Cluster and Cloud Computing Assignment 1 – Social Media Analytics

Problem Description

Your task in this programming assignment is to implement a parallelized application leveraging the University of Melbourne HPC facility SPARTAN. Your application will use a large Twitter dataset and a file containing the suburbs, locations and Greater Capital cities of Australia. Your objective is to:

- count the number of different tweets made in the Greater Capital cities of Australia,
- identify the Twitter accounts (users) that have made the most tweets, and
- identify the users that have tweeted from the most different Greater Capital cities.

You should be able to log in to SPARTAN through running the following command:

ssh your-unimelb-username@spartan.hpc.unimelb.edu.au

with the password you set for yourself on *karaage* (https://dashboard.hpc.unimelb.edu.au/karaage). Thus, I would log in as:

ssh rsinnott@spartan.hpc.unimelb.edu.au
password = my karaage password (not my UniMelb password)

If you are a Windows user then you may need to install an application like Putty.exe to run *ssh*. (If you are coming from elsewhere with different firewall rules, then you may need to use a VPN).

The files to be used in this assignment are accessible at:

- /data/projects/COMP90024/bigTwitter.json
  - this is the main 18.74Gb+ JSON file to use for your final analysis and report write up, i.e., **do not use the bigTwitter.json file for software development and testing**.
- /data/projects/COMP90024/smallTwitter.json
  - smallTwitter.json this a small (242Mb) JSON file that should be used for testing
- /data/projects/COMP90024/tinyTwitter.json
  - tinyTwitter.json this a very small (1.5Mb) JSON file that should be used for initial testing
- /data/projects/COMP90024/sal.json
  - This is the file containing the suburbs, locations and Greater Capital cities of Australia.

You may decide to use the tiny/small JSON files on your own PC/laptop to start with for development and testing.

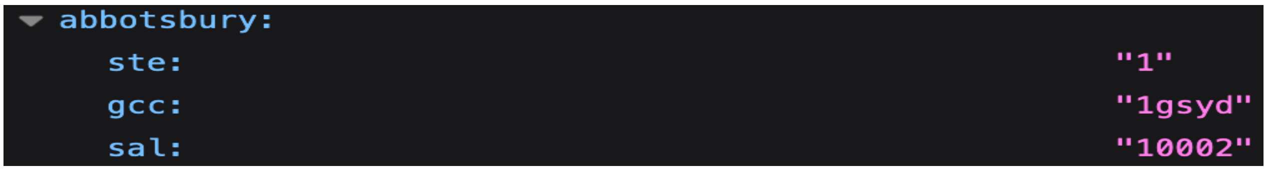
You should make a symbolic link to these files on SPARTAN, i.e., you should run the following commands at the Unix prompt **from your own user directory on SPARTAN**:

```
ln –s /data/projects/COMP90024/bigTwitter.json
ln –s /data/projects/COMP90024/smallTwitter.json
ln –s /data/projects/COMP90024/tinyTwitter.json
ln –s /data/projects/COMP90024/sal.json
```

Once done you should see something like the following **in your home directory**:

```
lrwxrwxrwx 1 rsinnott unimelb   40 Mar 22 15:06 bigTwitter.json -> /data/projects/COMP90024/bigTwitter.json
lrwxrwxrwx 1 rsinnott unimelb   39 Mar 22 15:06 smallTwitter.json -> /data/projects/COMP90024/smallTwitter.json
lrwxrwxrwx 1 rsinnott unimelb   38 Mar 22 15:06 tinyTwitter.json -> /data/projects/COMP90024/tinyTwitter.json
lrwxrwxrwx 1 rsinnott unimelb   41 Mar 22 15:06 sydGrid.json -> /data/projects/COMP90024/sal.json
```

The *sal.json* file has the structure as shown in the figure below for a given suburb.



```
▾ abbotsbury:
    ste:                                                    "1"
    gcc:                                                    "1gsyd"
    sal:                                                    "10002"
```

The suburb here is Abbotsbury and this is part of Greater Sydney (indicated by **1gsyd** above). You will find that some suburb names (strings) are repeated many times, e.g., the suburb name Richmond occurs several times in New South Wales, Victoria, South Australia, Tasmania etc and also appears as substrings of suburb names, e.g., North Richmond, Richmond Lowlands, Broad Water (Richmond Valley – NSW) and some of these are in Greater Capital cities and rural locations. Your solution should aim to tackle such issues and dealing with potential ambiguities and/or flag cases where tweets cannot be resolved to a single unique location.

The tweets themselves can include the precise location, e.g., *Kempsey, New South Wales*, or the approximate location of where the tweet was made, e.g., *Australia* or *New South Wales* or indeed potentially non-Australian locations. Such approximate and non-Australian locations can be ignored from the analysis, i.e., unless the specific string is in the *sal.json* file it can be ignored. You can assume that city names given in tweets such as Sydney, New South Wales corresponds to Sydney (13730), Melbourne, Victoria corresponds to Melbourne (21640), Brisbane, Queensland corresponds to Brisbane (30379) etc.

Each tweet also contains a unique author id for the tweeter, e.g., *836119507173154816* for the first tweet in the *tinyTwitter.json* file.

Task 1: Your solution should count the number of tweets made by the same individual based on the *bigTwitter.json* file and return the top 10 tweeters in terms of the number of tweets made irrespective of where they tweeted. The result will be of the form (where the author Ids and tweet numbers are representative).

Rank	**Author Id**	**Number of Tweets Made**
#1	*123456789101112131*	*999*
#2	*234567891011121415*	*888*
#3	*345678910111214161*	*777*
#4	*456789101112141617*	*666*
…		
#10	*etc*	

Task 2: Using the *bigTwitter.json* and the *sal.json* file you will then count the number of tweets made in the various capital cities by all users. The result will be a table of the form (where the numbers are representative).

Greater Capital City	**Number of Tweets Made**
1gsyd (Greater Sydney)	99,999
2gmel (Greater Melbourne)	88,888
3gbri (Greater Brisbane)	77,777
4gade (Greater Adelaide)	66,666
5gper (Greater Perth)	55,555
6ghob (Greater Hobart)	44,444
…other capital cities	

For this task, you may ignore tweets made by users in rural locations, e.g., *1rnsw* (Rural New South Wales), *1rvic* (Rural Victoria) etc.

Task 3: Finally, your solution should identify those tweeters that have tweeted in the most Greater Capital cities and the number of times they have tweeted from those locations. The top 10 tweeters making tweets from the most different locations should be returned and if there are equal number of locations, then these should be ranked by the number of tweets made as shown in the table below – again with representative data.

Rank	**Author Id**	**Number of Unique City Locations and #Tweets**
#1	*5678910111213141516*	*6 (#54 tweets - #9gsyd, #9gmel, #9gbri, #9gade, #9gper, #9ghob)*
#2	*6789101112141516171*	*6 (#48 tweets - #8gsyd, #8gmel, #8gbri, #8gade, #8gper, #8ghob)*
#3	*7891011121416171819*	*5 (#200 tweets - #40gsyd, #40gmel, #40gbri, #40gade, #40gper)*
#4	*8910111214161718192*	*5 (#100 tweets -#90gsyd, #5gmel, #3gbri, #1gade, #1gper)*
#5	*9101112141617181920*	*4 (#500 tweets - #10gsyd, #480gmel, #9gbri, #1gade)*
…		
#10	*etc*	

Note that for this task, only those tweets made in Greater Capital cities should be counted, e.g., if author Id = *5678910111213141516* tweets 1000 times from rural New South Wales then these can be ignored.

Your application should allow a given number of nodes and cores to be utilized. Specifically, **your application should be run once** to search the *bigTwitter.json* file on each of the following resources:

- 1 node and 1 core;
- 1 node and 8 cores;
- 2 nodes and 8 cores (with 4 cores per node).

The resources should be set when submitting the search application with the appropriate *SLURM* options. Note that you should run a single *SLURM* job three separate times on each of the resources given here, i.e. you should not need to run the same job 3 times on 1 node 1 core for example to benchmark the application. (This is a shared facility and this many COMP90024 students will consume a lot of resources!).

You can implement your solution using any routines and libraries you wish however it is strongly recommended that you follow the guidelines provided on access and use of the SPARTAN cluster. Do not for example think that the job scheduler/SPARTAN automatically parallelizes your code – it doesn't! You may wish to use the pre-existing MPI libraries that have been installed for C, C++ or Python, e.g., mpi4py. You should feel free to make use of the Internet to identify which JSON processing libraries you might use. You may also use any regular expression libraries that you might need for string comparison.

Your application should return the final results and the time to run the job itself, i.e. the time for the first job starting on a given SPARTAN node to the time the last job completes. You may ignore the queuing time. The focus of this assignment is not to optimize the application to run faster, but to learn about HPC and how basic benchmarking of applications on a HPC facility can be achieved and the lessons learned in doing this on a shared resource.

Final packaging and delivery

You should write a brief report on the application – **no more than 4 pages**!, outlining how it can be invoked, i.e. it should include the scripts used for submitting the job to SPARTAN, the approach you took to parallelize your code, and describe variations in its performance on different numbers of nodes and cores. Your report should include the actual results tables as outlined above and a single graph (e.g., a bar chart) showing the time for execution of your solution on 1 node with 1 core, on 1 node with 8 cores and on 2 nodes with 8 cores.

Deadline

The assignment should be submitted to Canvas as a zip file. The zip file must be named with the students named in each team and their student Ids. That is, *ForenameSurname-StudentId:ForenameSurname-StudentId* might be *<SteveJobs-12345:BillGates-23456>.zip*. Only one report is required per student pair and only one student needs to upload this report. The deadline for submitting the assignment is: **Wednesday 12th April (by 12 noon!)**.

It is strongly recommended that you do not do this assignment at the last minute, as it may be the case that the Spartan HPC facility is under heavy load when you need it and hence it may not be available! You have been warned…!!!!

Marking

The marking process will be structured by evaluating whether the assignment (application + report) is compliant with the specification given. This implies the following:

- A working demonstration – **60% marks**
- Report and write up discussion – **40% marks**

Timeliness in submitting the assignment in the proper format is important. **A 10% deduction per day will be made for late submissions.**

You are free to develop your system where you are more comfortable with (at home, on your PC/laptop, in the labs, on SPARTAN itself - but not on the *bigTwitter.json* file until you are ready!). Your code should of course work on SPARTAN.