Cluster and Cloud Computing Assignment 1 – Social Media Analysis

Jiqiang Chen
student Id : 1171420

Aobo Li
student Id : 1172339

April 9, 2023

1 Introduction

This project aim to implement a parallelized application leveraging the University of Melbourne HPC facility SPARTAN. A large Twitter dataset and a file containing the suburbs locations and Greater Capital cities of Australia is used to analyse following objective:

- 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
- identify the users that have tweeted from the most different Greater Capital cities

2 Method

2.1 Location Matching

The location information in the twitter file is locations split by comma in string. Two examples in the file: "Central Coast, New South Wales", "Australia". Our approach to link with Sal gcc data using Re package, for single location information, our procedure is:

1. Lowercase all twitter information and split them by comma, use a list to store the location names after splitting
2. For efficiency purpose, check if the first element in the list is in the suburb level, for example, if it was 'Australia', then there is no suburb information, we will skip those records. Most of the twitter information comes from CBD of the Greater Capital cities. The gcc for CBD is also stored in a dictionary, so we can match to the CBD
3. If we can not find the gcc value by above steps, we have to use Re package to syntax match with the suburb data in Sal file. To improve efficiency, we find if there contains a state information so we can shrink down the numbers of possible suburbs.
4. If possible suburbs is matched in the Sal data with the state information stored in location, we can return its gcc value. Otherwise, we go through all possible match of suburbs to check if they have same gcc value. Return the gcc value if condition is meet.

2.2 Reading large JSON file

Read the large twitter JSON file takes enormous memory space. The memory size is not enough to read it all into memory. We considered two ideas to encounter the problem: 1. split the JSON file in to chucks 2. load the file by lines, extract only useful data. We tried splitting the JSON file into chunks by filesize, however it creates more rows than loading the file with pandas. If we read the file by line separates "`\n`", it read less rows. Hence we used method 2 for reading the data.

2.3 HPC

To complete the twitter task and test our parallelization efficiency, High performance computers from Melbourne HPC facility SPARTAN is used. 3 different SLURM scripts are written in order to use three different parallelization set up:

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

example of a SLURM script for 1 node and 8 cores

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=8
#SBATCH --output=18out.txt
#SBATCH --error=18error.txt
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-core=4G
#SBATCH --time=0:15:00

# The modules to load:
module load python/3.7.4
module load mpi4py/3.0.2-timed-pingpong
module load foss/2019b
source ~/virtualenv/python3.7.4/bin/activate

mpirun -np 8 python ass1.py bigTwitter.json
my-job-stats -a -n -s
```

2.4 parallelization

MPI4PY is used in this project to achieve parallelization. The brief idea of our parallelization is: After reading the JSON file by root node, we split the twitter data into equal size of chucks(numbers of chuck is equal to numbers of node we have). Then send the chucks to different nodes, processing the data and apply the location matching in parallel. Collect the result of all the nodes to root node and print the result. Figure 1 is a example of the parallelization for 4 cores task.

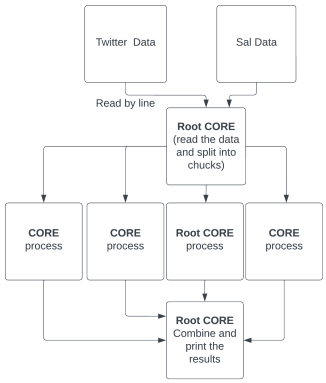

Figure 1: diagram of 4 cores parallelization

3 Result

3.1 parallelization result

Result(in second)	1 node 1 cores	1 node 8 cores	2 node 8 cores
Reading time	206	205	228
Process time	516	68	71
Total time	734	291	318

Our code consists of serial and parallel parts. Reading the JSON file line by line is serial, and we can see that the performance of different cores is basically the same. Finding the GCC of each location is parallelized, and the processing time for this part of the script decreased dramatically, with a speedup of 7.58 (1 node 8 core) and 7.26 (2 nodes 4 core each) for.

Based on Amdahl's Law: $\frac{\sigma+\pi}{\sigma+\pi/N}$, speed up of the program is limited by the non- parallelizable part of the program(reading the json file). Total speedup of our program should be 2.675. Actual speed up of our program achieved close to this value, with a total speed of 2.52 for 1 node with 8 cores and 2.31 for 2 nodes with 8 cores.

It was observed that using 2 nodes with 8 cores each resulted in slower processing compared to a single node with 8 cores. This could be attributed to the fact that while data can be transferred between nodes, it takes longer for the two nodes to interact and communicate with each other as opposed to using a single node with all the cores. For instance, when reading the file bigTwitter.json, it took 23 seconds longer to process on the 2 node 8 core setup than on the 1 node 8 core setup, and 3 seconds longer when finding the gccs.

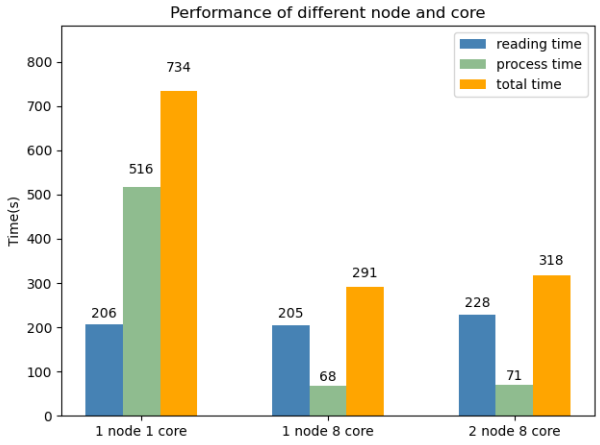

Figure 2: Time of different parallelization method

Figure 3 presents the findings of the tweet analysis conducted on the bigTwitter.json file. The results indicate that the majority of tweets were generated in Greater Melbourne (2.28 million) and Greater Sydney (2.20 million), which aligns with our expectation given that these cities are the most populated. Additionally, author ID 1498063511204761601 had the highest number of tweets with 68477. Notably, the top 10 authors all tweeted in 8 different greater capital cities, with none of them tweeting from other regions. For a more detailed presentation of the results, please refer to Figure 3 on the following page.

```
Greater Capital City        Number of Tweets Made
-------------------------  ----------------------
2gmel(Greater Melbourne)                  2283562
1gsyd(Greater Sydney)                     2204728
3gbri(Greater Brisbane)                    859862
5gper(Greater Perth)                       589375
4gade(Greater Adelaide)                    464050
8acte(Greater Canberra)                    202434
6ghob(Greater Hobart)                       90762
7gdar(Greater Darwin)                       46393
9oter(Other Territory)                        160

Rank | Author ID           | Number of Tweets Made
-----------------------------------------------------
#1   | 1498063511204761601 | 68477
#2   | 1089023364973219840 | 28128
#3   | 826332877457481728  | 27718
#4   | 1250331934242123776 | 25350
#5   | 1423662808311287813 | 21034
#6   | 1183144981252280322 | 20765
#7   | 1270672820792508417 | 20503
#8   | 820431428835885059  | 20063
#9   | 778785859030003712  | 19403
#10  | 1104295492433764353 | 18781


Rank | Author ID           | Number of Unique City Locations and
#Tweets
---------------------------------------------------------------------
----
#1   | 1429984556451389440 | 8 (#1918 tweets - 1878gmel, 2gade,
13acte, 7gper, 10gsyd, 6gbri, 1ghob, 1gdar)
#2   | 702290904460169216  | 8 (#1223 tweets - 263gmel, 128gade,
47acte, 235gbri, 42ghob, 21gdar, 333gsyd, 154gper)
#3   | 17285408            | 8 (#1190 tweets - 1042gsyd, 40gbri,
23acte, 60gmel, 11ghob, 3gade, 4gdar, 7gper)
#4   | 87188071            | 8 (#398 tweets - 28gade, 65gbri,
51gper, 114gsyd, 86gmel, 34acte, 15ghob, 5gdar)
#5   | 1361519083          | 8 (#266 tweets - 36gmel, 193gdar,
18gsyd, 9gade, 1gper, 6acte, 2ghob, 1gbri)
#6   | 502381727           | 8 (#250 tweets - 214gmel, 8gbri, 8ghob,
3gper, 4gade, 2gsyd, 1gdar, 10acte)
#7   | 921197448885886977  | 8 (#208 tweets - 37gbri, 24gade,
28gper, 49gsyd, 58gmel, 7acte, 4ghob, 1gdar)
#8   | 601712763           | 8 (#146 tweets - 44gsyd, 10acte,
39gmel, 1gdar, 14gper, 8ghob, 19gade, 11gbri)
#9   | 2647302752          | 8 (#80 tweets - 32gbri, 3gdar, 3gade,
4gper, 4acte, 13gsyd, 16gmel, 5ghob)
#10  | 322866759           | 8 (#45 tweets - 31acte, 4gsyd, 1gper,
4gmel, 1gade, 2gbri, 1ghob, 1gdar)
```

Figure 3: Results

4 Conclusion

To conclude, 1 node 8 core shows better performance among the three configurations mentioned in the requirements, using the least time to complete the required task. Our results matches Amdahl's law and speeds up based on the formula (Total time is not read time + process time since printing data requires some conditioning to print out the results). Eventually, the speed up of a script are based on the proportion of the script that can be parallelized.