

Mini Project 2 Report

Introduction

For this practical we had to use the Hadoop Framework to process large amounts of data. We had to get used to the MapReduce programming model and the idea of horizontal scalability. In addition we had to utilize Hadoop to read twitter data, which is stored in JSON.

The second part of the practical involved testing how varying the number of mappers and threads per mapper, affected the runtime of a particular job.

Usage Instructions Extensions

Each of the Extensions is listed on the Menu UI for testing.

1. Basic Requirements, hashtag collecting
2. Sorting the results
3. Get English hashtags only.
4. Get Most retweeted tweets.
5. Get most Retweeted users.
6. Delete the output directory.

After each of the above tasks you must delete the output directory (using option 6 or file explorer) before you can complete another task, unless you restart the program with a different output path.

Sorting

The sort will always occur on the output directory provided. If you wish to test the sorting ensure that in

```
java -cp "lib/*:bin" ProjectMain <input_path> <output_path>
```

that <output_path> contains a file part-r-00000 with the data to be sorted. The output of the sort will be in <output_path>/sorted/...

Design

Basic Requirements

ProjectMain

This method checks that the input and paths necessary for Hadoop to run have been provided, and calls a textUI class for the user to select any additional options they want, such as getting the top 40 most used hashtags.

In order to gather data from the palain host server I made a set of for loops that can be seen commented out.

MenuUI

Provides a simple Text UI for the user to interact with.

TweetsCount

This class configured all the details necessary for the job to run according to the specifications required. I overloaded the mapReduceHashtags method so that when testing I could specify exactly how many mappers and threads I wanted to run.

Each separate job had a method associated with it (i.e. finding most retweeted users had one method that would initialize the correct mapper and reducer for the task). I had each job use 12 mappers simply to increase performance.

ScanTweetsMapper

The twitter API tells us to navigate to the entities object which has an array of hashtags that a particular tweet has used. This made it simple for any particular mapper to quickly retrieve hashtags from a tweet.

If the tweet is retweeting someone then it will contain that other persons tweet. This adds the problem that the inner tweet will have hashtags of its own. I opted to ignore this aspect for simplicity as if I considered this retweet object, I would have to make sure multiple users aren't retweeting the same tweet and multiplying the appearance of a particular hashtag.

Experiment

In order to test the performance of the program, I chose to examine an hour of twitter data as this gives a suitably large dataset to ensure that the time taken was processing rather than the setup time for the job. I tested up to 12 mappers with up to 6 concurrent threads each. I repeated each job 10 times and took an average in order to reduce the effects of noise on any results. I then repeated the test (with only 4 threads maximum) for 2 hours of twitter data. Then

All testing was conducted on the palain host server, which has 24 processors.

Extensions

Sorting

The simplest method for sorting is to perform a second MapReduce on a list. The mapping step would be to swap the key's and value's positions, then using the inbuilt functionality in Hadoop to sort the new keys (the old values representing how many times the key has appeared) in descending order.

The reducer then produces a list with two columns. The left column is a number representing how many times the object on the right column has appeared. Using [Figure 4](#), as an example, the left side represents how many times a user has been retweeted and the right side is the list of users have been retweeted that many times.

Most Retweeted Tweets

RetweetsMapper

The purpose of RetweetsMapper and RetweetsReducer is to get the most retweeted tweets from a dataset. The Twitter API tells us that if a tweet is a retweet, it contains that retweeted object within itself. So in order to find the most retweeted tweets one must both check if the tweet *is a* retweet, and if the provided tweet *has been* retweeted.

I chose to use the tweet id_str as the key because it uniquely identifies a tweet and it is easy to check the tweet by going to https://twitter.com/statuses/<id_str>.

RetweetsReducer

The reducer receives a tweet id and a list of retweet counts. However since the retweet counts have already been aggregated I do not sum the values. As twitter data can change over time, the most accurate retweet count for a tweet would be from the data that comes latest. However as it would be very complicated to implement a MapReduce that takes the time of tweet into account when reducing the data, I opted to instead choose the highest retweet count for simplicity as I believe it is unlikely for a large number of users to un-retweet someone.

Most Retweeted User

MostRetweetUserMapper

In order to get accurate results for this, I made the key the user id of the tweet and I had to make the output value a composite of the tweet's id and the retweet count. This is because I don't want to double count a particular retweet for an individual.

MostRetweetedReducer

The reducer takes in a key representing the user and a list of tuples containing tweets and their retweet counts. As the list is iterated through it stores each tweet not already stored, along with its retweet count. If the tweet is already in the list it compares the retweet counts and stored the higher value.

At the end it takes all the retweet counts being stored and sums them, as each count will represent a unique tweet for a particular user this produces an accurate number for how many retweets a particular user gets in the set of data.

Note

After completing the most retweeted user extension, I realised that the `MostRetweetedUserMapper` produces the data needed (the tweet id and retweet count) for the Most retweeted tweet extension and I could minimize code written by simply writing a different reducer for it. However as I had already implemented the most retweeted tweet function, I chose to keep it for simplicity and because it's possible that there would be a slight performance increase since the reducer wouldn't have to receive and remove unneeded data (user id).

English Hashtags

This MapReduce program will replicate the basic requirements but with the added condition that checks if the tweet has the value "en" for its "lang" value.

Implementation

Basic Requirements

TweetCount

I made two public methods: one that specifies the number of threads and mappers to be used for testing, and for general use that uses the default number of mappers and threads.

ScanTweetsMapper

The mapper uses a JSON reader to navigate to the hashtags array in the entities object of a tweet. It then iterates through the array, picking out the text value which would represent what the hashtags actually is.

CountHashtagsReducer

The reducer simply iterated through the list of values associated with a specific key and would output the key alongside the sum of its values.

Extensions

Most Retweeted Tweet

RetweetsMapper

If the tweet provided is a retweet, the mapper gets the id_str of the retweet (as this is unique) and the retweet_count of that tweet and outputs them as the (key,value) pair. If the tweet provided isn't a retweet it checks if it has retweet_count greater than 0 and outputs the id_str and retweet count if it does.

RetweetsReducer

The RetweetsMapper gives lists containing the retweet_count, as this retweet count already sums all the instances that a particular sum has been retweeted, we don't need to sum the list of values associated with the key for this reducer.

Most Retweeted User

MostRetweetedUserMapper

The output key for this class is the user id belonging to the owner of the retweet, and the output value is a Text value in the format <tweetid,retweet_count> This is in order to sum only unique tweets during the reducing step. I chose the comma to delimit the two values for simplicity.

MostRetweetedReducer

First I make a Hashmap with the Key as the tweet id and the Value as the retweet count. I chose to use a Hashmap as it will have constant lookup time increasing the speed of the operation. This efficiency may decrease if the list of values is large enough.

As the list of values is iterated through, I split the value into the tweet Id and the retweet count. Then check if the tweet id has been encountered before, if it hasn't then I put the retweet count into the Hashmap with the tweet id as the key. If the id has been encountered then I compare the counts and store the higher count, as this is more likely to be the higher value.

Testing

Basic Requirements

For the basic requirements, the stacscheck tests were sufficient as they covered a variety of data and for different sizes.

Extensions

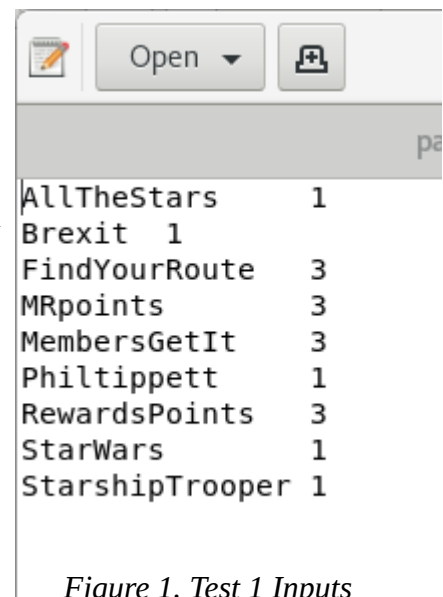
Test 1 – Sorting

Text Brief: given the output of a MapReduce, the sorting MapReduce job should be able to list the most popular hashtags in descending order.

Input: I used the data-very-small set from stacscheck to make a list of hashtags used (*Figure 1.*).

Expected Result: There should be another list which has the number of hashtags as keys and the hashtag as the value.

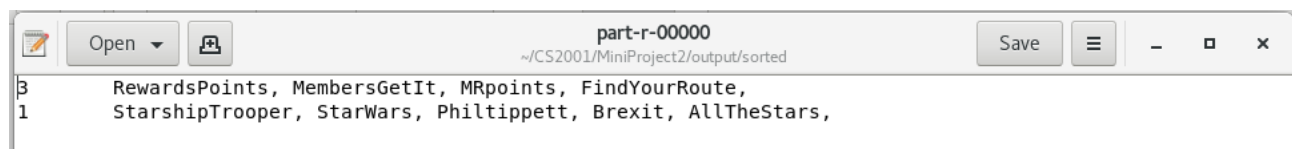
Actual Result: Successful. Figure



```

AllTheStars 1
Brexit 1
FindYourRoute 3
MRpoints 3
MembersGetIt 3
Philtippett 1
RewardsPoints 3
StarWars 1
StarshipTrooper 1
  
```

Figure 1. Test 1 Inputs



```

part-r-00000
~/CS2001/MiniProject2/output/sorted
3 RewardsPoints, MembersGetIt, MRpoints, FindYourRoute,
1 StarshipTrooper, StarWars, Philtippett, Brexit, AllTheStars,
  
```

Figure 2. Test 1 Results.

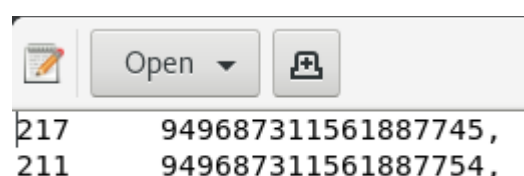
Test 2 – Get Most Retweeted Tweets

Test Brief: Users should be able get a list of the most retweeted tweets. If there are multiple of the same tweet, then the tweet with the higher retweet count should be selected.

Input: Modified data-very-small. I copied the retweet (found in 00.json) so that there were 3 instances. I then edited the 2nd instance to have a higher retweet count at 217, and the 3rd to have a different id_str and a lower count at 211.

Expected Results: 2 tweets. One with 217 retweets and one with 211.

Actual Results: Successful.



```

217 949687311561887745,
211 949687311561887754,
  
```

Figure 3. Test 2 Results.

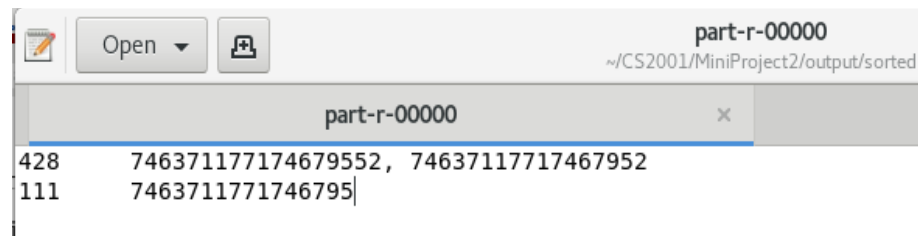
Test 3 – Most Retweeted User

Test Brief: The user should be able to get the most retweeted user's id in descending order.

Input: the modified data set from Test 2. Modified again so that the original tweet is copied so there are 2 more instances. These new instances will have different user id_str for the retweet and have a retweet count of 111, and 428 respectively.

Expected Result: two users with 428 retweets, and one user with 111

Actual Result:
Successful.



part-r-00000	
428	746371177174679552, 74637117717467952
111	7463711771746795

Figure 4. Test 3 Results

Test 4 – English Hashtags

Test Brief: Users should be able to conduct a hashtag search for only English results

Input: 1 hour of data. Jan 10th, 00 hours. This data contains lots of foreign language hashtags.

Expected Output: Only the English Hashtags remain.

Actual Results: Semi-Successful. According to the twitter API the language detection is machine-driven. This probably means that it won't always be accurate to the tweet as can be seen in figure 5. However most of the foreign language tweets have been removed.



part-r-00000	
200userofcomfor	1
2hockey	1
2k	1
2kdraftme	1
2ndNTC	3
2point0	1
30DayPhotographyChallenge	1
32ndGDA	26
32ndGDARedCarpet	18
32ndGoldenDiscAwards	23
32회골든	7
3D	1
3DModeling1	1
3DaysUntilCamila	2
3d	1
44thBirthday	1
4FTL	1
4PipesOnly	1
4YearsOfEpic1NENOKKADINE	2
4YearsOfEpic1NENOKKADINetheater	1
4YearsofYaariyan	1
4YrsOfBlockbusterJILLA	1
4YrsOfPongalWinnerVEERAM	1
6YearsWithSehun	5
6words	1
7News	1
7news	1
90s	1
99c	2
99p	1
9GAGFunOff	2

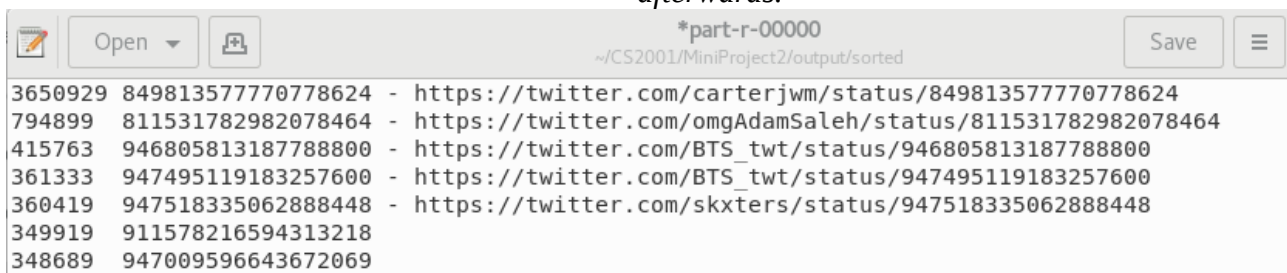
Figure 5. Test 4 Results

Discussion on Twitter data mined

Twitter Data Mined: 2018 January 10th, 01-04 (inclusive).

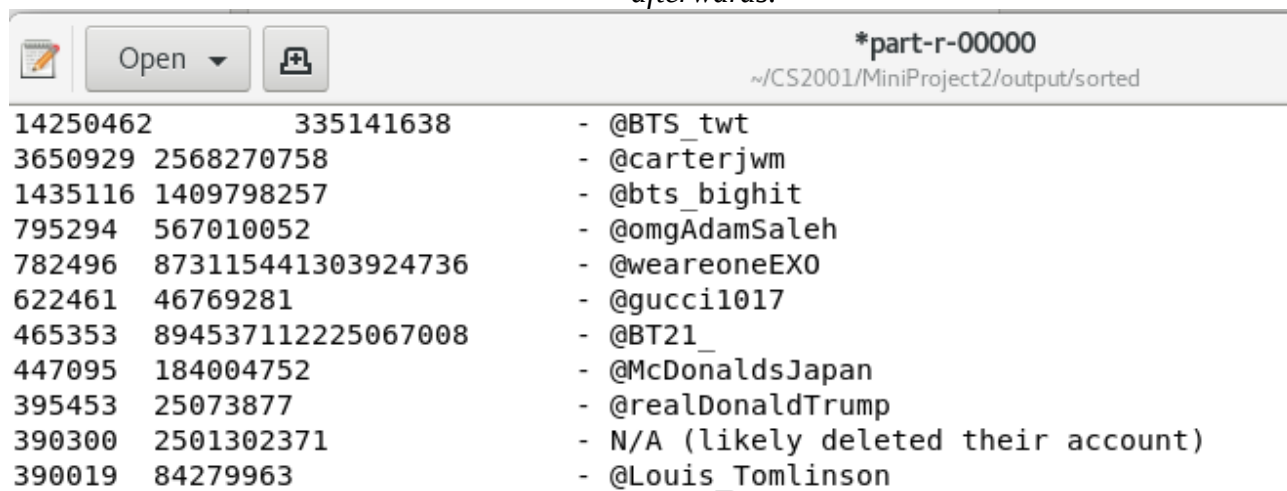
The most popular hashtags at this hour were largely Korean Pop group related. Similarly the most retweeted users were largely singers such as Gucci Mane, Louis Tomlinson and BTS (K-Pop group), with some other famous names such as the President of the USA.

Figure 6. Most retweeted tweets – links added afterwards.



File Name	Content
*part-r-00000	~/CS2001/MiniProject2/output/sorted
3650929 849813577770778624	- https://twitter.com/carterjwm/status/849813577770778624
794899 811531782982078464	- https://twitter.com/omgAdamSaleh/status/811531782982078464
415763 946805813187788800	- https://twitter.com/BTS_twt/status/946805813187788800
361333 947495119183257600	- https://twitter.com/BTS_twt/status/947495119183257600
360419 947518335062888448	- https://twitter.com/skxters/status/947518335062888448
349919 911578216594313218	
348689 947009596643672069	

Figure 7. Most retweeted users – names added afterwards.



File Name	Content
*part-r-00000	~/CS2001/MiniProject2/output/sorted
14250462 335141638	- @BTS_twt
3650929 2568270758	- @carterjwm
1435116 1409798257	- @bts_bighit
795294 567010052	- @omgAdamSaleh
782496 873115441303924736	- @weareoneEX0
622461 46769281	- @guccil017
465353 894537112225067008	- @BT21_
447095 184004752	- @McDonaldsJapan
395453 25073877	- @realDonaldTrump
390300 2501302371	- N/A (likely deleted their account)
390019 84279963	- @Louis_Tomlinson

Analysis of Performance

The results seem to indicate that, at first, increasing the number of mappers has a similar performance improvement to increasing the number of threads. This can be seen from Appendix B where the average job using 1 thread and 2 mappers took a similar amount of time to the job using 2 threads and 1 mapper.

From my data this seems to be when the number of mappers multiplied by the number of threads is equal to 20. This can be seen in Figure 1 where the largest increases in time taken for a job occur between 9 and 10 mappers for the 2 thread data series, 6 and 7 mappers for the 4 thread data series, and so on.

This is likely because at first there is a performance boost as a large task is split up into much smaller task across different nodes and threads. However as the number of nodes x threads increase there is an increase in the amount of time wasted organising the information being handled between the mappers and reducers. This continues until there reaches a point where the time gained by splitting up the job is overtaken by the time lost due to organising the data between the different mappers and reducers.

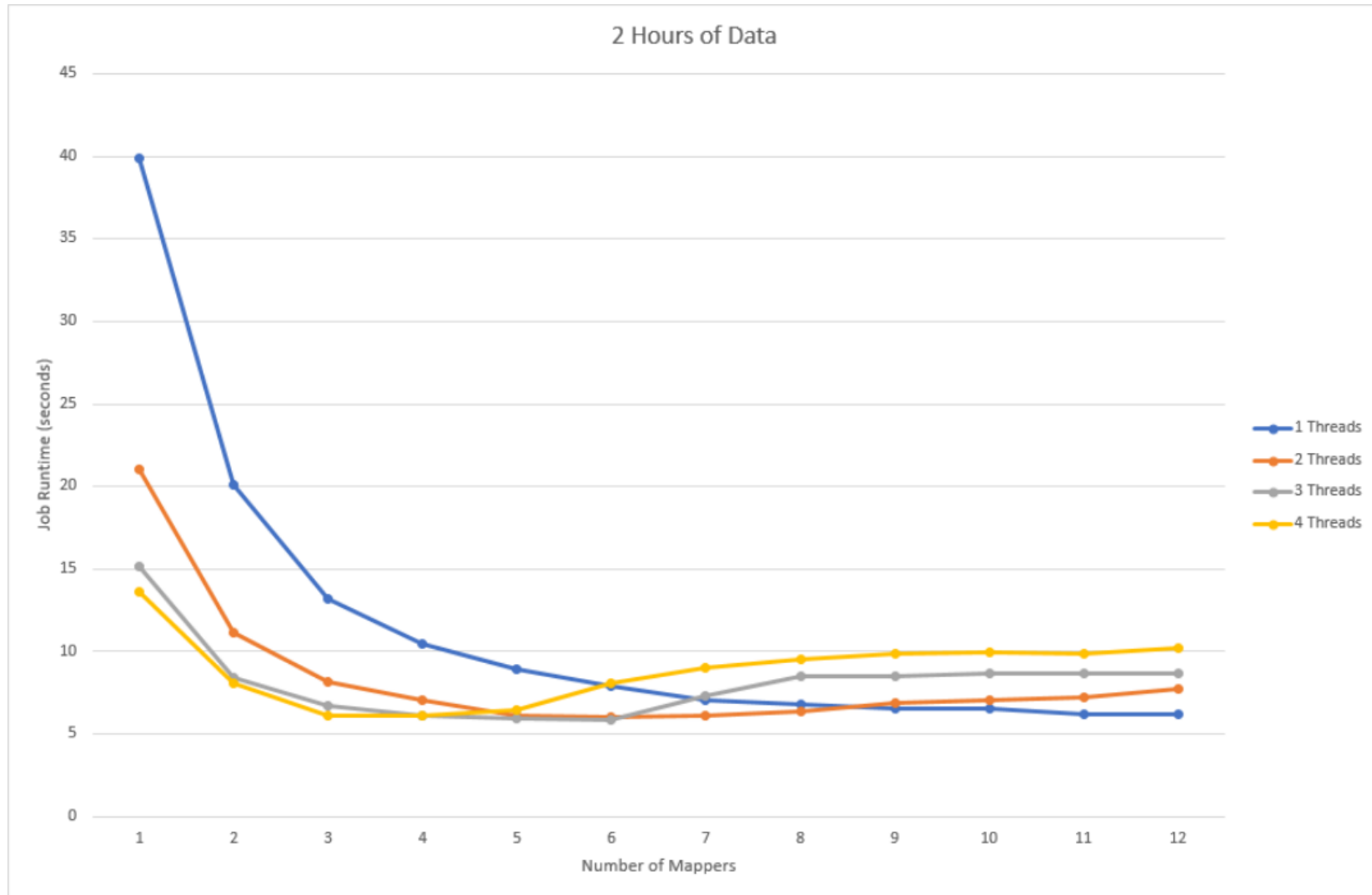
As this occurred at around the point where number of concurrent threads * number of mappers = 20 regardless of whether 2 hours or 1 hour of data is being processed. I suspect that this behaviour will happen independent of the amount of data being processed. Furthermore this is likely to due to the fact that the Palain servers have 24 cores.

8658	iHeartAwards
8407	BestFanArmy
4031	방탄소년단
3833	워너원
3149	EXOL
2690	BTS
2457	BTSARMY
1970	EXO
1595	BTSArmy
1514	JIMIN
1477	지민
1314	WANNAONE
1313	골든디스크
1109	BLACKPINK
1091	강다니엘
1031	GoldenDiscAwards
988	정국
919	GDA
857	뷔
840	웅성우
824	박지훈
755	GDA2018
---	---

Figure 8. Most Used Hashtags

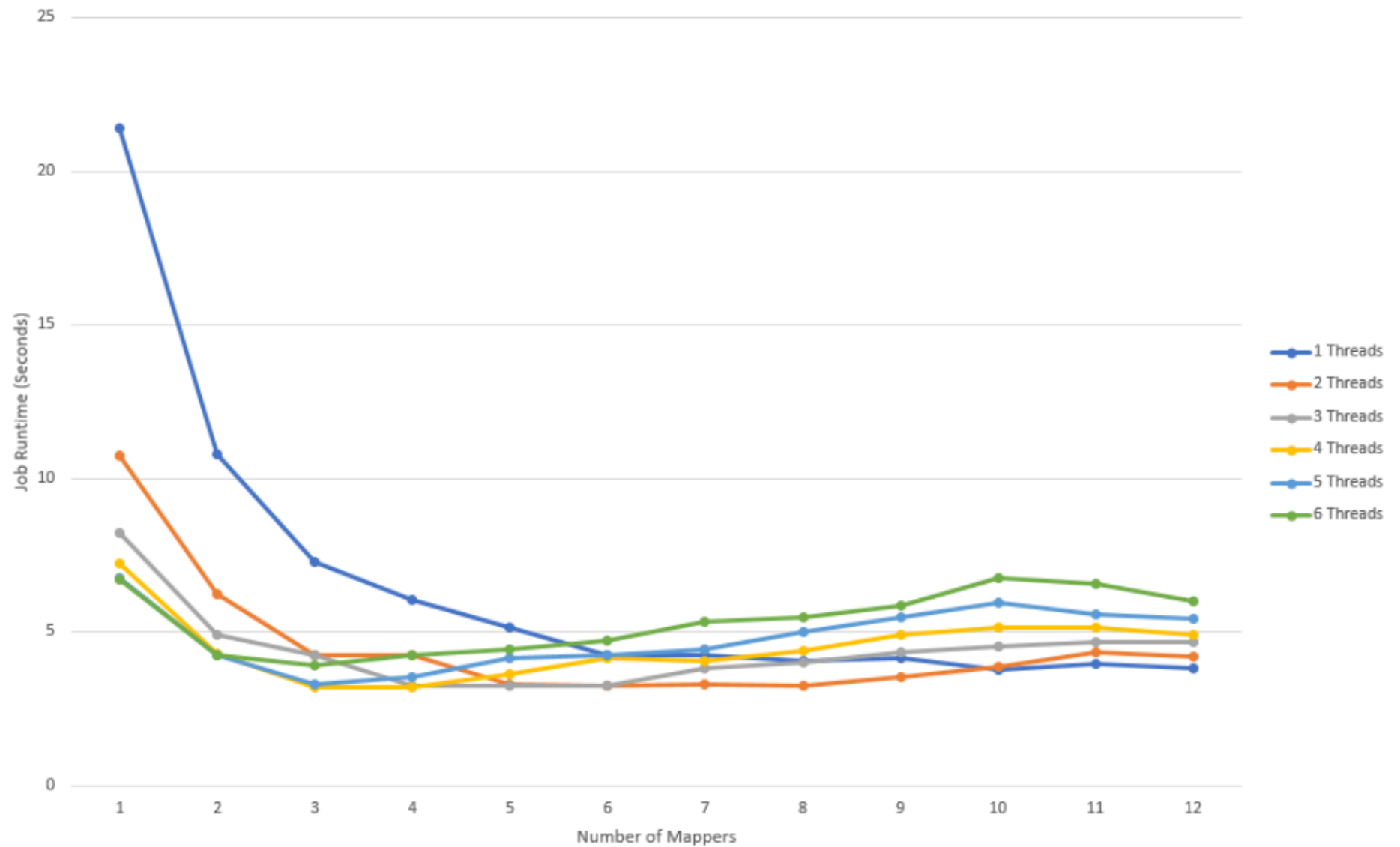
Appendix

Appendix A



Appendix B

1 Hour of Data



Bibliography

<https://commons.apache.org/proper/commons-io/javadocs/api-2.5/org/apache/commons/io/FileUtils.html>

FileUtils was used as a convenient way to delete a directory in program, so the user wouldn't have to delete a directory in file explorer if they wanted to run a MapReduce more than once.

https://studres.cs.st-andrews.ac.uk/CS2101/Examples/CS2101_JsonReaderExample/

https://studres.cs.st-andrews.ac.uk/CS2101/Examples/CS2101_MapReduceExample/

https://studres.cs.st-andrews.ac.uk/CS2101/Examples/CS2101_MapReduce_MultipleRunningMappers_Example/

https://studres.cs.st-andrews.ac.uk/CS2101/Examples/CS2101_MapReduce_MultiThreaded_Example/

The above example code was used to understand how to use the Hadoop framework and read JSON objects in java

<https://tweeterid.com/>

Converted twitter users id to their name using the above website.