module, the two nodes are resource manager and node manager. In an oversimplified view, the resource manager accepts the MapReduce tasks and distributes the tasks to different node manager. The node manager is a per-machine agent who monitors the resource usage of that machine.

In this guide, our machines have two roles: master and slave. The master machine (hadoopmaster) has the name node (as well as the secondary name node) and the resource manager started. The slave machines (hadoopslave) has a data node and a node manager started.

The following steps is a quick guide for setting up a pseudo distributed Hadoop cluster. For the details, you should visit: http://hadoop.apache.org/docs/current/index.html. DO NOT simply copy and paste the settings, you may need to replace the variables accordingly.

You can also try the multi node cluster using Amazon's AWS service the tutorial for which is available on:
`https://blog.insightdatascience.com/spinning-up-a-free-hadoop-cluster-step-by-step-c406d56bae42#.s7cmxssbq`
Just follow each step carefully and you will have a multi node cluster set up. In that case you do not need to do the below steps until after starting all the daemons.

1. We will use a dedicated Hadoop user account for running Hadoop.

   ```
   sudo addgroup hadoop
   sudo adduser --ingroup hadoop hduser
   sudo adduser hduser sudo
   ```

2. Install ssh : sudo apt-get install ssh

3. Passwordless ssh

   ```
   su hduser
   hduser@laptop:~$ ssh-keygen -t rsa -P ""
   hduser@laptop:/home/k$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
   We can check if ssh works:
   hduser@laptop:/home/k$ ssh localhost
   ```

4. Install Hadoop

   ```
   hduser@laptop:wget http://apache.mirrors.tds.net/hadoop/common/hadoop-2.7.1
   /hadoop-2.7.1.tar.gz -P ~/Downloads
   hduser@laptop:sudo tar zxvf ~/Downloads/hadoop-* -C /usr/local
   hduser@laptop:sudo mv /usr/local/hadoop-* /usr/local/hadoop
   hduser@laptop:sudo chown -R hduser:hadoop /usr/local/hadoop
   ```

5. Add the following lines to the .bashrc on all machines:

   ```
   # Set Hadoop-related environment variables
   export HADOOP_HOME=/usr/local/hadoop

   # Set JAVA_HOME (we will also configure JAVA_HOME directly for Hadoop later on)
   ```

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

# Some convenient aliases and functions for running Hadoop-related commands
unalias fs &> /dev/null
alias fs="hadoop fs"
unalias hls &> /dev/null
alias hls="fs -ls"

# If you have LZO compression enabled in your Hadoop cluster and
# compress job outputs with LZOP (not covered in this tutorial):
# Conveniently inspect an LZOP compressed file from the command
# line; run via:
#
# $ lzohead /hdfs/path/to/lzop/compressed/file.lzo
#
# Requires installed 'lzop' command.
#
lzohead () {
    hadoop fs -cat $1 | lzop -dc | head -1000 | less
}

# Add Hadoop bin/ directory to PATH
export PATH=$PATH:$HADOOP_HOME/bin
```

After adding those lines into your .bashrc, you can use the command “source .bashrc” or re-login to make the changes happen.

6. edit /usr/local/hadoop/etc/hadoop/hadoop-env.sh

```
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
 *(change it according to the java package that you download)
```

7. /usr/local/hadoop/etc/hadoop/core-site.xml:

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

Change the file core-site.xml:
<configuration>
 <property>
  <name>hadoop.tmp.dir</name>
  <value>/app/hadoop/tmp</value>
  <description>A base for other temporary directories.</description>
 </property>

 <property>
  <name>fs.default.name</name>
  <value>hdfs://localhost:54310</value>
  <description>The name of the default file system.  A URI whose
  scheme and authority determine the FileSystem implementation.  The
  uri's scheme determines the config property (fs.SCHEME.impl) naming
  the FileSystem implementation class.  The uri's authority is used to
  determine the host, port, etc. for a filesystem.</description>
 </property>
</configuration>
```

8. /usr/local/hadoop/etc/hadoop/mapred-site.xml.

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

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

edit the mapred-site.xml:
configuration>
 <property>
  <name>mapred.job.tracker</name>
  <value>localhost:54311</value>
  <description>The host and port that the MapReduce job tracker runs
  at.  If "local", then jobs are run in-process as a single map
  and reduce task.
  </description>
 </property>
</configuration>
```

9. /usr/local/hadoop/etc/hadoop/hdfs-site.xml

```
hduser@laptop:~$ sudo mkdir -p /usr/local/hadoop_store/hdfs/namenode
hduser@laptop:~$ sudo mkdir -p /usr/local/hadoop_store/hdfs/datanode
hduser@laptop:~$ sudo chown -R hduser:hadoop /usr/local/hadoop_store

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

<configuration>
 <property>
  <name>dfs.replication</name>
  <value>1</value>
  <description>Default block replication.
  The actual number of replications can be specified when the file is created.
  The default is used if replication is not specified in create time.
  </description>
 </property>
 <property>
   <name>dfs.namenode.name.dir</name>
   <value>file:/usr/local/hadoop_store/hdfs/namenode</value>
 </property>
 <property>
   <name>dfs.datanode.data.dir</name>
   <value>file:/usr/local/hadoop_store/hdfs/datanode</value>
 </property>
</configuration>
```

10. Format the new hadoop file system :

    `hduser@laptop:~$ hadoop namenode -format`

    If you see "`INFO util.ExitUtil:  Exiting with status 0`" in the last few lines, then you are good. If the returned status is 1, you can check the error message on the screen and fix the problem. Also before moving forrward check the cluster id is the same in datanode/current/VERSION and namenode/current/VERSION

11. Check everything is working

    (a) Check the log files in `$HADOOP_HOME/logs`.

    (b) The following webpage should be accessible:

    ```
    For the status of HDFS: http://hadoopmaster:50070/
    For the status of the Hadoop cluster: http://hadoopmaster:8088/cluster
    ```

    (c) Use the following command to start all the daemons:

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

    (d) Use "`jps`" command to list the Java daemons in the background. If all the processes are started successfully, you should see something similar to the follows:

    ```
    23465 Jps
    22540 ResourceManager
    16804 NameNode
    17056 SecondaryNameNode
    22837 NodeManager
    16927 DataNode
    ```

    Every line starts with a process ID and then followed by the process name.

12. Run the basic Hadoop example.

    ```
    mkdir input  # create a local folder
    vim input/file  # create and edit a file. Type down some words in the file.
    hdfs dfs -copyFromLocal input/ /input # copy the local directory to HDFS
    hdfs dfs -ls /input # list the files in the directory /input on HDFS
    hadoop jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-
        x.x.x.jar wordcount /input /output # run the famous hadoop example
    hdfs dfs -ls /output # you should see all the outputs
    hdfs dfs -copyToLocal /output # download the result from HDFS
    ```

13. Stop Hadoop

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

    Note: we show the most basic commands for starting and stopping Hadoop for teaching purpose; however, you may use other commands that are more convenient to do the same task.

Compile a Hadoop Application

Tutorial of developing a Hadoop application: http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html

1. Compile (the command "`hadoop classpath`" returns all the dependencies required by Hadoop):
   ```
   javac -classpath `hadoop classpath` -d build src/TextAnalyzer.java
   ```
   The backtick (`) symbol is the same key as the tilde (˜) and is located on the top-left of your keyboard.

2. Jar the Hadoop application: (You can remove the v in -cvf for a clean and quite output)
   ```
   jar -cvf TextAnalyzer.jar -C build/ .
   ```

Run a Hadoop Application

1. Make sure you have uploaded the input files onto HDFS.

2. Use the command to remove the output directory before the next run. Hadoop does not overwritten the existing output directory and it stops the task if there exists one.
   ```
   hdfs dfs -rm -r /output
   ```

3. Use the command to run your Hadoop application. The fourth field (which is TextAnalyzer in the example) is the class that contains the main method.
   ```
   hadoop jar TextAnalyzer.jar TextAnalyzer /input /output
   ```

Performance Evaluation

Besides the correctness of the results, your Hadoop application also needs to reduce the messages that are sent from mappers to reducers. At the end of each Hadoop job, the "Map-Reduce Framework" section of the on-screen information provides the number of messages between mappers and reduces.

If your mappers emit every pair of contextword and queryword, there should be 10,075,695 records sent to reducers. In this assignment, you need to reduce the number to 6,659,870 or 5,303,614. There are two ways to reduce the messages. One is using Combiners and the other is developing a better Mapper function. You can use both or either one of them in this assignment.

Useful Links

1. *MapReduce paper* – http://research.google.com/archive/mapreduce.html

2. *Hadoop* – http://hadoop.apache.org/

3. *Setup Hadoop* – http://hadoop.apache.org/docs/current/index.html

4. *Develop a Hadoop Application* – http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html