Assignment 3 – Exploring Hadoop: a Mapreduce Based Big Data Processing Framework

(Due by Midnight, Thursday, April. 16, 15% Grade)

---

1 Summary

Hadoop MapReduce [1] is a software framework for easily writing applications which process vast amounts of data in-parallel on large clusters of commodity hardware in a reliable, fault-tolerant manner. A MapReduce job usually splits the input data-set into independent chunks which are processed by the **map** tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the **reduce** tasks. Typically both the input and the output of the job are stored in a file-system (e.g., HDFS). The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks. In this Assignment, you will get familiar with the basics of the Hadoop framework with several tasks.

2 Install Hadoop

2.1 Experimental Environment

We will the Ubuntu Linux VM provided by Watson School, or any other Linux machine. Please make sure the Ubuntu version is **16.04 LTS** (that way, the TAs can help you with the configuration if you need help).

2.2 Hadoop Installation

We start this assignment using an existing hadoop container image.
(To learn more about the hadoop framework, please refer to [2] and [3] for manual installations. To set up a hadoop cluster, you will need to install JAVA and Hadoop binary (the mapreduce programming framework and HDFS file system), and configure the environment. The following container image will save our time for all these setups.)

1. Pull docker image:

```
$ sudo docker pull kiwenlau/hadoop:1.0
```

2. Clone the github repository:
(This repo includes several automation scripts for setting a hadoop cluster with the above container image. To find out how hadoop is installed and configured within a container, please refer to the "Dockerfile" file)

```
$ git clone https://github.com/kiwenlau/hadoop-cluster-docker
```

3. Create hadoop network (that later all hadoop containers will be connected):

```
$ sudo docker network create --driver=bridge hadoop
```

4. Start hadoop container clusters:
(You will start 3 containers with 1 master and 2 slaves. You will get into the /root directory of hadoop-master container):

```
$ cd hadoop-cluster-docker
$ sudo ./start-container.sh
```

5. Start hadoop clusters (You are in hadoop-master container.):

$ `./start-hadoop.sh`

3 Task I: WordCount — Hadoop's "Hello World'

We will get familiar with Hadoop by looking at its "Hello World" program — the "wordcount" program, which counts the number of occurrences of each word in a large collection of documents. To run this programm, we simply run the following script:

$ `./run-wordcount.sh`

Please read the above script and the output information carefully, and then answer the following questions:

- Q.1. Can you figure out what are the main steps do we need to run a hadoop mapreduce task (i.e., wordcount here)?
- Q.2 What does this command mean — "hdfs dfs -put ./input/* input"? (Hint, HDFS is Hadoop's distributed file system. Please refer to [4].)
- Q.3 How many mappers and reducers are launched for executing the above wordcount program?
- Q.4 How much time do mappers and reducers spend for the above tasks, separately?
- Q.5 After execution, what are the files in the output folder in HDFS, and what content do they contain?

4 Task II: Resizing the Hadoop Cluster

1. Exit your hadoop-master container and go back to the "hadoop-cluster-docker" repo. We are going to "re-size" our hadoop cluster using the following script. Please notice that it may take quite a while for resizing the hadoop cluster, as it needs to re-build/re-install the hadoop container image:

$ `sudo ./resize-cluster.sh 5`

2. Once the resizing process succeeds, start the hadoop containers — note that you have to **modify** the "start-container.sh" script to launch the correct number of containers:

$ `sudo ./start-container.sh`

Please answer the following questions:

- Q.6 How many master and slave containers do you launch separately this time?
- Q.7 Please figure out what a master container/node and a slave container/node are used for.

3. Start the hadoop clusters:

4. This time, we are going to add one more file for the wordcount program:

$ `wget http://cs.binghamton.edu/~huilu/text.txt`

5. Modify the "run-wordcount.sh" script to add the above "text.txt" to the input of the wordcount program so as to have three files (i.e., file1.txt, file2.txt and text.txt), and then execute it:

```
$ ./run-wordcount.sh
```

Please answer the following questions:

- Q.8 How many mappers and reducers are launched for executing the above wordcount program?
- Q.9 How much time do mappers and reducers spend for the above tasks, separately?
- Q.10 What are the two most frequently occurring words, and how many times do they occur?

5 Task III: Read the Source Code of the WordCount Program

Assume you are in the hadoop master container, download the source code of the WordCount program:

```
$ wget http://cs.binghamton.edu/~huilu/WordCount.java
```

From the source code, you will read about that, basically the programmer simply needs to provide two functions, one is the map function and the other is the reduce function. The hadoop framework hides most of the details from the developers such as creating, launching, and scheduling jobs. The map (and reduce) function reads data from the distributed file system (i.e., HDFS), and writes back intermediate (or final) results back to the file system (i.e., HDFS). For big data processing, during execution, the hadoop framework also takes care of failures — once one node fails, it needs to recover the job execution of that node on another healthy node.

Note that, the input to the reducer is the sorted output of the mappers. The hadoop framework also groups reducer inputs by keys. Based on this, please read the source code and answer the following questions.

- Q.11 Please describe the basic steps in the map function of `WordCount.java`.
- Q.12 Please describe the basic steps in the reduce function of `WordCount.java`.

Let's compile the above WordCount program:

```
$ export HADOOP_CLASSPATH=$JAVA_HOME/lib/tools.jar
$ hadoop com.sun.tools.javac.Main WordCount.java
$ jar cf wc.jar WordCount*.class
```

Let's execute the compiled WordCount program. Put the following first line in the proper place of the "run-wordcount.sh" and execute it (please follow the default run-wordcount.sh script with two files as the input):

```
$ hadoop jar wc.jar WordCount input output
$ ./run-wordcount.sh
```

Please answer the following questions:

- Q.13 How many mappers and reducers are launched for executing the above wordcount program?
- Q.14 How much time do mappers and reducers spend for the above tasks, separately?

6 Task IV: Write Your Own Simple Hadoop Program

In this task, we are going to write a variant of the WordCount program. Instead of counting the number of the occurrences of each word from the input files, we only focus on a specific word. Particularly, we would like to only count the occurrences of word "`and`". Please write your Hadoop map and reduce functions to realize this. For example, the output will look like (if we use the "text.txt" file as the input):

```
$ wordcount output:
$ and 18
```

7 Task V: Other Hadoop Use Cases

Please come up with your hadoop use case or find (through Internet) another use case for data processing. Write your program (or convert an existing hadoop program) and execute it in our container-based hadoop clusters. Please demo your use case.

8 Submission & Grading

Submit your assignment on the blackboard as one **tar-gzipped** file (generated using the tar command with cvzf options). In the tar-gzipped file, include all your (java) code, a README file, and a PDF report with all answers to the questions in all tasks. If you do not know how to do this, please contact us for help. DO NOT submit each file individually – include only the files you change. Make sure your code can run without errors. If the code cannot run, you will get 0 points.

- Task I: Answer all the questions (Q.1 to Q.5) in the report. – 25 points.
- Task II: Answer all the questions (Q.6 to Q.10) in the report. – 25 points.
- Task III: Answer all the questions (Q.11 to Q.14) in the report – 20 points.

In addition, you will also schedule a Zoom appointment with Professor (not TA this time) to have a demo before the deadline. **Notice that, we will release the time slots for demo. It is your responsibility to schedule the demo time. You will lose points if you fail to do so**. We will grade your assignment based this report and the demo.

- Task II: Demo (Q.8 to Q.10) – 5 points.
- Task III: Demo (Q.13 to Q.14) – 5 points.
- Task IV: Demo – 10 points.
- Task V: Demo – 10 points.

References

[1] Apache Hadoop. `https://hadoop.apache.org/`.

[2] Hadoop: Setting up a Single Node Cluster. `http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html`.

[3] Hadoop Cluster Setup. http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html.

[4] HDFS Architecture Guide. https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html.