Program Assignment 3

HAO LUO
RUID: 162006265

Path:

S3 BUcket:	s3://hldata539
wordcount_largedata	s3://hldata539/word_large
chess_smalldata	partA: s3://hldata539/chess_small/chess_out1
	partB: s3://hldata539/chess_small/chess_out2
	partC: s3://hldata539/chess_small/chess_out3
chess_smalldata	partA: s3://hldata539/chess_large/chessout1
	partB: s3://hldata539/chess_large/chessout2
	partC: s3://hldata539/chess_large/chessout3

Result of chess_small data partA :

Black 9559 0.38
Draw 4582 0.18
White 11028 0.44

Result of chess_large data partA :

Black 13187704 0.47
Draw 1074045 0.04
White 14005748 0.5

Description of work:

To do this assignment, first I install Hadoop on my local computer, set up a Single Node Cluster, run the wordcount example on my local computer. Then I follow the instruction of aws, get familiar with the environment. Test the wordcount program on large dataset.And I modified the wordcount program to count the “win, lose, draw” in part A.

And here is the work of certain parts.

In part A, my program works well on local environment. But get the infinit number on aws. I realize that in large dataset, reduce works before the end of map. So the count/sum will lead to overflow. So I make two job client, one to count the sum, one to write the result.

In part B, I use string to store the number of win, lose and draw, like playerA-white numof win-numoflose-numofdraw. I read the data set, choosing ”{ }” to separate each game.

In part C, I use 3 job client. First to get the sum, second to get the frequency and the last one sort the frequency result.

Description of code:

The program of partA is MatchCount.java with the corresponding jar is mc.jar.

The program of partB is PlayerCount.java with the corresponding jar is pc.jar.

The program of partC is FrequencySort.java with the corresponding jar is fs.jar.

And when use mc.jar. Its class is MatchCount.class. Also it needs input path and output path. The output path should not be already existed.

And when use pc.jar. Its class is PlayerCount.class. Also it needs input path and output path. The output path should not be already existed.

And when use fs.jar. Its class is FrequencySort.class. it needs 3 paths. An input path, an output path and a temporary path to store the data . The output and temporary path should not be already existed.

When calculating the sum of counts, mapreduce still need output path to complete one job.So I use “Temp, temp11” as folder name, if they affect the running of program , delete these folders.

Use of more clusters:

When use 3 node one job needs almost 17-20 minutes. Increase more nodes will decrease the time. And the number of output files is proportional to the number of cluster node.