Cluster and Cloud Computing Assignment 1 – HPC Twitter GeoProcessing

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 search a large geocoded Twitter dataset to identify tweet hotspots around Melbourne.

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

ssh your-unimelb-username@spartan2.hpc.unimelb.edu.au

with your University password. Thus I would log in as:

ssh rsinnott@spartan2.hpc.unimelb.edu.au

If you are a Windows user then you may need to install an application like Putty.exe to run *ssh*.

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

- /data/projects/COMP90024/bigTwitter.json
  - this is the main 10Gb JSON file to use for your final analysis and report write up;
- /data/projects/COMP90024/smallTwitter.json
  - tinyTwitter.json this a 32Mb JSON file should be used for final testing, i.e. do not use the bigTwitter.json file for software development and testing.
- /data/projects/COMP90024/tinyTwitter.json
  - tinyTwitter.json this a 3.2Mb JSON file should be used for initial development and testing, i.e. at the risk of being repetitive, do not use the bigTwitter.json file for software development and testing.
  - You may also decide to use the JSON files on your own PC/laptop to start with.
- /data/projects/COMP90024/melbGrid
  - this is a small JSON-based Grid file for Melbourne.

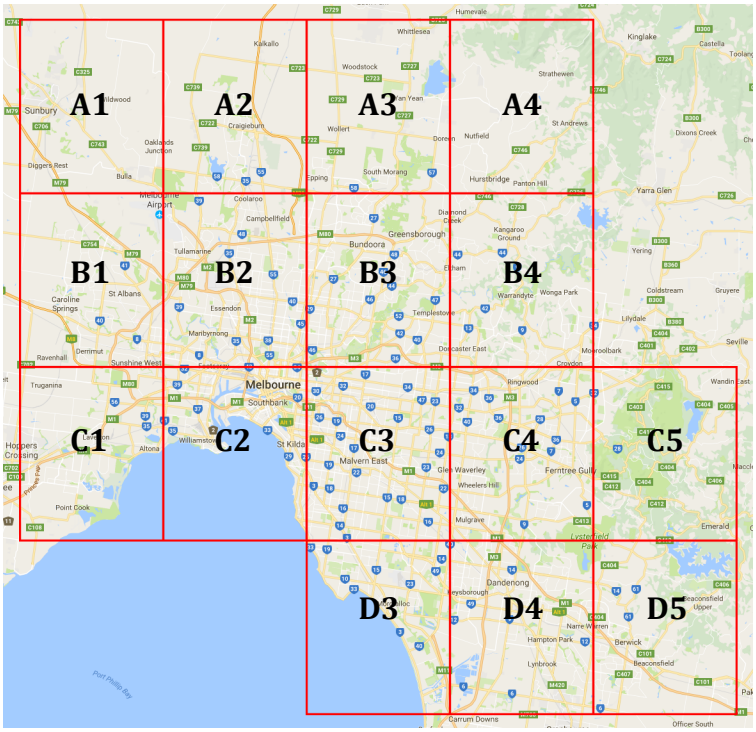
You should make a symbolic link to these files, i.e. you should run the following commands at the Unix prompt **from your own user directory on SPARTAN**:

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

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.



Your assignment is to (eventually!) search the large Twitter data set (*bigTwitter.json*) to identify Twitter activity around Melbourne. Specifically you should:

- Order (rank) the Grid boxes based on the total number of tweets made in each box and return the total count of tweets in each box, e.g.
  - C3: 123,456 tweets,
  - B2: 122,345 tweets,
  - D1: 121,234 tweets,
  - …
  - Down to the square with the least number of tweets;
- Order (rank) the rows based on the total number of tweets in each row, e.g.
  - B-Row: 567,890 tweets,
  - A-Row: 456,789 tweets,
  - C-Row: 234,567 tweets,
  - D-Row: 123,456 tweets.
- Order (rank) the columns based on the total number of tweets in each column, e.g.
  - Column 3: 789,012 tweets,
  - Column 2: 678,901 tweets,
  - Column 4: 567,890 tweets,
  - Column 1: 456,789 tweets,
  - Column 5: 123,456 tweets

(Obviously these numbers are representative of how many tweets each box contains!)

An individual tweet can be considered to occur in the box if its geo-location information (the tweet latitude and longitude given by the tweet coordinates) is within the box identified by the set of coordinates in *melbGrid.json*. It should be noted that the file *bigTwitter.json* includes many tweets that are not in this grid, e.g. they are from other Australian cities or from other parts of Victoria. You should filter/remove these tweets since only the tweets in the grid boxes identified here are of interest.

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 200+ 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++, Python or Java. 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 3 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 submitted to the lecturer via LMS. The zip file must be named with your Forename-Surname-Student ID, e.g. *<Joe-Smith-0123456>.zip*.

The deadline for submitting the assignment is: **Thursday 6th 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.