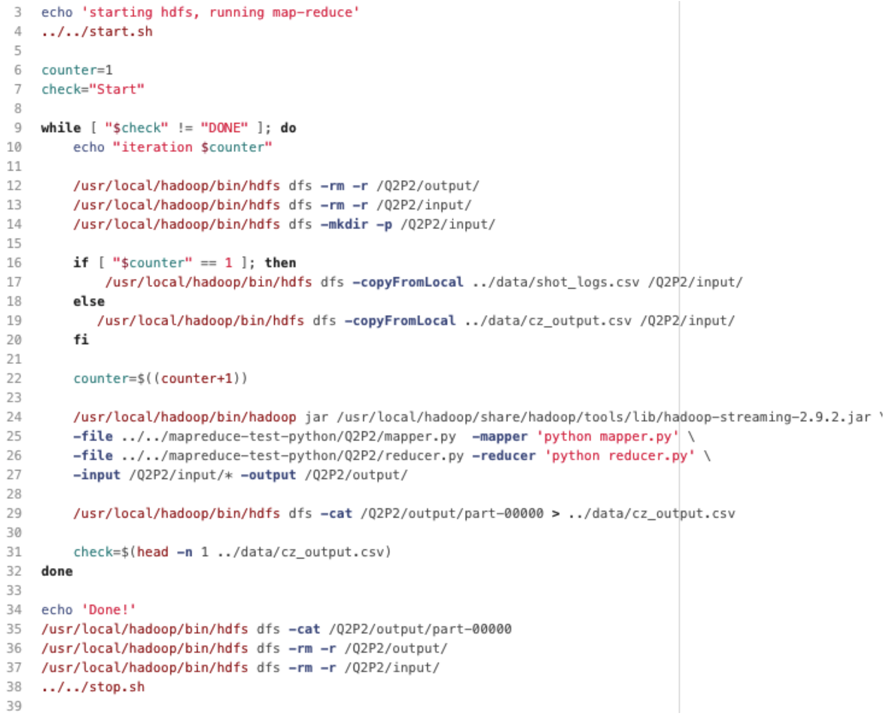
CISC 5950 — Lab 2

Congratulations on successfully completing the project 1. At the current stage, you should feel comfortable to read and write MapReduce based programs.

In the project 1, you developed a MapReduce based K-Means classifier. You should have observed that developing an interative program, which involves multiple Maps and Reduces, with MapReduce programming framework is definitely not a trivial task. The have to use a loop in the shell program to start the iteration and utilize an indicator to stop the loop (Fig. 1).



```
echo 'starting hdfs, running map-reduce'
../../start.sh

counter=1
check="Start"

while [ "$check" != "DONE" ]; do
    echo "iteration $counter"

    /usr/local/hadoop/bin/hdfs dfs -rm -r /Q2P2/output/
    /usr/local/hadoop/bin/hdfs dfs -rm -r /Q2P2/input/
    /usr/local/hadoop/bin/hdfs dfs -mkdir -p /Q2P2/input/

    if [ "$counter" == 1 ]; then
        /usr/local/hadoop/bin/hdfs dfs -copyFromLocal ../data/shot_logs.csv /Q2P2/input/
    else
       /usr/local/hadoop/bin/hdfs dfs -copyFromLocal ../data/cz_output.csv /Q2P2/input/
    fi

    counter=$((counter+1))

    /usr/local/hadoop/bin/hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.2.jar \
    -file ../../mapreduce-test-python/Q2P2/mapper.py  -mapper 'python mapper.py' \
    -file ../../mapreduce-test-python/Q2P2/reducer.py -reducer 'python reducer.py' \
    -input /Q2P2/input/* -output /Q2P2/output/

    /usr/local/hadoop/bin/hdfs dfs -cat /Q2P2/output/part-00000 > ../data/cz_output.csv

    check=$(head -n 1 ../data/cz_output.csv)
done

echo 'Done!'
/usr/local/hadoop/bin/hdfs dfs -cat /Q2P2/output/part-00000
/usr/local/hadoop/bin/hdfs dfs -rm -r /Q2P2/output/
/usr/local/hadoop/bin/hdfs dfs -rm -r /Q2P2/input/
../../stop.sh
```

Figure 1: MapReduce based interative programming

This challenge is caused by the fact that Hadoop is design to utilize the storage space in the cluster. However, each MapReduce program requires to output the data into the hard drive. The feature leads to a large amount of read/write of HDFS, which significantly limits the performance.

Spark Programming

The spark system implements the Resilient Distributed Dataset (RDD) to maximize the memory space in the cluster. With RDD, most of the operation is done in the memory (Fig. 2).

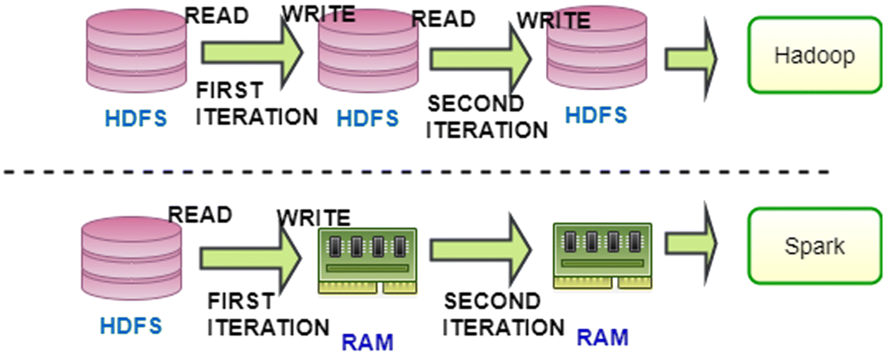


Figure 2: Hadoop v.s. Spark

To develop a K-Means algorithm in spark, you just need to transform the previous RDD into a new one for the next interation.

Programming in Lab 2

In this lab, please, based on your previous code, inplement the K-Means algorithm, you can use any spark related library package.

1. Please redo Project 1 Part 2 Question 2 (Part 1).
2. Please redo Project 1 Bonus Question 1(Part 2).
3. Please redo Project 1 Part 1 Question 1 with different levels of parallelism, 2, 3, 4, 5 with (Part 3, just change one line in the test.sh file, add –conf spark.default.parallelism=2 after spark-submit to set parallelism level to 2).

Installing the spark cluster GitHub Link.

Grading Rubric

You should complete the lab in groups of 2 students.

(50%) Part 1;
(20% * 2) Part 2 and 3.
(10%) Report;

Submission

Please submit your code, readme files and a report to Blackboard.

Submission Format: In a zip/tar file, you should have 3 folders for each of the part and one report for the whole lab.