Cluster and Cloud Computing Assignment 1 – The Happiest City

Problem Description

Your task in this programming assignment is to implement a simple, parallelized application leveraging the University of Melbourne HPC facility SPARTAN. Your application will use a large Twitter dataset, a grid/mesh for Melbourne and a simple dictionary of terms related to sentiment scores. Your objective is to calculate the sentiment score for the given cells and hence to calculate the area of Melbourne that has the happiest/most miserable people!

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 14Gb+ 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.** Note that this data covers several cities in Australia (not just Melbourne).
- /data/projects/COMP90024/smallTwitter.json
  - smallTwitter.json this a 35Mb+ JSON file that can be used for testing;
- /data/projects/COMP90024/tinyTwitter.json
  - tinyTwitter.json this a small JSON file that should be used for initial testing
  - You may also decide to use the smaller JSON files on your own PC/laptop to start with.

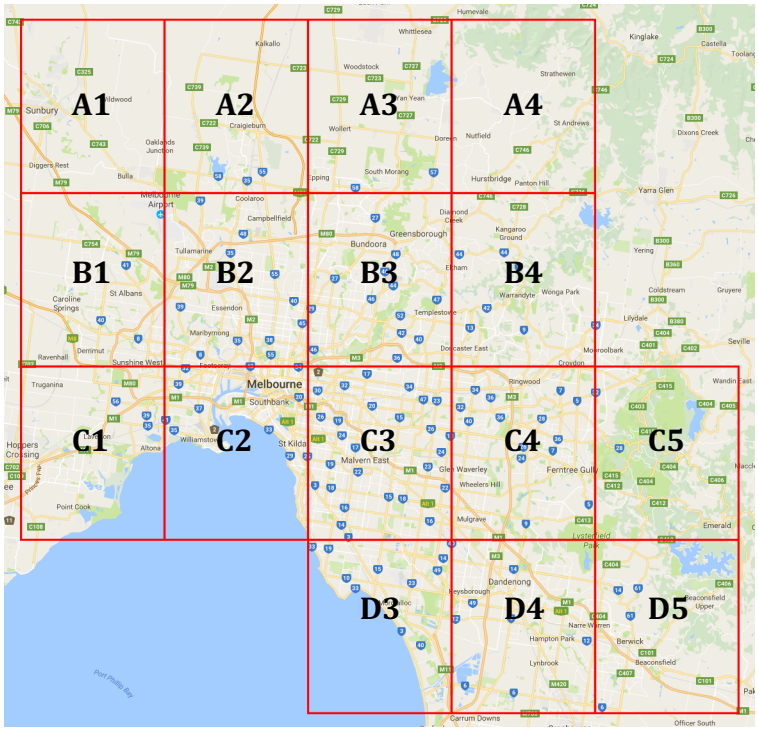
You should make a symbolic link to these files, i.e. you should run the following commands at the Unix prompt **from your own user directory on SPARTAN**:

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

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 melbGrid.json -> /data/projects/COMP90024/melbGrid.json
lrwxrwxrwx 1 rsinnott unimelb  42 Mar 22 15:06 AFINN.txt -> /data/projects/COMP90024/AFINN.txt
```

The *melbGrid.json* file includes the latitudes and longitudes of a range of gridded boxes as illustrated in the figure below, i.e., the latitude and longitude of each of the corners of the boxes is given in the file.



lexicon

The AFINN.txt file contains a list of words with a score related to the sentiment of the word, i.e., the extent that the words are happy or sad. For example:

abandon -2
abandoned -2
abandons -2
…
happy +3
…
sad -2

counter {key: locations, scores for exact matches}

Your assignment is to (*eventually*!) search the large Twitter data set (*bigTwitter.json*) and **using just the tweet text** and the **tweet location** (lat/long) that contain **exact matches** of the terms in the AFINN.txt file, count the total number of tweets in a given cell and aggregate the sentiment score for each grid cell for all of the data. The final result will be a score for each cell with the following format, where the numbers are obviously representative.

Cell	#Total Tweets	#Overal Sentiment Score
A1	11,111	+123
A2	22,222	-234
A3	33,333	+345
A4	44,444	-456
…		
D3	55,555	+678
D4	66,666	-789
D5	77,777	+890

Only exact matches are required for the tweet text. Thus *"#abandon*" or *"@abandon*" or "*abandoning*" or "*abandon23*" or "*abandon-COMP90024*" etc are not an exact match and can be ignored. If a word ends in one of the following forms of punctuation: **! , ? . ' "** then it can be regarded as an exact match, e.g. "*COMP90024 is a course you should not abandon!*" would match on "*abandon*". A tweet may have multiple matches, e.g., "*Sad to abandon COMP90024*" would score -4 (-2 abandon, -2 sad). The words should be treated as case insensitive, e.g., "*Abandon*" and "*abandon*" and "*AbAnDoN*" can be considered as the same word and hence a match.

boundary cases

If a tweet occurs right on the border of two cells, e.g., exactly between the B1/B2 cell border then assume the tweet occurs in B1 (i.e., to the cell on the left). If a tweet occurs exactly on the border between B2/C2 then assume the tweet occurs in C2 (i.e., to the cell below).

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 search using any routines that you wish from existing libraries 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. You should feel free to make use of the Internet to identify which JSON processing libraries you might use.

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 also include 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.

The deadline for submitting the assignment is: **Monday 14th 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.