COMP90024 - Cluster and Cloud Computing

Assignment 1 - Social Media Analytics

Jiahao Chen 1118749, Zhiquan Lai 1118797

Introduction

The aim of this project is to explore the utilization of parallelized applications for large-scale data processing and compare the performance of different numbers of nodes or cores.

Methodology

Figure 1 depicts the comprehensive workflow of our program. Given n processors, the size of the Twitter dataset is initially determined, and each processor is assigned the starting and ending pointers of the file chunk to be processed, with the chunk size being evenly divided. Upon completion of their respective tasks, the data is consolidated at processor 0 to generate and present the results.

Figure 2 illustrates the procedure employed by each processor to handle the data, focusing specifically on the parallelizable aspects, which encompass data loading and analysis. Given the considerable size of the Twitter dataset, a conservative assumption is made that 95% of the program can be executed in parallel. Assuming the availability of 8 processors, Amdahl's law yields a speed-up factor of *1 / (0.05 + (0.95 / 8)) = 6.7x*, whereas Gustafson-Barsis's Law produces a factor of *0.05 + 0.95(8 - 0.05) = 7.6x*.

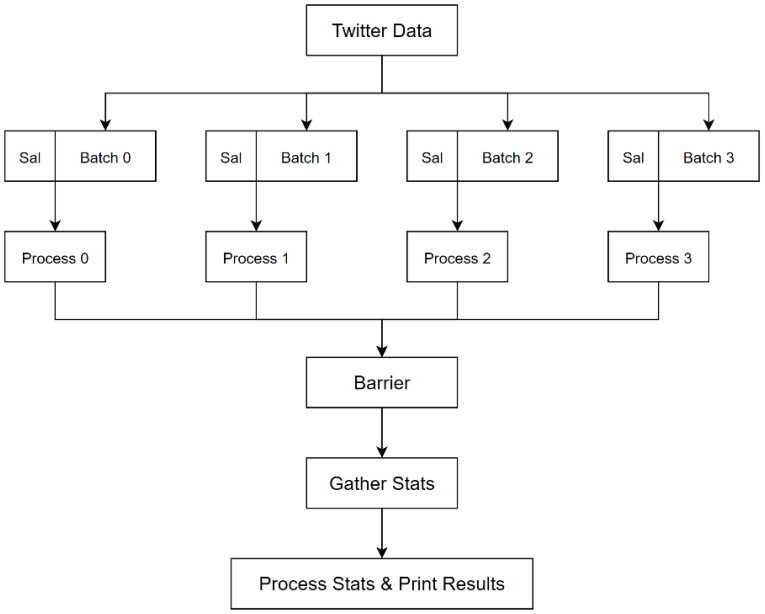

Figure 1 - Program Flow

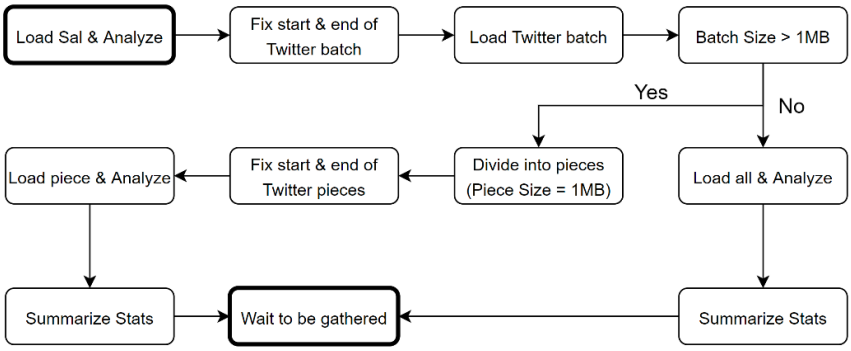

Figure 2 - Process Flow

Results

Task 1: Identify the Twitter accounts (users)that have made the most tweets

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

Task 2: Count the number of different tweets made in the Greater Capital cities of Australia

Greater Capital City	Number of Tweets Made
1gsyd (Greater Sydney)	2116810
2gmel (Greater Melbourne)	2286235
3gbri (Greater Brisbane)	857923
4gade (Greater Adelaide)	462596
5gper (Greater Perth)	589322
6ghob (Greater Hobart)	90672
7gdar (Greater Darwin)	46357
8acte (Greater Canberra)	202477

Task 3: Identify the users that have tweeted from the most different Greater Capital cities

Rank	Author Id	Number of Unique City Locations and #Tweets
#1	1429984556451389440	8(#1920 tweets - #10gsyd, #1880gmel, #6gbri, #2gade, #7gper, #1ghob, #1gdar, #13acte)
#2	17285408	8(#1208 tweets - #1060gsyd, #60gmel, #40gbri, #3gade, #7gper, #11ghob, #4gdar, #23acte)
#3	702290904460169216	8(#1129 tweets - #288gsyd, #252gmel, #218gbri, #113gade, #151gper, #41ghob, #20gdar, #46acte)
#4	87188071	8(#395 tweets - #110gsyd, #86gmel, #65gbri, #29gade, #51gper, #15ghob, #5gdar, #34acte)
#5	774694926135222272	8(#272 tweets - #37gsyd, #38gmel, #37gbri, #28gade, #34gper, #36ghob, #28gdar, #34acte)
#6	1361519083	8(#260 tweets - #12gsyd, #36gmel, #1gbri, #9gade, #1gper, #2ghob, #193gdar, #6acte)
#7	502381727	8(#250 tweets - #2gsyd, #214gmel, #8gbri, #4gade, #3gper, #8ghob, #1gdar, #10acte)
#8	921197448885886977	8(#205 tweets - #46gsyd, #58gmel, #37gbri, #24gade, #28gper, #4ghob, #1gdar, #7acte)
#9	601712763	8(#146 tweets - #44gsyd, #39gmel, #11gbri, #19gade, #14gper, #8ghob, #1gdar, #10acte)
#10	2647302752	8(#80 tweets - #13gsyd, #16gmel, #32gbri, #3gade, #4gper, #5ghob, #3gdar, #4acte)

In task 3, users with the same amount of unique city locations are sorted based on their total number of tweets made in Greater Capital Cities.

Performance

As illustrated in Figure 3, utilizing 8 cores results in an approximate 7.5x speedup, which closely aligns with the theoretical outcome derived from Gustafson-Barsis's Law. Amdahl's law appears to be less accurate in this case, likely due to the immense size of the dataset. It was initially hypothesized that a 2N8C configuration would exhibit slower performance than a 1N8C setup, as cores situated on separate nodes might require additional time for intercommunication. Nevertheless, the observed performances are strikingly similar, falling within the acceptable margin of error.

Initially, a strategy of reading one tweet at a time was employed to circumvent potential out-of-memory errors, as dividing the file into smaller segments could disrupt the structure of the tweets. However, Figure 4 demonstrates that this approach is markedly inefficient, necessitating the exploration of alternative strategies for processing the data.

The choice of batch size limit is a point worth exploring in more detail, which will be discussed in the next section.

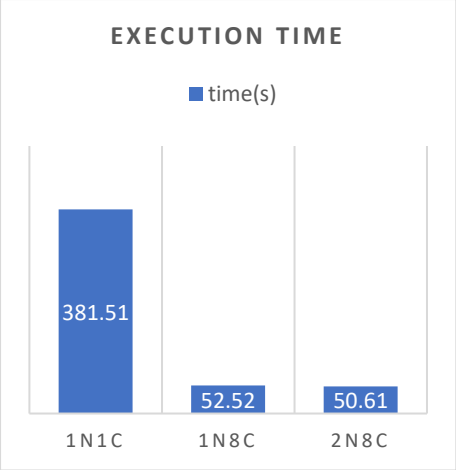

Figure 3 - Execution Time

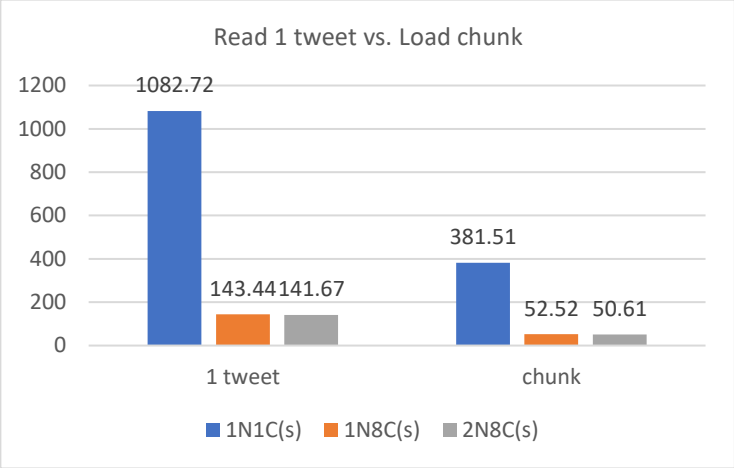

Figure 4 - 1 tweet vs. chunk

Discussion

Location Matching

Two primary challenges arise in the context of geographical information: the occurrence of repeated suburb names and the presence of incorrect or imprecise state names.

On the one hand, several suburbs in distinct states share identical names, exemplified by the existence of two suburbs named *'Falls Creek'* one in Victoria and another in New South Wales. To address this issue, the SAL dataset employs a dictionary structure with states and suburbs serving as keys and values respectively. Utilizing the state information as a key for retrieving the suburbs allows for the avoidance of duplicate suburb names.

On the other hand, certain state names are found to be incorrect or imprecise, such as instances where the state information is given as *'Perth (WA)'* instead of *'Western Australia'*. This challenge is surmounted by expanding the abbreviation within the parentheses using a dictionary to store all abbreviations corresponding to each state. Moreover, certain location entries consist solely of capital cities rather than state names, as exemplified by *'Bentleigh, Melbourne'*. To address this issue, a dictionary is constructed to establish a correspondence between capital cities and their respective state names.

Batch Size Limit

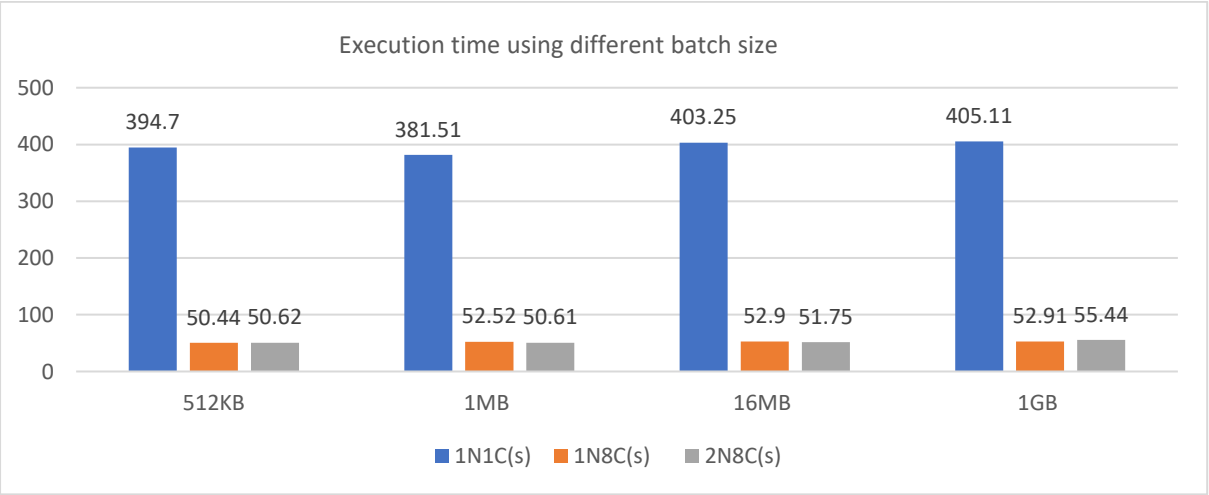

Figure 5 - Execution time using different batch size

As depicted in Figure 2, the batch will be partitioned into smaller segments if its size exceeds 1 megabyte. Numerous experiments have been conducted to investigate the optimal batch size limit. Hypothetically, a larger batch size might yield better performance, as it necessitates fewer data partitions and is potentially more

memory efficient. Nonetheless, as demonstrated in Figure 5, the optimal size appears to be 1 megabyte. Possible explanations for this observation are as follows:

Firstly, reading files in Python using file.read() may consume a considerable amount of time when processing large files, potentially impeding subsequent operations. Secondly, a smaller batch size might utilize the CPU cache more effectively, as it reduces the communication between the CPU and memory. Consequently, the observed results suggest that a batch size of 1 megabyte strikes the optimal balance between processing efficiency and memory management.

Slurm

```
#!/bin/bash
#SBATCH --job-name="Social_Media_Analysis"
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=8
#SBATCH --time=0-1:00:00
#SBATCH --output=1n8c.out

module load python/3.7.4
module load mpi4py/3.0.2-timed-pingpong

mkdir ~/virtualenv
virtualenv ~/virtualenv/python3.7.4
source ~/virtualenv/python3.7.4/bin/activate
pip install numpy
pip install pandas
pip install ijson

time mpirun -np 8 python3 main.py

deactivate

##DO NOT ADD/EDIT BEYOND THIS LINE##
##Job monitor command to list the resource usage
my-job-stats -a -n -s
```

Note

Instructions on running the program locally or on Spartan can be found in README included in the zip file.