Big Data Programming Project 2: League of Legends Match

Krish Moodbidri
University of Alabama at Birmingham
krish94@uab.edu

www.krishm.com

ABSTRACT

This is a project based report for the course Big Data programming (FA2017 CS 716-7R / 616-7R / 416-7R) taught by Dr. Jeremy Blackburn at University of Alabama, Birmingham.

The tasks were to analyze and summarize the patterns of the League of Legends, a popular multi player game. The dataset provided contained 41.5K matches data.

The summary of the tools used, the data analysis performed, the code and executing the jobs, is mentioned in this paper. Finally the analysis to the tasks given are performed and the results summarized.

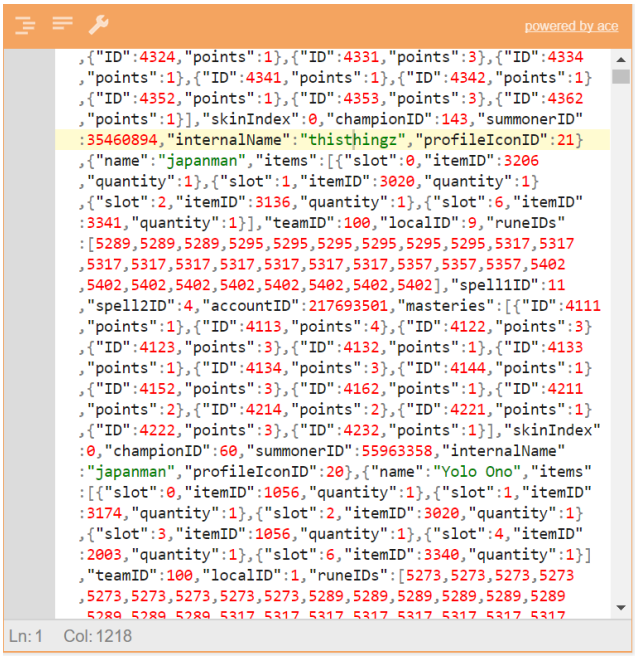
1 INTRODUCTION

League of Legends is a multiplayer online battle arena video game developed and published by Riot Games for Microsoft Windows and macOS. The game follows a freemium model and is supported by microtransactions, and was inspired by the Warcraft III: The Frozen Throne mod, Defense of the Ancients.

In League of Legends, players assume the role of an unseen "summoner" that controls a "champion" with unique abilities and battle against a team of other players or computer-controlled champions. The goal is usually to destroy the opposing team's "nexus", a structure which lies at the heart of a base protected by defensive structures, although other distinct game modes exist as well. Each League of Legends match is discrete, with all champions starting off fairly weak but increasing in strength by accumulating items and experience over the course of the game. The champions and setting blend a variety of elements, including high fantasy, steampunk, and Lovecraftian horror. *(ref. Wikipedia)*

About the Data:

The data provided was a JSON file having various fields associated with the players, events, etc. throughout the game /match. The data provided had 41.5k records of such matches.

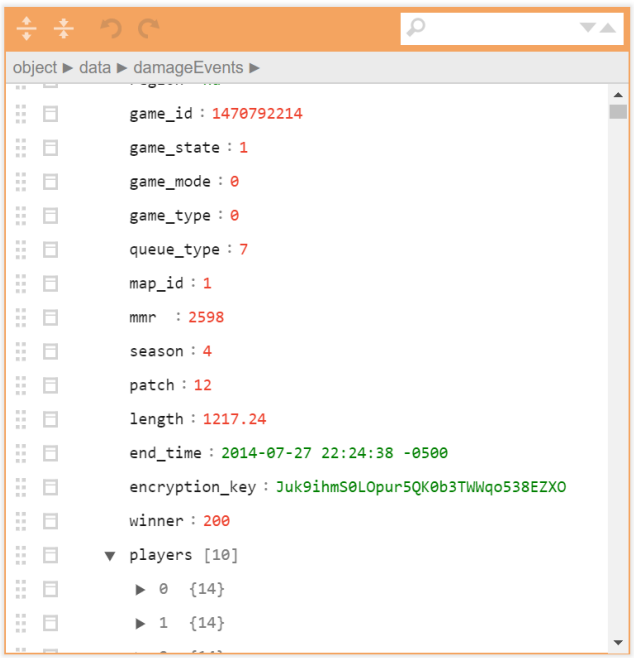


I pulled one of the JSON record and created a tree view from http://jsoneditoronline.org/ website. The result was as shown in the image below.

The tree view helped me visualize the data and understand the fields associated. As given in the project description document

Some examples of the meta-data:

- region - the region the game was played in (e.g., NA for North America)
- game id - a unique game identifier



2 DESIGN AND ARCHITECTURE

2.1 Platform Introduction

Apache Hadoop is an open-source software framework used for distributed storage and processing of dataset of big data using the MapReduce programming model. It consists of computer clusters built from commodity hardware. All the modules in Hadoop are designed with a fundamental assumption that hardware failures are common occurrences and should be automatically handled by the framework.

The core of Apache Hadoop consists of a storage part, known as Hadoop Distributed File System (HDFS), and a processing part which is a MapReduce programming model. Hadoop splits files into large blocks and distributes them across nodes in a cluster. It then transfers packaged code into nodes to process the data in parallel. This approach takes advantage of data locality,[3] where nodes manipulate the data they have access to. This allows the dataset to be processed faster and more efficiently than it would be in a more conventional supercomputer architecture that relies on a parallel file system where computation and data are distributed via high-speed networking. *(ref. Wikipedia)*

2.2 Platform Architecture

Hadoop employs a master/slave architecture for both distributed storage and distributed computation".In the distributed storage, the NameNode is the master and the DataNodes are the slaves. In the distributed computation, the Jobtracker is the master and the Tasktrackers are the slaves which are explained in the following sections.

MapReduce Job Processing

An entire Hadoop execution of a client request is called a job. Users can submit job requests to the Hadoop framework, and the framework processes the jobs. Before the framework can process a job, the user must specify the following:

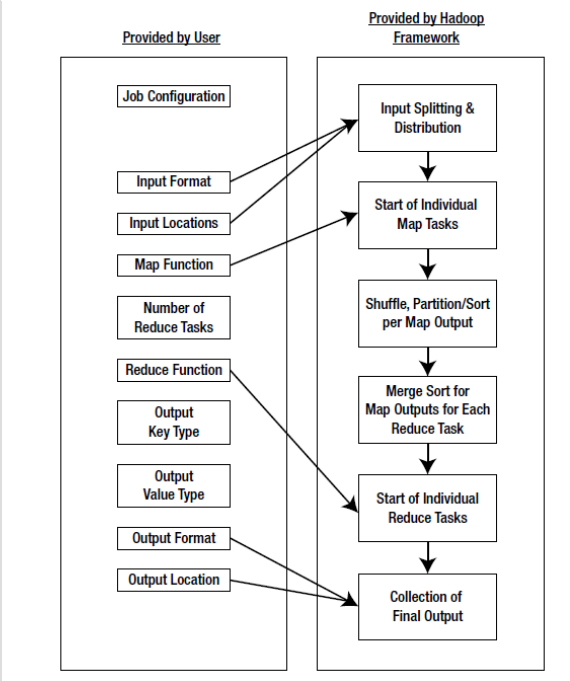
- The location of the input and output files in the distributed file system
- The input and output formats
- The classes containing the map and reduce functions

Hadoop has four entities involved in the processing of a job:[1]

- The user, who submits the job and specifies the configuration.
- Hadoop architecture The JobTracker, a program which coordinates and manages the jobs. It accepts job submissions from users, provides job monitoring and control, and manages the distribution of tasks in a job to the TaskTracker nodes.[2] Usually there is one JobTracker per cluster.
- The TaskTrackers manage the tasks in the process, such as the map task, the reduce task, etc. There can be one or more TaskTracker processes per node in a cluster.
- The distributed file system, such as HDFS.

The user specifies the job configuration by setting different parameters specific to the job. The user also specifies the number of reducer tasks and the reduce function. The user also has to specify the format of the input, and the locations of the input. The Hadoop framework uses this information to split of the input into several pieces. Each input piece is fed into a user-defined map function. The map tasks process the input data and emit intermediate data. The output of the map phase is sorted and a default or custom partitioning may be applied on the intermediate data. Accordingly, the reduce function processes the data in each

partition and merges the intermediate values or performs a user-specified function. The user is expected to specify the types of the output key and the output value of the map and reduce functions. The output of the reduce function is collected to the output files on the disk by the Hadoop framework. *(ref.* https://hadooptutorial.wikispaces.com/Hadoop+architecture *)*



Parts of a MapReduce job

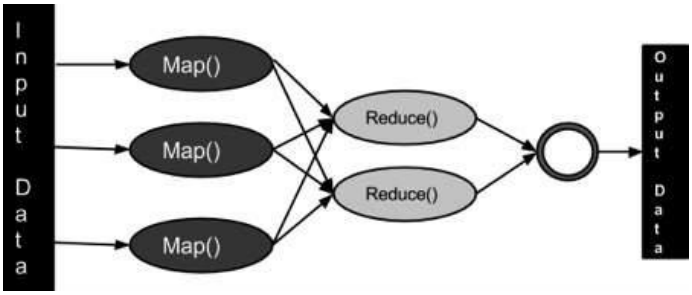
Map-Reduce:

- MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.

Map stage : The map or mapper's job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.

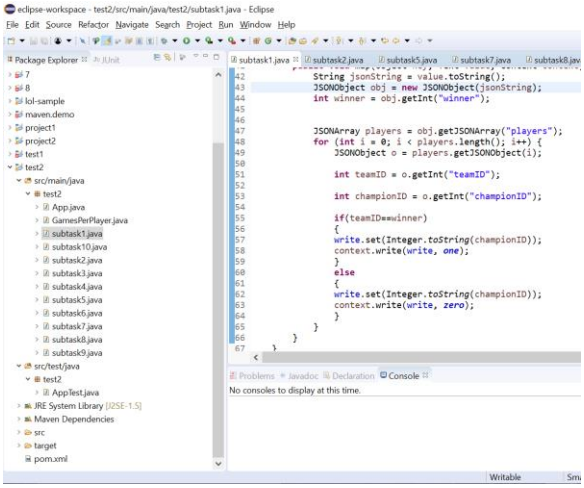
Reduce stage : This stage is the combination of the Shufflestage and the Reduce stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.

- During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.
- The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.
- Most of the computing takes place on nodes with data on local disks that reduces the network traffic.
- After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.



Writing code for this problem:

I coded in java, using Eclipse Oxygen as my IDE. I used the Maven project and used maven to convert the code to a jar file and execute the executable jar on the cluster.

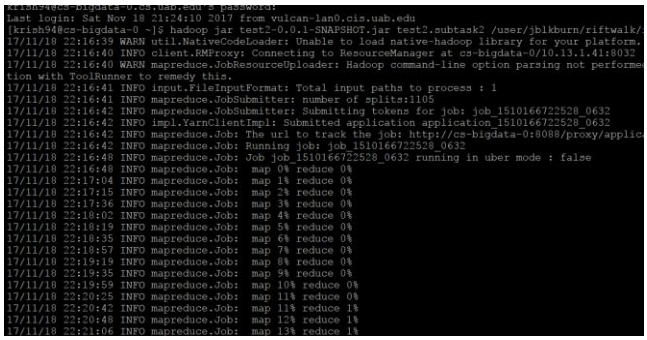


The above is a snippet of the code, IDE and platform.

Executing the Code:

I executed my code on the cs-bigdata-0.cs.uab.edu

This cluster was set up by Dr. Blackburn and Greg Bowersock for the course Big data Programming

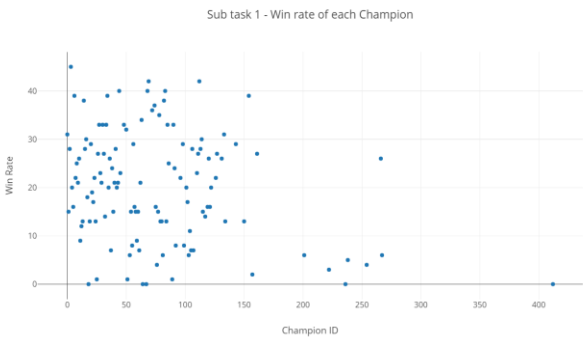


3 EXECUTION STRATEGY

I started by understanding the format of the data and did so by first converting the JSON data to a tree structure (using an online tool – link in the ref.). This helped me understand the attributes and logically differentiate the relevant ones for the ones not necessarily needed for the project and also the structure of the necessary data.

Task 1: Calculate win rate

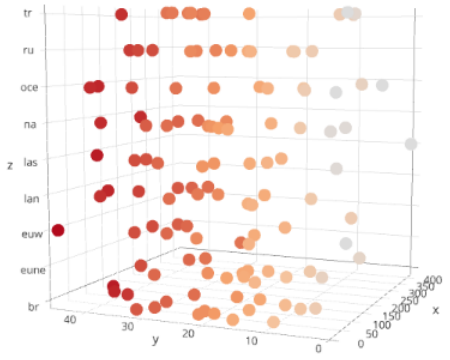
- I did so by finding the winning team-> figuring out the championID associated with each of the 10 players of the game -> checking team association of each of the champion.
- I have assumed that one team wins, and the other loses. Haven't considered the case of a tie.
- Also, I haven't considered the banned champions, as I couldn't relate if they're association with the game is banned during or for future games, like play again with same group.

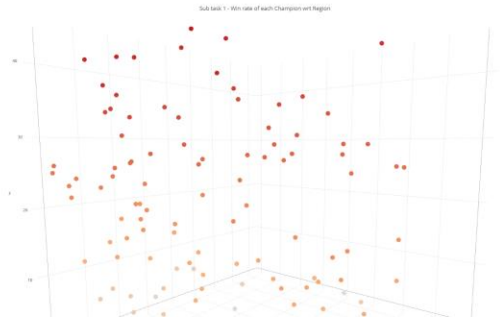


- The above scatter plot graph was plotted using Plot.ly

Task 2: Do win-rates vary between regions?

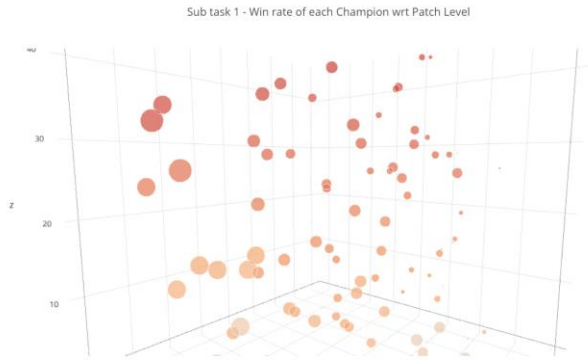
- I did so by extending the previous code and adding the region variable to it. I appended the region and the championID, thus enabling the reduce function to merge the results and plotted them using plot.ly
- I didn't do any data filling for regions having N/A as the value. Thus the output determination is an estimation having error = games having region = na/ 41.5k

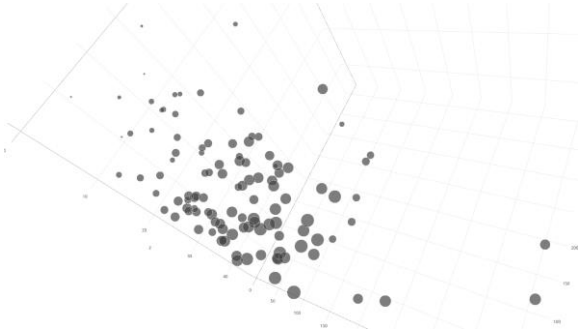


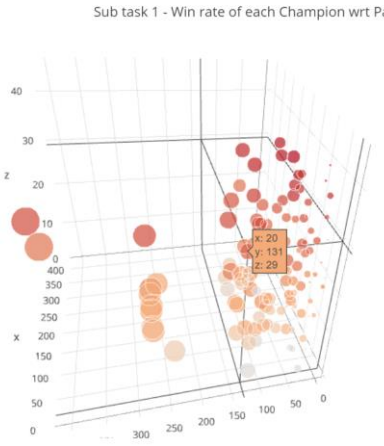


Task 3: Do win-rates vary based on the patch level?

- I pretty much used the previous code by replacing the region with patch field to analyze and estimate the answer.
- Since patch level is common for an entire game, the association wrt each player wasn't quite clear.

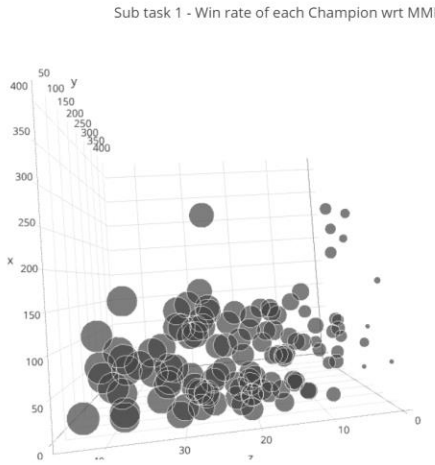






Task 4: Do win-rates vary based on mmr?

- I pretty much used the previous code by replacing the region with mmr field to analyze and estimate the answer.

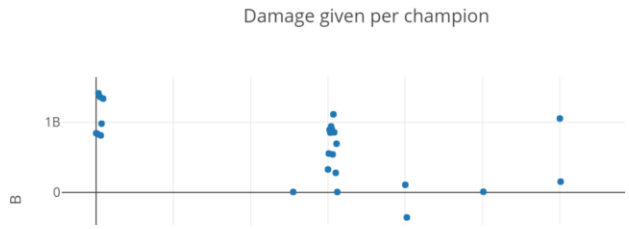


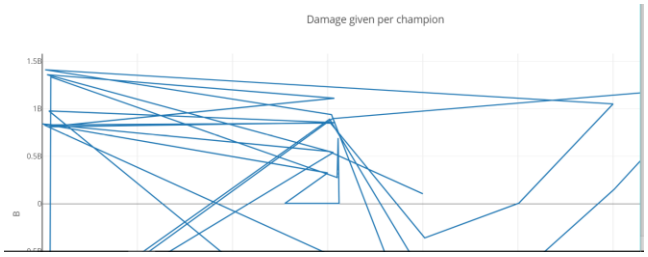
Task 5: Compute damage done per player

- Damage events had the following fields associated


```
▼ damageEvents [19272]
  ▼ 0 {5}
      time : 93.23726
      receiverUnitID : 500
      dealerUnitID : 500
      damage : 4000
      type : 36
```

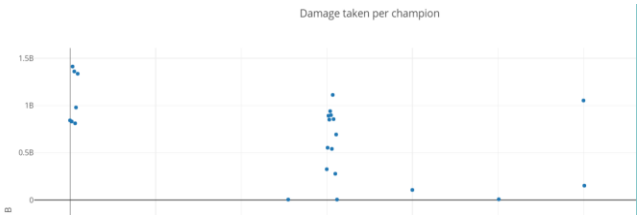

- The formula I used was damage done per champ = damage * type/time.
- There seemed to be an inverse relation between the time and type field.

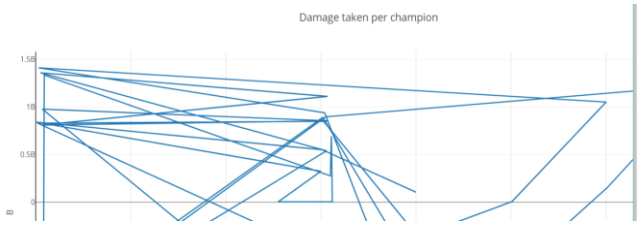




Task 6: Compute damage taken per player

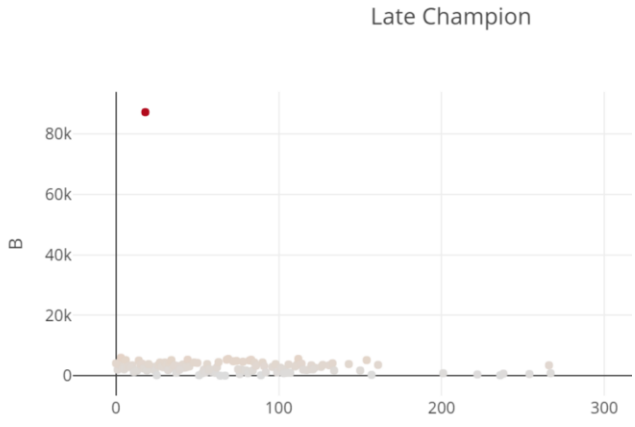
- This was pretty similar to the previous task with the difference of the tag considered

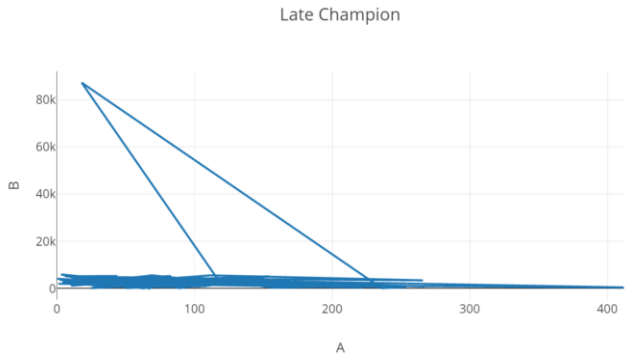




Task 7: Can you determine which champions are “late game" champions? I.e., they are relatively weak early and relatively powerful the longer the match goes.

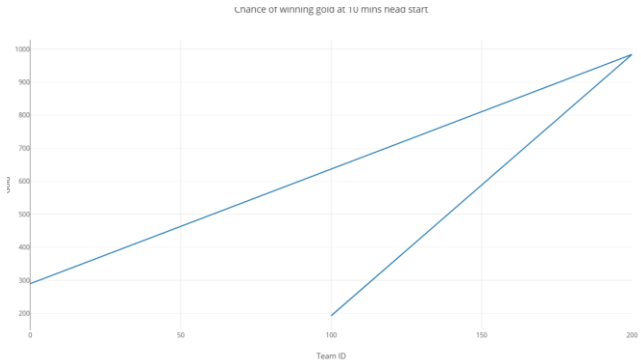
- I’m not quite sure if my code for this problem is good enough. I found the max each champion was alive during the game (which I guess is a late champion) and did a reduce on the same.





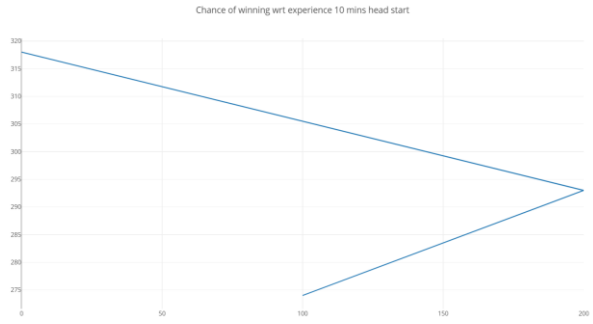
Task 8: What is the probability of winning a match if a team is ahead with respect to gold at 5 minutes, 10 minutes, or 15 minutes?

- I checked the gold earned by each champion and reduced this wrt the team that won or lost



Task 9: Compute damage taken per player

- I used the same code as the previous problem just replacing the gold earned with experience



Task 10: Do something with this positional data.

- I'm still working on this as I write my report.

5 CONCLUSIONS

I learnt a lot during the project. The data set seemed really interesting and the data association as well. I would specially thank 3 people – Dr. Blackburn, Amalee and Mashuir for their help and support. It took me quite some time to figure out the system and dataset. Though I did my best on the project, I would like to work on the error cases for each of the questions to determine a more accurate analysis. Overall, it was quite a fun and educational project.

REFERENCES

[1] http://jsoneditoronline.org/
[2] https://www.tutorialspoint.com/hadoop/hadoop_mapreduce.htm
[3] https://hadoop.apache.org/docs/r2.8.0/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html
[4] https://stackoverflow.com/questions/44019051/how-to-parse-json-file-in-hadoop
[5] https://community.hortonworks.com/questions/78882/store-and-process-json-files.html
[6] https://chandramanitiwary.wordpress.com/category/mapreduce/

I read in a journal while doing my research to multiply the results by 100 and divide it late once the output is b=obtained. This helps in more accurate results. I haven't done a lot of research on this, but have used this concept for task 1,2,3 and 4.

I pulled the results and divided them by 100 on excel using excel commands

I also copied a lot of information about hadoop, LoL game and map reduce from the internet for my report.

All the work submitted was done solely by me and due to time restrictions, i couldn't complete task 10 accurately.

Apologies for any grammatical or written English errors. I am still working on my English (I studied in another language back in India).