Assignment 1

HPC Twitter Data Processing

Farah Zaib Khan

Report analyzing running the same job against 10GB Twitter data using different number of nodes and cores per node.

Student ID: 613572 | Cluster and Cloud Computing

Contents

Introduction:

The objective of the assignment is to develop a simple search engine to parse a 10GB twitter comma separated values file and count following three values:

- Occurrence of a specific term in the data
- Top ten trending topics (with '#' as prefix)
- Top ten most mentioned tweeters (with '@' as prefix)

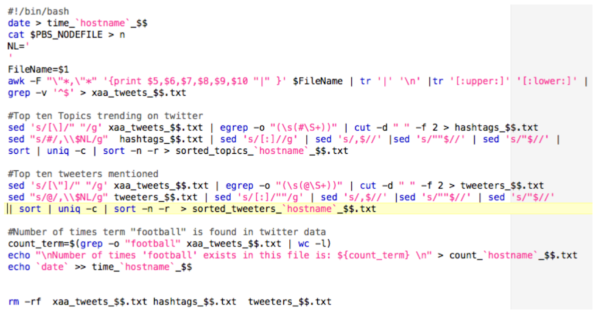
This task is achieved using three different sets of attributes i.e. one node one processor, one node eight processors and two nodes with four processors per node. The target was to analyze the power of parallel computing as compared to serial solution of the problem. For this purpose time elapsed under each attribute setting was noted and comparison is added in the results section.

Scripts:

Following are the scripts written in bash to extract information mentioned in introduction:

1. **Twitter_Parser.sh**

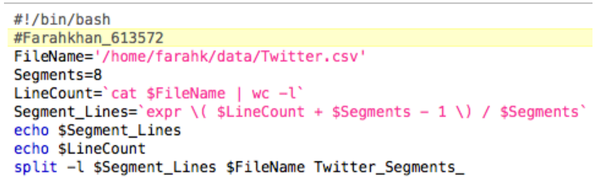
   Twitter_Parser.sh is the search engine to extract occurrence of a specific term, top ten topics and top ten mentioned tweeters. It was written in bash and can deal with any size of file including the 10GB twitter data file. As the only interesting field is the tweet itself therefore it was necessary to get rid of the metadata surrounding the original tweet to extract exact information. Twitter_Parser.sh does that trimming before looking for required information. A collection of operations is performed including *awk, sed, grep, cut, tr, sort* and *uniq* to extract the information. In addition, time is recorded for all the cores if job runs against multiple cores. All the intermediate files are deleted before the script ends in order to get rid of unnecessary files.



```
#!/bin/bash
date > time_`hostname`_$$
cat $PBS_NODEFILE > n
NL='
'
FileName=$1
awk -F "\"*,\"*" '{print $5,$6,$7,$8,$9,$10 "|" }' $FileName | tr '|' '\n' |tr '[:upper:]' '[:lower:]' |
grep -v '^$' > xaa_tweets_$$.txt

#Top ten Topics trending on twitter
sed 's/[\]/" "/g' xaa_tweets_$$.txt | egrep -o "(\s(#\S+))" | cut -d " " -f 2 > hashtags_$$.txt
sed "s/#/,\\$NL/g"  hashtags_$$.txt | sed 's/[:]//g' | sed 's/,$//' |sed 's/""$//' | sed 's/"$//' |
sort | uniq -c | sort -n -r > sorted_topics_`hostname`_$$.txt

#Top ten tweeters mentioned
sed 's/[\"]/" "/g' xaa_tweets_$$.txt | egrep -o "(\s(@\S+))" | cut -d " " -f 2 > tweeters_$$.txt
sed "s/@/,\\$NL/g" tweeters_$$.txt | sed 's/[:]/""/g' | sed 's/,$//' |sed 's/""$//' | sed 's/"$//'
|| sort | uniq -c | sort -n -r  > sorted_tweeters_`hostname`_$$.txt

#Number of times term "football" is found in twitter data
count_term=$(grep -o "football" xaa_tweets_$$.txt | wc -l)
echo "\nNumber of times 'football' exists in this file is: ${count_term} \n" > count_`hostname`_$$.txt
echo `date` >> time_`hostname`_$$


rm -rf  xaa_tweets_$$.txt hashtags_$$.txt  tweeters_$$.txt
```

2. **FileDivider.sh**

   In order to run the script across multiple cores and achieve speedy output, usually there are two options; either we divide the operations into parts and run each part on different core or divide the data into fragments and run the same script for the small fragments on different cores and in the end combine results. In our case there was no way the process could be divided into parts instead dividing data into small fragments and processing on different cores seemed a reasonable approach. This script serves the purpose and divides the Twitter data into 8 equal parts. The files are saved with the same prefix "*Twitter_Segments_*" which proved helpful later. The time elapsed was added to the final time calculations.



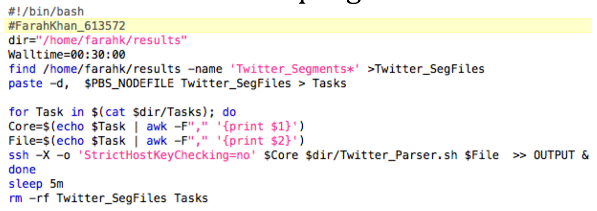
```
#!/bin/bash
#Farahkhan_613572
FileName='/home/farahk/data/Twitter.csv'
Segments=8
LineCount=`cat $FileName | wc -l`
Segment_Lines=`expr \( $LineCount + $Segments - 1 \) / $Segments`
echo $Segment_Lines
echo $LineCount
split -l $Segment_Lines $FileName Twitter_Segments_
```

3. **PBS_script.sh**

   The main PBS script states all the requirements including wall time, number of nodes, cores and error file name. The working directory was set to be the directory from where the job is submitted to queue. In the end the next bash script ""*JobDistribution.sh*" is executed using this file.

4. **JobDistribution.sh**

   This script will fetch the assigned nodes from *$PBS_NODEFILE* and assign each file fragment a processor depending on the availability of resources and run *Twitter_Parser.sh* on the respective core. Sleep time is set to 5 minutes and the script shows the output after the sleep time expires. The sleep time is set so that the process runs in the background and terminates only after the completion of the program and generation of the output. The execution time is measured in *Twitter_Parser.sh*, which does the actual parsing and processing. Therefore despite the 5 minutes wall time, the time returned in a file from every core is the actual execution time of the program.

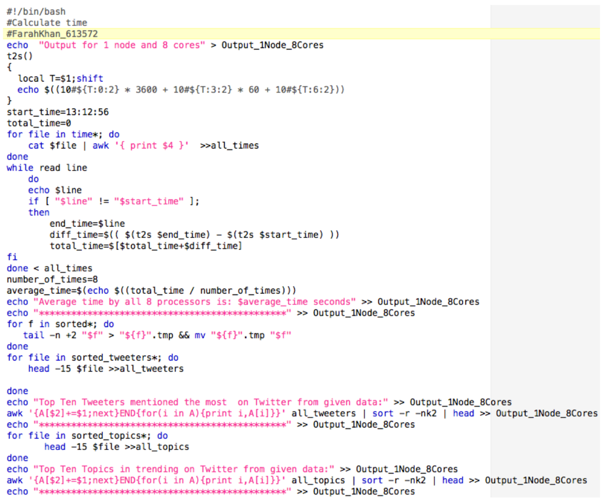


```
#!/bin/bash
#FarahKhan_613572
dir="/home/farahk/results"
Walltime=00:30:00
find /home/farahk/results -name 'Twitter_Segments*' >Twitter_SegFiles
paste -d,  $PBS_NODEFILE Twitter_SegFiles > Tasks

for Task in $(cat $dir/Tasks); do
Core=$(echo $Task | awk -F"," '{print $1}')
File=$(echo $Task | awk -F"," '{print $2}')
ssh -X -o 'StrictHostKeyChecking=no' $Core $dir/Twitter_Parser.sh $File  >> OUTPUT &
done
sleep 5m
rm -rf Twitter_SegFiles Tasks
```

5. **postProcessing.sh:**

Each core returns 4 output files including sorted topics, sorted tweeters, count of the term and time elapsed in execution. This script does not take more than a second and compute the final results out from the four aforementioned files from each core. It takes the files with time as input and for each processor calculates the time difference between the start time (which is same for all cores in one run; hence hard coded) and end time. The average time is computed for 8 cores as the process starts at the same time on all the allocated nodes but there is difference of few seconds in end time. In the similar fashion, the count is accumulated from 8 files and added up to write in the *Output_xNode_xCore(s)* file which contains all the final output.

In addition, in order to avoid loss of any important information *Twitter_Parser.sh* is optimized to return the sorted list of all tweeters and all topics with count in the first column and term in the second column. In *postProcessing.sh*, the head of these files is extracted (15 lines) and concatenated in the same file called *all_tweeters* and *all_topics*. In the end, *awk, sort* and *head* is used to generate the top ten results for tweeters and topics. This script is executed for all three scenarios by changing the start time and file name resulting in generation of three files as *Output_1Node_1Core*, *Output_1Node_8Cores* and *Output_2Node_8Cores* containing time elapsed, count of the term (football), top ten topics and top ten tweeters. The time elapsed was added into the final execution time.



```
#!/bin/bash
#Calculate time
#FarahKhan_613572
echo  "Output for 1 node and 8 cores" > Output_1Node_8Cores
t2s()
{
  local T=$1;shift
  echo $((10#${T:0:2} * 3600 + 10#${T:3:2} * 60 + 10#${T:6:2}))
}
start_time=13:12:56
total_time=0
for file in time*; do
    cat $file | awk '{ print $4 }'  >>all_times
done
while read line
    do
    echo $line
    if [ "$line" != "$start_time" ];
    then
        end_time=$line
        diff_time=$(( $(t2s $end_time) - $(t2s $start_time) ))
        total_time=$[$total_time+$diff_time]
fi
done < all_times
number_of_times=8
average_time=$(echo $((total_time / number_of_times)))
echo "Average time by all 8 processors is: $average_time seconds" >> Output_1Node_8Cores
echo "**********************************************" >> Output_1Node_8Cores
for f in sorted*; do
   tail -n +2 "$f" > "${f}".tmp && mv "${f}".tmp "$f"
done
for file in sorted_tweeters*; do
    head -15 $file >>all_tweeters

done
echo "Top Ten Tweeters mentioned the most  on Twitter from given data:" >> Output_1Node_8Cores
awk '{A[$2]+=$1;next}END{for(i in A){print i,A[i]}}' all_tweeters | sort -r -nk2 | head >> Output_1Node_8Cores
echo "**********************************************" >> Output_1Node_8Cores
for file in sorted_topics*; do
       head -15 $file >>all_topics
done
echo "Top Ten Topics in trending on Twitter from given data:" >> Output_1Node_8Cores
awk '{A[$2]+=$1;next}END{for(i in A){print i,A[i]}}' all_topics | sort -r -nk2 | head >> Output_1Node_8Cores
echo "**********************************************" >> Output_1Node_8Cores
```

Analysis:

The results were of course similar in terms of output but the time to run the job differs in three cases. As mentioned in the assignment due to large number of students in class and queuing lags, it was not appropriate to run qsub for the same job multiple times to benchmark the time calculations therefore execution time will be correct with offset of few seconds if the same code is executed again later.

Results:

The results for three sets of attributes with respect to nodes and cores are added to the appendix of the report.

As shown in the screen shots of the output, the top ten tweeters, top ten topics and the number of times term “football” exists in the file are noted along with the total time consumed by the *Twitter_parser.sh*.

Table 1 shows the comparison between the three given scenarios showing the time in seconds. For 1 node and 1 core the file does not need any preprocessing in terms of dividing the big file in fragments hence no time for preprocessing is included. In total the *Twitter_parser.sh* took 220 seconds and the results were processed a bit after the *Twitter_parser.sh*, which took 0.037 seconds. In both other cases, we need to divide the file into 8 fragments as the maximum number of cores was 8. Instead of adding that part in the main execution, *FileDivider.sh* was used separately as explained before in order to use fragments from one scenario (1N8C) in the other (2N8C) as well.

As shown in table, 1 node with 8 cores took in total 152.153 seconds and on the other hand 2 nodes with 8 cores took at most 120.3 seconds.

The reason for the difference between 1 node 8 cores and 2 nodes 8 cores is not evident as Edward assigned all cores on the same node in case of 2 nodes scenario too. After reading LMS discussion it became clear that Edward’s priority is to assigns all cores on the same node if available otherwise switch to the different node for remaining cores. That being noticed, this slight difference of time is not clear from these results. One assumption is that this difference is because of absence of benchmarking and may be running the code for 1 node 8 cores again we would get time with offset of few seconds and that might be the reason.

Scenario	PreProcess	Execution	PostProcess	Total
1N1C	0 sec	220 sec	0.037 sec	220.037 sec
1N8C	71 sec	81 sec	0.153 sec	152.153 sec
2N8C	71 sec	49 sec	0.3 sec	120.300 sec

Table 1 Time difference between 1N1C, 1N8C and 2N8C

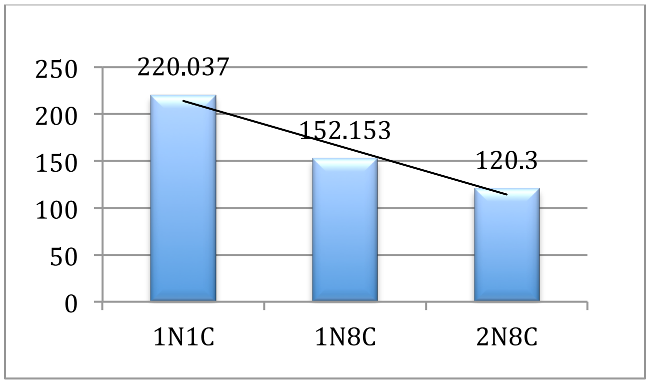


Figure 1 Graph showing the time difference with scenarios on x-axis and time in seconds on y-axis

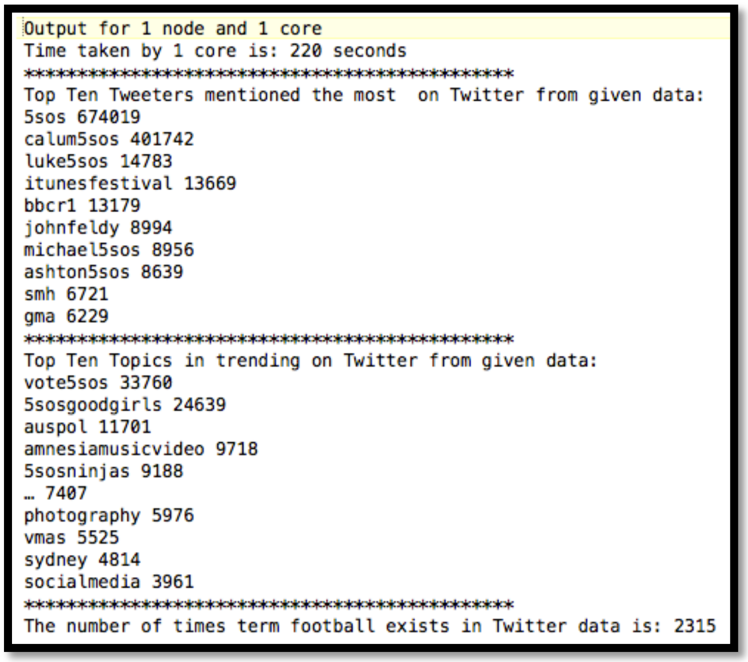
Figure 1 shows time difference stated in the table as it can be seen clearly that time decreases as we increase the number of cores.

Conclusion:

After carefully analyzing the three scenarios it can be concluded that HPC utilizes all the available resources and get the results quicker as compared to using only one core. This task was simple yet sophisticated as in the end the goal has been achieved with focus on HPC instead of search engine itself. The use of qsub no doubt required patience especially in the end because of queue time but the time of execution of the task was cut down remarkably.

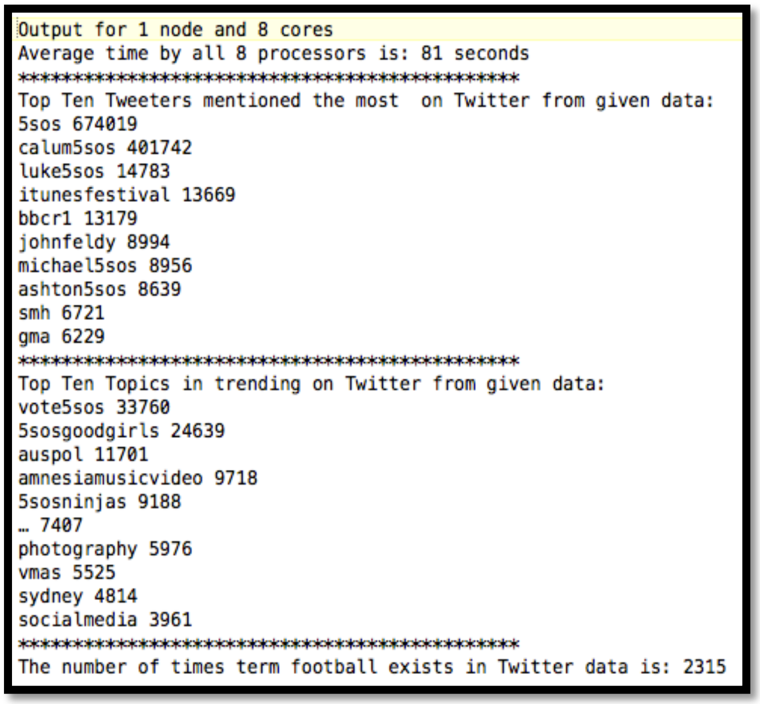
Appendix:

1. **One node One Core:**



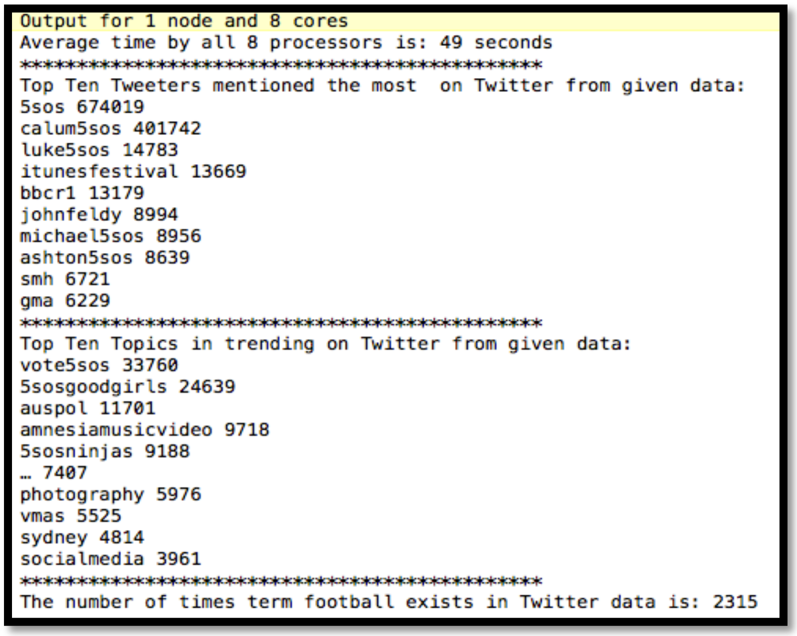
```
Output for 1 node and 1 core
Time taken by 1 core is: 220 seconds
**********************************************
Top Ten Tweeters mentioned the most  on Twitter from given data:
5sos 674019
calum5sos 401742
luke5sos 14783
itunesfestival 13669
bbcr1 13179
johnfeldy 8994
michael5sos 8956
ashton5sos 8639
smh 6721
gma 6229
**********************************************
Top Ten Topics in trending on Twitter from given data:
vote5sos 33760
5sosgoodgirls 24639
auspol 11701
amnesiamusicvideo 9718
5sosninjas 9188
… 7407
photography 5976
vmas 5525
sydney 4814
socialmedia 3961
**********************************************
The number of times term football exists in Twitter data is: 2315
```

2. **One node Eight Cores:**



```
Output for 1 node and 8 cores
Average time by all 8 processors is: 81 seconds
**********************************************
Top Ten Tweeters mentioned the most  on Twitter from given data:
5sos 674019
calum5sos 401742
luke5sos 14783
itunesfestival 13669
bbcr1 13179
johnfeldy 8994
michael5sos 8956
ashton5sos 8639
smh 6721
gma 6229
**********************************************
Top Ten Topics in trending on Twitter from given data:
vote5sos 33760
5sosgoodgirls 24639
auspol 11701
amnesiamusicvideo 9718
5sosninjas 9188
… 7407
photography 5976
vmas 5525
sydney 4814
socialmedia 3961
**********************************************
The number of times term football exists in Twitter data is: 2315
```

3. **Two nodes Eight Cores:**



```
Output for 1 node and 8 cores
Average time by all 8 processors is: 49 seconds
**********************************************
Top Ten Tweeters mentioned the most  on Twitter from given data:
5sos 674019
calum5sos 401742
luke5sos 14783
itunesfestival 13669
bbcr1 13179
johnfeldy 8994
michael5sos 8956
ashton5sos 8639
smh 6721
gma 6229
**********************************************
Top Ten Topics in trending on Twitter from given data:
vote5sos 33760
5sosgoodgirls 24639
auspol 11701
amnesiamusicvideo 9718
5sosninjas 9188
… 7407
photography 5976
vmas 5525
sydney 4814
socialmedia 3961
**********************************************
The number of times term football exists in Twitter data is: 2315
```

Note: In top ten topics there was a topic written in Chinese which was not detected and the Chinese alphabets were replaced by three dots as "…".