Programming Assignment 3
Due Wednesday December 16th, 2015 11:59pm
Hadoop and MapReduce (20 points)

The goal of this project is to get hands-on experience with Hadoop and MapReduce. You will run MapReduce jobs on the Amazon Hadoop EC2 clusters and report on the results.

What you should do:

1. Install hadoop locally and follow the Wordcount tutorial (see document on Sakai)
2. Follow the Elastic MapReduce tutorial given in the Sakai Programming Assignment Resources, this will show you how to run MapReduce tasks in the cloud.
3. Implement MapReduce tasks that on the Chess dataset.

Your results should be in a s3 bucket. You will need to give us to see the content of the bucket, otherwise we will not be able to see your results. See the file with the detailed instructions on how to set permissions on Sakai.

Chess Datasets:

Most of the work (except for the wordcount tutorial) will be run on Chess datasets.

1. The data is in PGN format, which is a notation standard for chess. Description of the format is in the file gnspecs.txt on Sakai.
2. The Sakai folder Chess Test Data contains three small PGN files (6-7Mb) on which you can test your data.
3. You will have to run your mapReduce jobs on large PGN files that reside on my S3 bucket `s3n://chessdatasets/`. The bucket contains several directories with various sizes of data sets for testing. SmallChessInput is the same as the one on Sakai. VeryLargeChessInput has 22Gb of data, in 22 files of around 1Gb each.
4. I downloaded the files from http://www.ficsgames.org/ in case you are interested in retrieving more data.
5. You can use an existing library to parse PGN data. I found a few online, such as Chesspresso (http://sourceforge.net/projects/chesspresso/files/chesspresso-lib/0.9.2/), which unfortunately has scant documentation. If you use an existing library, you should cite it. Since the parsing needed for this project is basic, you can also implement your own parser.

MapReduce Tasks on Chess Data:

You will have to implement the following MapReduce tasks:

A. Return the total number of games won by White, Black, and that ended in a draw, as well as what percentage of the total games this represent. The result should look like
Black 100 0.5
White 90 0.45
Draw 10 0.05

B. For each player, return the percentage number of games they won, lost or drew when playing White and Black. The result should look like
Player1id White0.5 0.4 0.1
Player1id Black0.55 0.4 0.05
Player2id White 0.35 0.65 0
Player2id Black0.3 0.5 0.2

C. We are trying to figure out how long (in terms of number of moves), games last. For this part, you can assume that the number of moves is given in the tag Plycount. Return the length of games sorted by frequency in the data set. The result should look like:
45 7%
32 6.9%
110 6.7%
...
Sorted by game length frequency

Graduate students should use a combine function when possible to get full credit. Undergraduate students may add a combine function for extra credit.

What to submit:

1. The **path** to your s3 bucket which contains your output. The TA and instructor should have permission to read the bucket. If we do not, you will not get credit for your work. A **Readme** file which clearly identifies the paths where we can find all the results, as well as contains the results for task A (part b and c, see below). It should also have a brief documentation of your code and description of what you did. **(2 points)**
2. The **output** of running the wordcount example from the tutorial on the EC2 cluster on the input directory `s3://worddatasets/WordCountInput` Note that this is a different input directory from the one used in the tutorial. The results should be in your S3 output path. The specific output path should be clear from your documentation**(3 points)**
3. The **output** of the Chess MapReduce tasks (A, B and C) both on the small data set (running the task either locally or on EC2) and on the VeryLargeChessInput Data set on

Amazon S3:
`s3://chessdatasets/VeryLargeChessInput`
For each task you should submit:

a. The java file you wrote and the corresponding jar files you created**(2 points each)**
b. The result of the task on the small data set (either submit the output files, or submit the s3 output path, clearly labeled)created **(1 points each)**
c. The output path on s3 for the result over the large outputcreated **(2 points each)**

(for Task A, since the outputs is small, please submit the values output in steps b and c it in your documentation)

Extra Credit (all):

To get extra credit, you should try different numbers of clusters and study on the impact of the number of clusters on the processing time of the MapReduce job on the VeryLargeChessInput. Extra credit will depend on the depth of the analysis.