**EE 382N: Distributed Systems**
**Instructor: Professor Vijay Garg (email: garg@ece.utexas.edu)**
**TA: Changyong Hu (email: colinhu9@utexas.edu)**

**Assignment 3**
**Deadline**: Friday, 1st November

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. **All program for this assignment should be compatible with Java 8. Please put BOTH and ONLY the jar and java files in the folder.**

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 construct a weighted undirected graph 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. Your program needs to construct a weighted undirected graph $G(V, E, w)$ of all the words in the novel defined as follows. $V$ is the set of vertexes, $E$ is the set of edges, and $w$ is a weight function for each edge. Each word in the input text is a vertex in $G$, and the name of the vertex is the word it represents. For any two vertexes (words) $v_1$ and $v_2$, there is an edge between $v_1$ and $v_2$ if $v_1$ and $v_2$ appear together at a same line of the input text. The weight of edge $(v_1, v_2)$, denoted as $w(v_1, v_2)$, is defined as the number of lines word $v_1$ and $v_2$ appear together. Note that since $(v_1, v_2)$ and $(v_2, v_1)$ represent the same edge, we must have $w(v_1, v_2) = w(v_2, v_1)$. Consider the following example.

**Mr. Bingley was good-looking and gentlemanlike.**
**Mr. Bingley had a pleasant countenance, and easy, unaffected manners.**

For the text given as above, the vertex set of the graph is composed of all the words appearing in the text. Note that although "and" appears twice, there is a single vertex in the graph which represents "and". There is an edge between vertex "Bingley" and "gentlemanlike" with weight 1, ,i.e, $w$("Bingley", "gentlemanlike") = 1, since they appear together at the first line of the text. There is an edge between vertex "Bingley" and "and" with weight 2, i.e, $w$("Bingley","and") = 2, since they appear together in both lines. There is no edge between "was" and "had", since they appear in different lines.

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".

The output of your program is composed of each edge of the graph constructed from the input text along with its weight. Each edge has two entries in the output. For example, the edge between $word_1$ and $word_2$ should be outputed as follows:
$word_1$ $word_2$ $w(word_1, word_2)$
$word_2$ $word_1$ $w(word_2, word_1)$
Each line of the output corresponds to exactly one edge. The two vertexes and the weight of the edge should be seprated by any delimeter you prefer. A template is given in the github under the

hw4-template folder. The output corresponds the above example is also given under the template folder.

If the text of the novel is fully analyzed, one edge of the graph should be:

bingley darcy 87

## Hadoop Setup

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. In this assignment, you will run your program in a pseudo distributed Hadoop cluster. That is, the master and slave resident on a same machine. You do not need multiple matchines for this assignment. For the details, you should visit: http://hadoop.apache.org/docs/current/index.html.

We provide the following two options to set up a Hadoop environment. You are free to choose either one. For option I, we provide you a docker image which already has Hadoop environment set up. So, you only need to download docker and load the image. For option II, you need to set up Hadoop step by step in your local machine.

The instructions on how to compile and run your program on hdfs is given after the two options.

### OPTION I: Setting Up Hadoop using Docker

If you are not familiar with docker, you can find information on this site. `https://docs.docker.com/`

For this assignment, we provide a docker image with the environment set up. You do not have to install hadoop on you own if you choose this option. The steps to use our image are as follows:

1. Follow the steps and install Docker CE which is the free community version from this site. `https://docs.docker.com/`
2. Download the image provided by us from Google Drive.
   `https://drive.google.com/file/d/1iHPGD8-7hPniTzZY09iha8s9knt7ycAJ/view?usp=sharing`
3. Import the docker image using this command:
   ```
   $ sudo docker load --input hadoop.tar
   ```
4. Verify if it is imported using this command:
   ```
   $ sudo docker images
   ```
   And you should see a image like this:

```
REPOSITORY          TAG          IMAGE ID        CREATED      SIZE
ee360p/hadoop     latest     a0e7aecb0ab0    44 hours ago  2.56GB
```

5. Start a docker container using this command:

   ```
   $ sudo docker run -it ee360p/hadoop /home/bootstrap.sh -bash
   ```

   You should the following output and get the bash open

   ```
   Starting sshd:                                             [  OK  ]
   Starting namenodes on [294414232f14]
   294414232f14: starting namenode, logging to ...
   localhost: starting datanode, logging to ...
   Starting secondary namenodes [0.0.0.0]
   0.0.0.0: starting secondarynamenode, logging to ...
   starting yarn daemons
   starting resourcemanager, logging to ...
   localhost: starting nodemanager, logging to ...
   bash-4.1#
   ```

6. Now your hadoop environment is all set. You can follow the steps after 11(d) in the other option to get familiar with hadoop system. Do NOT run commands before 11(d).

7. Locate the working directory:

   ```
   bash-4.1# cd /home/EE360P/hw4/
   ```

   Inside that directory, you can see a input folder which contains all the input for this assignment. You can create your Java files and compile them here just like in the othe option.

### Problem shooting

1. If you cannot install docker, please make good use of Google.

2. If you cannot import the docker image, please come to us. We may be able to help.

3. If you mess up the docker environment and it is beyond repair, you can always close the current container and start a new one following the previous steps. Note that, any files (your program) you created within a docker container will be lost if you delete that container. So always backup your program.

4. If you accidentally close a container, you can find and restart it using these command:

   ```
   $ sudo docker ps -a
   ```

   And grab the *CONTAINER ID* from the output and restart the container and enter the container:

   ```
   $ sudo docker start CONTAINERID
   $ sudo docker exec -it CONTAINERID bash
   ```

5. If you find it hard to program inside the docker because you only have *vi* to edit files, you can program on your computer and copy the files in or out of the container. Details can be found here: `https://docs.docker.com/engine/reference/commandline/cp/`

## OPTION II: Set Up a Pseudo-Distributed Hadoop Cluster on Host

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:~$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
   We can check if ssh works:
   hduser@laptop:~$ ssh localhost
   ```

4. Install Hadoop

   ```
   hduser@laptop:wget http://mirrors.ocf.berkeley.edu/apache/hadoop/common/hadoop-3.1.2/
   hadoop-3.1.2.tar.gz
   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

   # 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:~$ hdfs namenode -format
    ```

11. Check everything is working

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

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

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

    (c) 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. 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.

## Run The Basic Hadoop Example

Before you move on to program for your assignment, you should run the following basic hadoop example to check if your environment is correctly set up.

```
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
```

## Compile a Hadoop Application

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

1. Go to the directory which contains your source files.
2. Compile (the command "`hadoop classpath`" returns all the dependencies required by Hadoop):
   `javac -classpath `hadoop classpath` *.java`
   The backtick (`) symbol is the same key as the tilde (~) and is located on the top-left of your keyboard.
3. Jar the Hadoop application: (You can remove the v in -cvf for a clean and quite output)
   `jar -cvf TextAnalyzer.jar .`

## Run a Hadoop Application

1. Make sure you have uploaded the input files onto HDFS. using the commond:
   `hdfs dfs -copyFromLocal input /input.`
   "input" is the local folder which contains all the input files. "/input" is the folder in the hdfs system.
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`
4. Copy results from the hdfs system to local using the command:
   `hdfs dfs -copyToLocal /output output`
   You program output is under the "output" directory.

## Useful Links

1. *MapReduce paper* – http://research.google.com/archive/mapreduce.html
2. *Hadoop* – http://hadoop.apache.org/
3. *Hadoop API* – https://hadoop.apache.org/docs/r2.7.4/api/
4. *Setup Hadoop* – http://hadoop.apache.org/docs/current/index.html
5. *Develop a Hadoop Application* – http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html