

## Mini Project 2

Instructions at the end

### Objective

The objective of this practical was to build a program that parses, collects and processes information from tweets retrieved from twitter using Apache Hadoop to implement a map reduce. We then had to experiment and test to find the optimal number of concurrent mappers, concurrent reducers and the optimal number of threads to process the data in the fastest time possible.

### Introduction







The tweets I used were from July 12<sup>th</sup> between the hours of 9 and 3. Using map reduce I found the most used hashtags, the most retweeted tweets, the most replied to tweets and the most retweeted and replied to users during that time.

I chose the July 12<sup>th</sup> because I knew that was Wimbledon final day, therefore I would expect lots of related tweets. This would possibly help me verify my results were correct

### Map Reduce Results

**The top 10 most retweeted tweets:**

Number of Retweets	Tweet
495	<div><b>Niall Horan</b> ✓ @NiallOfficial</div> <div>Why do I get the feeling this is gona be a long morning ? @Wimbledon</div> <div></div>
463	<div><b>Niall Horan</b> ✓ @NiallOfficial</div> <div>Yesssssssssssss !!!!! Get innnnnnn! Congratulations @DjokerNole ...</div> <div></div>
353	<div><b>5 Seconds of Summer</b> ✓ @5SOS</div> <div>First time I've been to breakfast with Michael... First time he's been to breakfast.</div> <div></div>

346	 <b>Basti Schweinsteiger</b> ✓ @BSchweinsteiger <div>Follow</div> <p>One last dream in my career come true. I am excited about the next chapter in my life with <a href="#">@ManUtd</a>.</p>
330	 <b>Michael Clifford</b> ✓ @Michael5SOS <div>Follow</div> <p>Taking a sleeping tablet to go back to sleep at 8 am....</p>
319	 <b>Niall Horan</b> ✓ @NiallOfficial <div>Follow</div> <p>This Wimbledon is absolutely incredible ! What a performance from both men</p>
309	 <b>Louis Tomlinson</b> ✓ @Louis_Tomlinson <div>Follow</div> <p>🎵 And I owe it all to you 🎵</p>
243	 <b>Calum Hood</b> ✓ @Calum5SOS <div>Follow</div> <p><a href="#">@5SOS</a> whoever tweeted this is a comedic genius. And witty.</p>
201	 <b>Niall Horan</b> ✓ @NiallOfficial <div>Follow</div> <p>Got up super early this morning to watch the tennis and keep up to date with how Westmeath are gettin on <a href="#">#bringonlurch</a></p>
193	<p>I couldn't find the original tweet, it's tweet id was 620331637289713664</p>

The top 10 most reused Hashtags:

Number of uses	Hashtag
3186	VideoVeranoMTV
3129	TeenChoice
1932	ChoiceMusicCollaboration
1865	PrettyGirls
1374	SDCC
1295	Wimbledon
1054	FanArmyFaceOff
988	WimbledonFinal
675	job
640	jobs

The top 10 most replied to tweets:

Number of Replies	Tweet
94	<div>  <div> <b>Calum Hood</b> ✓            @Calum5SOS         </div> <div>Follow</div> </div> <p>@5SOS whoever tweeted this is a comedic genius. And witty.</p> <div> <div>RETWEETS</div> <div>LIKES</div> </div> <div> <div>37,245</div> <div>56,308</div> </div> <div>  </div> <p>12:34 p.m. · 12 Jul 2015</p> <div>     </div>
86	<div>  <div> <b>5 Seconds of Summer</b> ✓            @5SOS         </div> <div>Follow</div> </div> <p>First time I've been to breakfast with Michael... First time he's been to breakfast.</p>
77	<div>  <div> <b>Michael Clifford</b> ✓            @Michael5SOS         </div> <div>Follow</div> </div> <p>Taking a sleeping tablet to go back to sleep at 8 am....</p>

60	<div>  <div> <b>Harry Styles.</b> ✓  @Harry_Styles </div> <div>Follow</div> </div> <p>Thank you everyone who came to the San Diego show last night. Perfect way to start the tour. Thank you so much for having us. All the love</p>
60	<div>  <div> <b>Alfie Deyes</b> ✓  @PointlessBlog </div> <div>Follow</div> </div> <p>It's been a while since I did a follow spree! [PB] in your name for a follow x</p>
51	<div>  <div> <b>Shawn Mendes</b> ✓  @ShawnMendes </div> <div>Follow</div> </div> <p>I love you guys soo much ❤️❤️❤️❤️</p>
46	<div>  <div> <b>Niall Horan</b> ✓  @NiallOfficial </div> <div>Follow</div> </div> <p>Why do I get the feeling this is gona be a long morning ? @Wimbledon</p>
36	<div>  <div> <b>Austin Mahone</b> ✓  @AustinMahone </div> <div>Follow</div> </div> <p>Good morning my beautiful Mahomies ❤️</p>
35	<div>  <div> <b>Niall Horan</b> ✓  @NiallOfficial </div> <div>Follow</div> </div> <p>Yesssssssssssss !!!!! Get innnnnnn!  Congratulations @DjokerNole ...</p>
34	<div>  <div> <b>Shawn Mendes</b> ✓  @ShawnMendes </div> <div>Follow</div> </div> <p>So many bug bites ! They got me while i was on stage! How unfair</p>

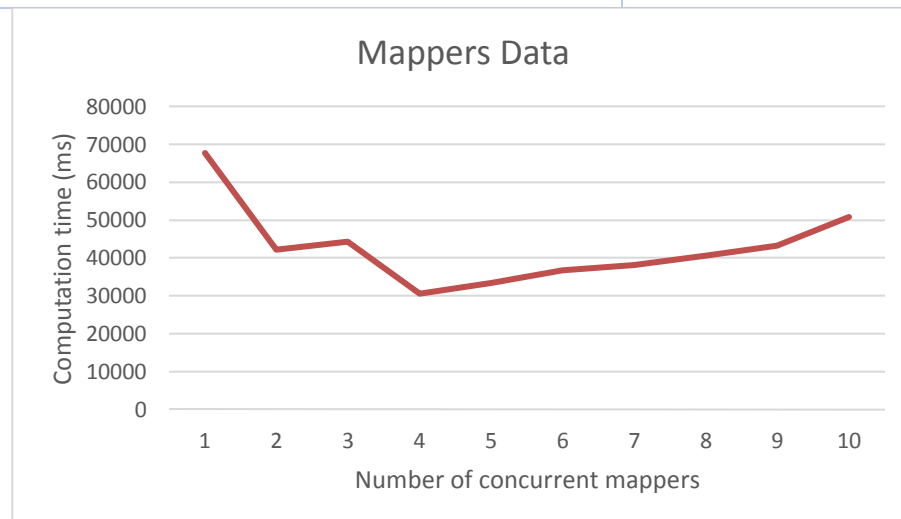
The top to most retweeted and replied to users:

Number of retweets and replies	User
2100	NiallOfficial
1036	Louis_Tomlinson
824	Harry_Styles
679	5SOS
666	all1dcrew
602	Michael5SOS
597	BieberAnnual
580	Calum5SOS
553	girlposts
527	Wimbledon

### Testing Results

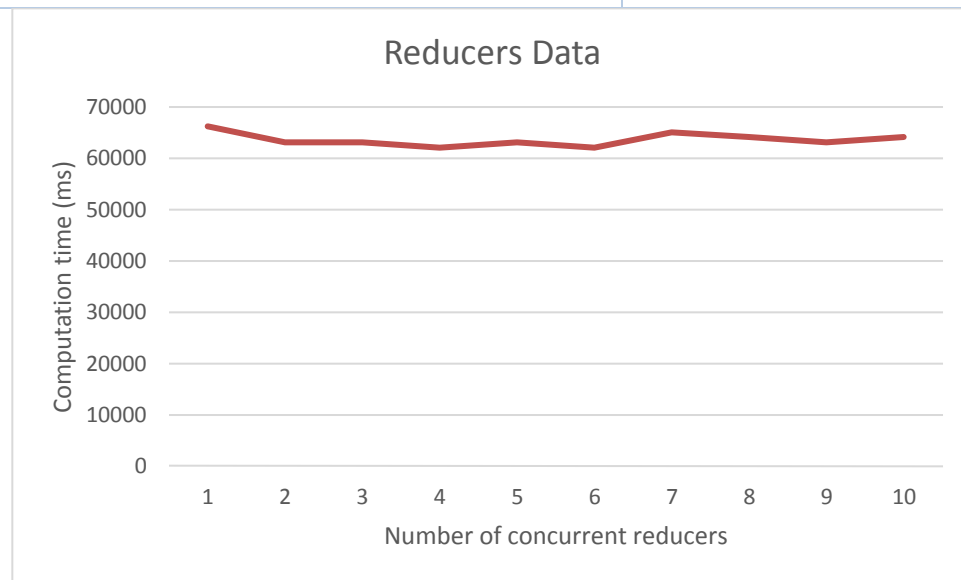
I first tested how the time taken for the program to complete when using a varying number of concurrent mapper tasks. These are the result I got:

Number of concurrent mapper tasks	Time taken for computation (ms)
1	67698
2	42240
3	44392
4	30526
5	33390
6	36708
7	38214
8	40650
9	43270
10	50765



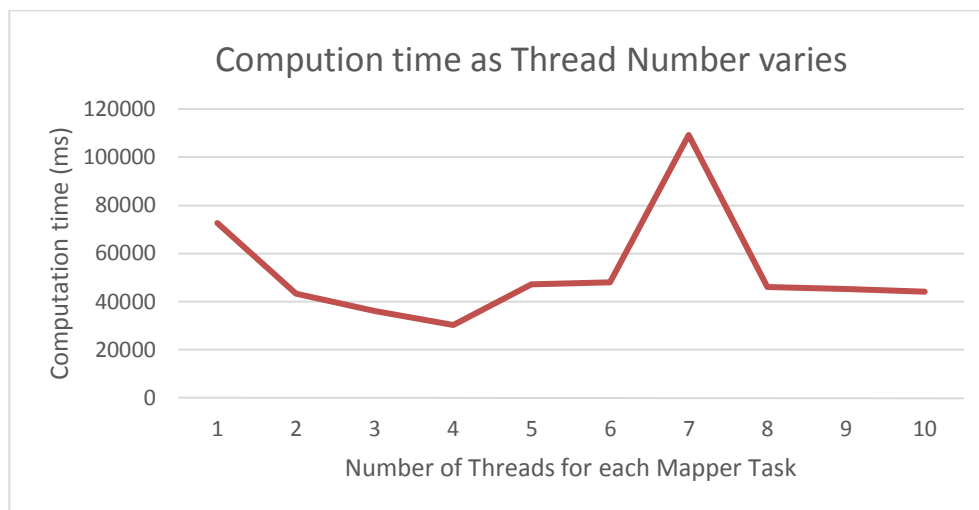
I then tested how much time it took with varying concurrent reduce tasks:

Number of concurrent reducer tasks	Time taken for computation (ms)
1	66222
2	63154
3	63193
4	62185
5	63116
6	62178
7	65168
8	64202
9	63168
10	64141

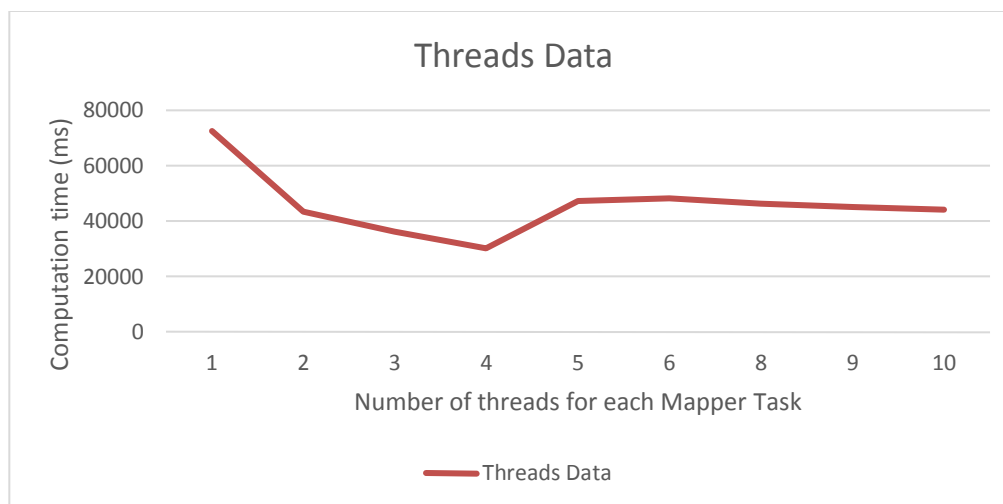


And then I tested with varying number of threads running for each mapping task:

Number of threads per mapper task	Time taken for computation (ms)
1	72540
2	43321
3	36144
4	30181
5	47150
6	48125
7	109140
8	46169
9	45128
10	44089



I identified that when I had 7 threads per map I had an anomaly therefore I re-plotted the graph with it removed.



## Explanation of Results

I found it hard to find information related to finding the optimal number of threads that was accessible to me, but these are possible reasons for the results. In reality it is probably a combination of reasons for each one.

As I increase the number of mappers the time it takes to compute initially drops (I am going to consider 3 threads an anomaly for this explanation, ideally I would have repeated the experiment) before starting to increase again of after 4. I believe this is due to Hadoop putting the new mapper on a new thread on a new core each time I make one (our computers have 4 cores). This increases the how many maps Hadoop can do a second and so the time to compute drops, but after 5 there are no new cores and so the computer starts multithreading which is slightly less efficient and so the performance drops and so does the time to compute.

As I vary the number of reducers the time taken to compute does not seem to vary very much beyond expected deviation. Therefore I believe the system may only utilize one reducer for my program no matter how many there are and so performance doesn't depend on the number of reducers. Ideally I would run the experiment again but changing other variables to see if how the reduces affects time depends on them.

As I vary the number of threads per mapper task, the time taken to compute decreases initially before increasing, the optimal is at 4. One possible reason for this optimal number of threads is that the computers we use are 4 core computers. This means for every thread we spawn below 5 we can use it on another core. Once we spawn 5 threads we have to use multi-threading. As Hadoop on 4 cores will push the cores to close to their limit. As multithreading is slightly less efficient the performance drops and the computing time starts to increase. I only tested this with one multithreaded mapper, if I repeated the practical I would test it with multiple number of mappers.

The reason multithreading is less efficient is because by have more threads than available cores threads have can still only same available CPU resources but there will be an overhead for each new thread with processes like context switching.

A possible reason for the flattening of computation time when varying number of mapper and number of threads per mapper is that the program becomes memory bound at the point of flattening, meaning that the bottleneck is how fast the memory can be written to. By having more threads, the writing cannot become faster and so the threads take in in turn to write to the memory therefore performance cannot improve.

One possible reason I had an anomaly is due to running out of memory as heap size possibly became too full, but as Hadoop writes the console very quickly I didn't catch the error that caused it.

## Code Layout

### Files

#### Mappers



Hashtag Mapper - If the tweet is English, it finds the hashtags used and maps them to 1

Most replied and retweeted user mapper - If the tweet is a retweet, and the original tweet was in English, it finds the original user and maps his name to 1. Else if the tweet is a reply, it finds the user name of the author of the tweet being replied to and maps his name to 1

Most replied tweets mapper - If the tweet is a reply, it finds the user name of the author of the original tweet and maps it to 1

Most retweeted tweet mapper - If the tweet is a retweet and the original tweet is in English, it maps the user name of the author of the original tweet to 1.

Swap mapper - It swaps the Text Key (e.g name of user, hashtag) and the number value associated with it so that the number is the key and the Text is the associated value

## **Reducers**

Count reducer - For each text key, it combines each value associated with the key every time it appears so that the text key appears once and is associated with the number of times it appears

Most popular reducer - For each key and value, it simply rewrites them

## **Tests**

Mappers testing - Creates and prints an array storing the time it took for the program to complete for each number of map tasks between 0 and the number assigned to `maxNumberOfMappers`.

Reducers testing - Creates and prints an array storing the time it took for the program to complete for each number of reduce tasks between 0 and the number assigned to `maxNumberOfReducers`.

Threads testing - Creates and prints an array storing the time it took for the program to complete for each number of threads run by each map task between 0 and the number assigned to `maxNumberOfThreads`.

## **Twitter Analysis**

Twitter Most Popular - In this class I have set up all the jobs that I am going to use to use useful information from the twitter data. I also have a method that extracts the top 40 results from the line I'm using and writes it to a new file.

## **Testing**

To test I used 4 minutes of data from one day, checking I got realistic values before running it with the complete set of data I downloaded

## Design Decisions

Data here could be a hashtag, a name of a user or a tweet id.

In my mappers, for each line of each Json file in the input folder, it converts the whole line into a twitter a Json object, from there I parse that Json object to in order to find the data I'm looking for, and checking the information is from tweets which are relevant for example when choosing the tweets I made sure that all the tweets were English using the 'lang' variable.

Once I find the data I'm looking for I map it to 1 and write it to a file containing the mappings for that data. Each piece of data will be a key and the value associated with it will be 1. Each key may be there multiple times, depending on the number of times the data came up, but each instance of it will be associated with 1. I then call the most popular reducer. For each key it add the value associated with it together, this allows me to get the total number of times data came up, and writes the key once with the value associated with it being the number of times it came up. If I was able to sort the mapping by value I would be able to be finished by setting it to be decending by values but as I'm not I have to call another MapReduce to swap the values.

So I then call the next job, when defining this job I set the sorting of the mapped and reduced files to be in descending order. The mapping task I call is 'swapMapper', this mapper simply swaps the key and the value. The keys are now the number of times a piece of data was called and the associated value is the data itself. This is so that we can sort them in terms of the number of times data was called as opposed to alphabetically as before. As I have set the order to be descending the most called will be at the top. As I already have it in order I call the reducer 'mostPopularReducer' which just rewrites the keys and the values to the file in the same order.

I was able to get the final results, in the correct descending order after the mapping stage in the second map reduce task. Therefore it may seem that my reducer was pointless, and it may have been, I could have just done it with a mapping task, but as we were asked to implement a map-reduce method I felt I should add it in. Also I'm not 100% sure how Hadoop stores the data after mapping, I believe its in 1 file but if it is more It would be necessary for me to reduce so that they are all put in one file

## Evaluation

In regards to the original objective of this project, I believe I completed all basic requirements and additional requirements to a high standard.

If I had more time, I could also have done had more experiments over a larger range. I didn't because it started to trend upwards, performance decreasing but there is no guarantee it would continue that way. I could have varied the number of map tasks when varying the number of threads and see how it affects performance. Also repeating my experiments on the number of mappers, reducers and threads would have made my data more reliable.

## Instructions

To define output in twitter most popular, and the testing files, where ever it says set input path or set output path change to the directory you wish to use

Also in testing change the parameter at the to Max number of ----- to test between 0 and that number for that parameter