# HPC Twitter GeoProcessing

COMP90024 Cluster and Cloud Computing

Login: hqq

Student Number: 964325

Login: gey3

Student Number: 925156

## Abstract:

The main purpose of the project is to process the given json data for Twitter and identify Twitter activities around Melbourne as well as presenting data in an orderly form, such as the post number and the most commonly used hashtags in each given grid cell. Moreover, the process is optimized by using a high performance computer. In this project, we chose Python as our programming language to process json files and obtain information we need. With Spartan, we use multi-node and multi-core for parallel programming to improve program performance.

## Methodology:

### 1. Data Analysis

The melbGrid json file contains the id and coordinates information of sixteen grids of Melbourne, the bigTwitter json file contains a large number of tweets and relevant information such as the coordinates of each tweet, the tweet texts and so on. Python offers a json library with which we could parse a json file into a python dictionary or list.

### 2. Programing

### 2.1 geographical location processing

The first step is to access melbGrid file to get geographic location of each grid cell, with which we could learn in which grid each given tweet lies. First, we import json module in our program. Then create an empty dictionary **"all_grid"** to store border coordinates of each grid, after processing, the form is like: {A1: {xmin:14.81, xmax: 14.85, ymin: -37.65, ymax: -37.5}, A2: {……}}.

### 2.2 Parallelizing code

The bigTwitter json file contains a large number of tweets so that the size of this file is very big, it takes a long time to process them if only one processor is used. Spartan is High Performance Computing (HPC) system which enables multi processing to be implemented on such large volume of data. Mpi4py is a module for python which allows python to parallelize coding which is used in this application. In this application, we load in bigTwitter json file line by line and equally allocate each line into one of several cores to process the data by invoking function call process_data. For example, if we have 2n lines of file to process and 2 cores to use, we could give line1, 3, 5, 7……2n-1of the file to the first core and line 2, 4, 6, 8……2n to the second core. After all of the data is processed, each processor then holds a part of results, we chose to use method MPI.COMM_WORLD.gather to merge all results together. It is worth to mention that, the type of data we tried to gathered is python dictionary. So after gathering, all the data is just simply collected in one list which means we still need to use loops to get final result.

### 2.3 Twitter data process

At first, we created two empty dictionaries to store results and append grid ids as keys, the first one **"postnum"** is used to store number of posts from grid A1 to D5, meanwhile the second one **"postrank"** is to hold top 5 commonly used hashtags in each grid. Both results should be ordered after posts number.

As mentioned above, we process the bigTwitter file line by line and each line contains all information of one tweet, after parsing one line of bigTwitter json file, we extract the longitude and latitude coordinates, then exact grid for each tweet could be defined regarding to the grid division coordinates obtained from melbGrid json file, for the coordinates lies on border, we classify them into top/left grids. Every time we find the grid id of one tweet, we add one in its corresponding value in the **"postnum"** to count posts number. In each line, the key 'text' in 'doc' dictionary contains the actual tweet content, and we could search for sign '#' to find hashtags in this tweet and save in a dictionary as a key in **"postrank",** but we won't count a single '#' as a hashtag. Every time we find the same hashtag in same grid, we add one in the value of that hashtag to count occurrence of this hashtag.

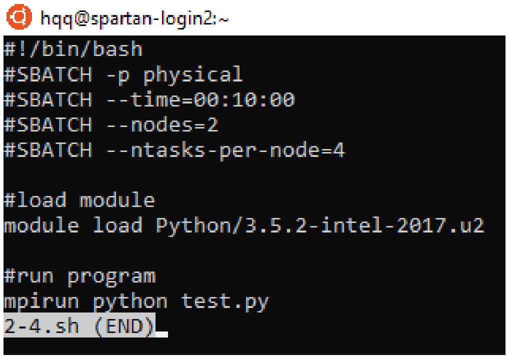
After gathering all results with method MPI.COMM_WORLD.gather from each cores, the form of dictionary **"postnum"** is like {A1: 40, A2: 500……} and the dictionary **"postrank"** is like {A1: {#melbourne:35, #beach: 60, ……}, A2: {} ……}.

### 2.4 Formatted output

The last step is formatting the results. For **"postnum"**, sorting it according to its values which stands for posts number and for **"postrank"**, the grid order is the same as **"postnum"**, but only show the top 5 most commonly hashtags in descending order.

## 3. Slurm scripts for Spartan

As Spartan is High Performance Computing (HPC) system, users are asked to use slurm scripts to submit our work to Spartan to ask for resource. For example, when we want to use 8 cores and every 4 cores are in one node, our script is as above in Img 2.1. we chose physical mode and set the maximum executing time as 10 minutes, for nodes and number of tasks on one node we set 2 and 4 respectively. Similarly, if we want to use 8 cores in 1 node, we could just change nodes=2 to nodes=1 and ntasks-per-node=4 to ntasks-per-node=8.



```
#!/bin/bash
#SBATCH -p physical
#SBATCH --time=00:10:00
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=4

#load module
module load Python/3.5.2-intel-2017.u2

#run program
mpirun python test.py
2-4.sh (END)
```

Img 1. slurm script

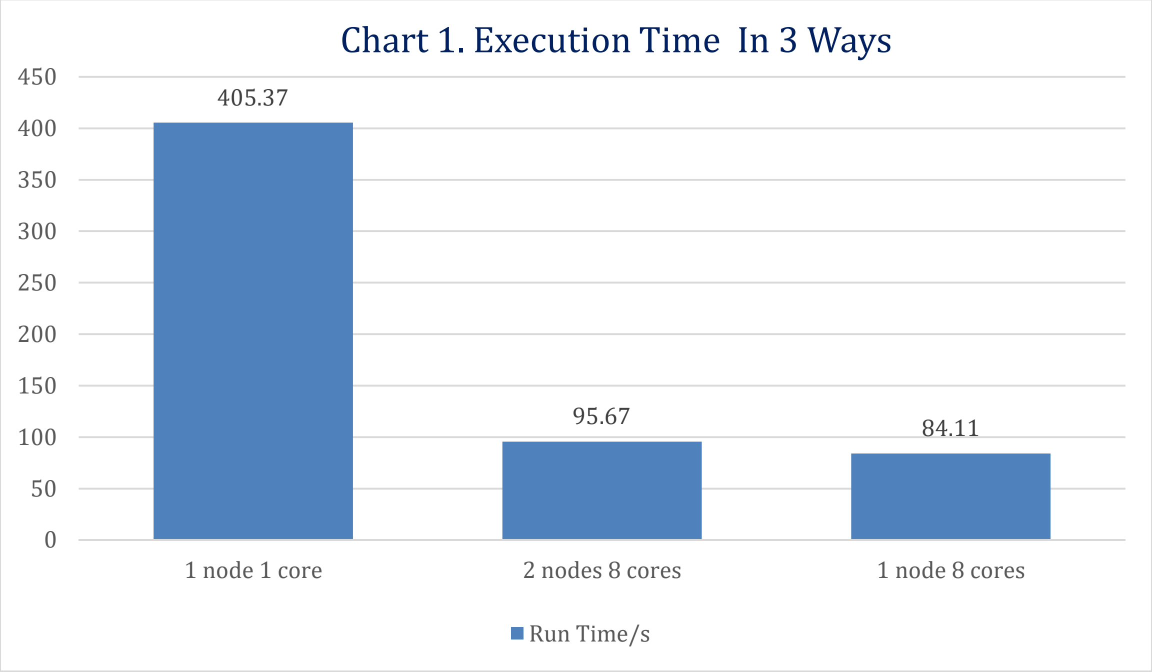
## Result

After executing our code in the mode of 1 node 1 core, 2 nodes 8 cores and 1 node 8 cores, we managed to get posts number of each grids ordered by those number, and the top 5 commonly used hashtags in each grid. The execution results of our program are listed in appendix.

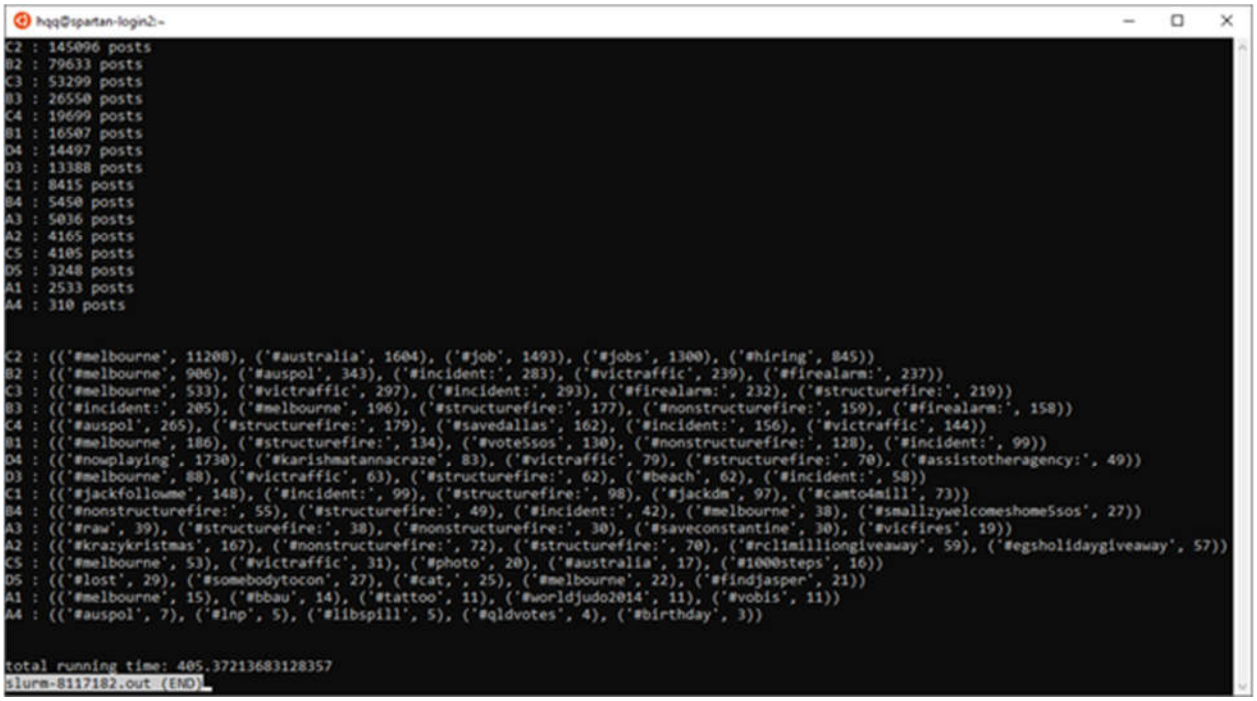
As shown in the bar chart blow, executing our program with 1 node and 1 core takes about 405.37, time for 2 nodes 8 cores, 1 node 8 cores are 95.67s and 84.11s respectively.

Comparing with 1 node 1 core, processing time of 1 node 8 cores is nearly 4.82 times faster, which means within certain range, multi-core processing can improve performance, but number of cores and performance may not be linear correlation, because in multi core mode, extra methods and functions are used to allocate data to different cores and gathering data after it is processed.

Although the mode 2 nodes with 4 nodes in each node have same amount of cores as 1 node 8 cores mode, but the performance is slightly worse, the reason for this may related to data transmission between 2 nodes. During data transmitting between two different nodes with hierarchical theorem, there would have some overheads produced when data is transmitted from layer to layer and it takes time to transmit data between two locations.



# Appendix

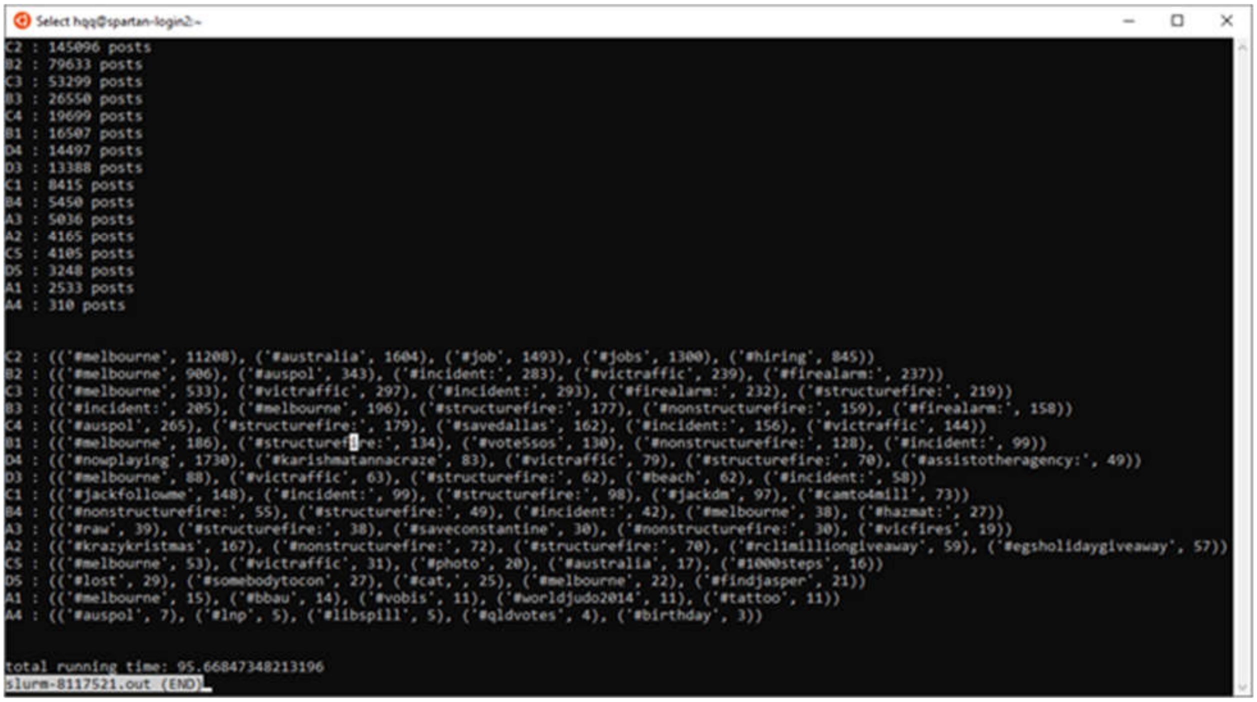


```
C2 : 145096 posts
B2 : 79633 posts
C3 : 53299 posts
B3 : 26550 posts
C4 : 19699 posts
B1 : 16507 posts
D4 : 14497 posts
D3 : 13388 posts
C1 : 8415 posts
B4 : 5450 posts
A3 : 5036 posts
A2 : 4165 posts
C5 : 4105 posts
D5 : 3248 posts
A1 : 2533 posts
A4 : 310 posts


C2 : (('#melbourne', 11208), ('#australia', 1604), ('#job', 1493), ('#jobs', 1300), ('#hiring', 845))
B2 : (('#melbourne', 906), ('#auspol', 343), ('#incident:', 283), ('#victraffic', 239), ('#firealarm:', 237))
C3 : (('#melbourne', 533), ('#victraffic', 297), ('#incident:', 293), ('#firealarm:', 232), ('#structurefire:', 219))
B3 : (('#incident:', 205), ('#melbourne', 196), ('#structurefire:', 177), ('#nonstructurefire:', 159), ('#firealarm:', 158))
C4 : (('#auspol', 265), ('#structurefire:', 179), ('#savedallas', 162), ('#incident:', 156), ('#victraffic', 144))
B1 : (('#melbourne', 186), ('#structurefire:', 134), ('#vote5sos', 130), ('#nonstructurefire:', 128), ('#incident:', 99))
D4 : (('#nowplaying', 1730), ('#karishmatannacraze', 83), ('#victraffic', 79), ('#structurefire:', 70), ('#assistotheragency:', 49))
D3 : (('#melbourne', 88), ('#victraffic', 63), ('#structurefire:', 62), ('#beach', 62), ('#incident:', 58))
C1 : (('#jackfollowme', 148), ('#incident:', 99), ('#structurefire:', 98), ('#jackdm', 97), ('#camto4mill', 73))
B4 : (('#nonstructurefire:', 55), ('#structurefire:', 49), ('#incident:', 42), ('#melbourne', 38), ('#smallzywelcomeshome5sos', 27))
A3 : (('#raw', 39), ('#structurefire:', 38), ('#nonstructurefire:', 30), ('#saveconstantine', 30), ('#vicfires', 19))
A2 : (('#krazykristmas', 167), ('#nonstructurefire:', 72), ('#structurefire:', 70), ('#rcl1milliongiveaway', 59), ('#egsholidaygiveaway', 57))
C5 : (('#melbourne', 53), ('#victraffic', 31), ('#photo', 20), ('#australia', 17), ('#1000steps', 16))
D5 : (('#lost', 29), ('#somebodytocon', 27), ('#cat,', 25), ('#melbourne', 22), ('#findjasper', 21))
A1 : (('#melbourne', 15), ('#bbau', 14), ('#tattoo', 11), ('#worldjudo2014', 11), ('#vobis', 11))
A4 : (('#auspol', 7), ('#lnp', 5), ('#libspill', 5), ('#qldvotes', 4), ('#birthday', 3))


total running time: 405.37213683128357
slurm-8117182.out (END)
```

Img 1 Execution of 1 node 1 core

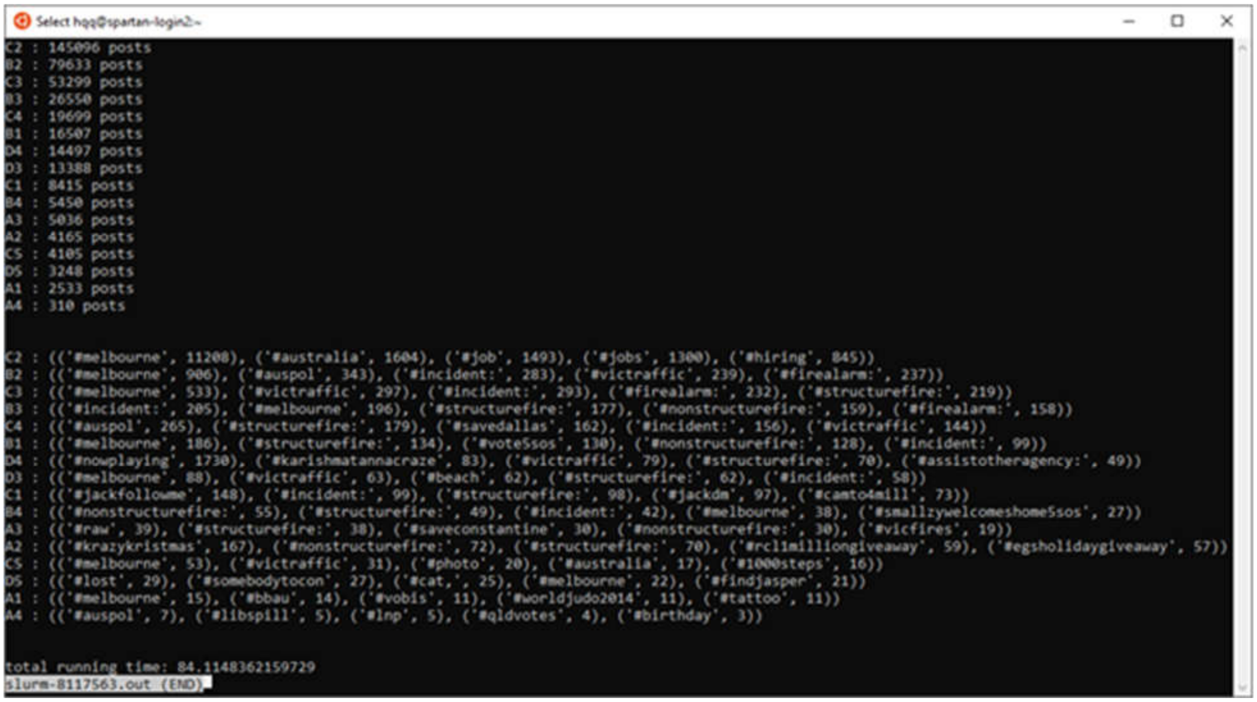


```
C2 : 145096 posts
B2 : 79633 posts
C3 : 53299 posts
B3 : 26550 posts
C4 : 19699 posts
B1 : 16507 posts
D4 : 14497 posts
D3 : 13388 posts
C1 : 8415 posts
B4 : 5450 posts
A3 : 5036 posts
A2 : 4165 posts
C5 : 4105 posts
D5 : 3248 posts
A1 : 2533 posts
A4 : 310 posts


C2 : (('#melbourne', 11208), ('#australia', 1604), ('#job', 1493), ('#jobs', 1300), ('#hiring', 845))
B2 : (('#melbourne', 906), ('#auspol', 343), ('#incident:', 283), ('#victraffic', 239), ('#firealarm:', 237))
C3 : (('#melbourne', 533), ('#victraffic', 297), ('#incident:', 293), ('#firealarm:', 232), ('#structurefire:', 219))
B3 : (('#incident:', 205), ('#melbourne', 196), ('#structurefire:', 177), ('#nonstructurefire:', 159), ('#firealarm:', 158))
C4 : (('#auspol', 265), ('#structurefire:', 179), ('#savedallas', 162), ('#incident:', 156), ('#victraffic', 144))
B1 : (('#melbourne', 186), ('#structurefire:', 134), ('#vote5sos', 130), ('#nonstructurefire:', 128), ('#incident:', 99))
D4 : (('#nowplaying', 1730), ('#karishmatannacraze', 83), ('#victraffic', 79), ('#structurefire:', 70), ('#assistotheragency:', 49))
D3 : (('#melbourne', 88), ('#victraffic', 63), ('#structurefire:', 62), ('#beach', 62), ('#incident:', 58))
C1 : (('#jackfollowme', 148), ('#incident:', 99), ('#structurefire:', 98), ('#jackdm', 97), ('#camto4mill', 73))
B4 : (('#nonstructurefire:', 55), ('#structurefire:', 49), ('#incident:', 42), ('#melbourne', 38), ('#hazmat:', 27))
A3 : (('#raw', 39), ('#structurefire:', 38), ('#saveconstantine', 30), ('#nonstructurefire:', 30), ('#vicfires', 19))
A2 : (('#krazykristmas', 167), ('#nonstructurefire:', 72), ('#structurefire:', 70), ('#rcl1milliongiveaway', 59), ('#egsholidaygiveaway', 57))
C5 : (('#melbourne', 53), ('#victraffic', 31), ('#photo', 20), ('#australia', 17), ('#1000steps', 16))
D5 : (('#lost', 29), ('#somebodytocon', 27), ('#cat,', 25), ('#melbourne', 22), ('#findjasper', 21))
A1 : (('#melbourne', 15), ('#bbau', 14), ('#vobis', 11), ('#worldjudo2014', 11), ('#tattoo', 11))
A4 : (('#auspol', 7), ('#lnp', 5), ('#libspill', 5), ('#qldvotes', 4), ('#birthday', 3))


total running time: 95.66847348213196
slurm-8117521.out (END)
```

Img 2 Execution of 2 nodes 8 cores



```
C2 : 145096 posts
B2 : 79633 posts
C3 : 53299 posts
B3 : 26550 posts
C4 : 19699 posts
B1 : 16507 posts
D4 : 14497 posts
D3 : 13388 posts
C1 : 8415 posts
B4 : 5450 posts
A3 : 5036 posts
A2 : 4165 posts
C5 : 4105 posts
D5 : 3248 posts
A1 : 2533 posts
A4 : 310 posts


C2 : (('#melbourne', 11208), ('#australia', 1604), ('#job', 1493), ('#jobs', 1300), ('#hiring', 845))
B2 : (('#melbourne', 906), ('#auspol', 343), ('#incident:', 283), ('#victraffic', 239), ('#firealarm:', 237))
C3 : (('#melbourne', 533), ('#victraffic', 297), ('#incident:', 293), ('#firealarm:', 232), ('#structurefire:', 219))
B3 : (('#incident:', 205), ('#melbourne', 196), ('#structurefire:', 177), ('#nonstructurefire:', 159), ('#firealarm:', 158))
C4 : (('#auspol', 265), ('#structurefire:', 179), ('#savedallas', 162), ('#incident:', 156), ('#victraffic', 144))
B1 : (('#melbourne', 186), ('#structurefire:', 134), ('#vote5sos', 130), ('#nonstructurefire:', 128), ('#incident:', 99))
D4 : (('#nowplaying', 1730), ('#karishmatannacraze', 83), ('#victraffic', 79), ('#structurefire:', 70), ('#assistotheragency:', 49))
D3 : (('#melbourne', 88), ('#victraffic', 63), ('#beach', 62), ('#structurefire:', 62), ('#incident:', 58))
C1 : (('#jackfollowme', 148), ('#incident:', 99), ('#structurefire:', 98), ('#jackdm', 97), ('#camto4mill', 73))
B4 : (('#nonstructurefire:', 55), ('#structurefire:', 49), ('#incident:', 42), ('#melbourne', 38), ('#smallzywelcomeshome5sos', 27))
A3 : (('#raw', 39), ('#structurefire:', 38), ('#saveconstantine', 30), ('#nonstructurefire:', 30), ('#vicfires', 19))
A2 : (('#krazykristmas', 167), ('#nonstructurefire:', 72), ('#structurefire:', 70), ('#rcl1milliongiveaway', 59), ('#egsholidaygiveaway', 57))
C5 : (('#melbourne', 53), ('#victraffic', 31), ('#photo', 20), ('#australia', 17), ('#1000steps', 16))
D5 : (('#lost', 29), ('#somebodytocon', 27), ('#cat,', 25), ('#melbourne', 22), ('#findjasper', 21))
A1 : (('#melbourne', 15), ('#bbau', 14), ('#vobis', 11), ('#worldjudo2014', 11), ('#tattoo', 11))
A4 : (('#auspol', 7), ('#libspill', 5), ('#lnp', 5), ('#qldvotes', 4), ('#birthday', 3))


total running time: 84.1148362159729
slurm-8117563.out (END)
```

Img 3. Execution of 1 node 8 cores