**01**

****

**COMP90024 Cluster and Cloud Computing**

**Semester 1 2019**

**Assignment 1 - HPC Twitter GeoProcessing**

The following report describes the process taken while implementing a parallelized application leveraging the University of Melbourne HPC facility SPARTAN[[1]](#footnote-1) to read a large geocoded Twitter dataset “bigTwitter.json” to identify Twitter usage around Melbourne and the most frequently occurring hashtag per area.

**Parallelization** Approach

We adapted the Single-Program Multiple-Data with Master Worker approach. The job is divided to 3 main Tasks:

1

**Serial Operation** **#1**:

* 1. Read Melbourne Grid and store in memory
  2. Get Twitter file size and calculate the chunk block read byte offset based on number of processors available for parallelization (only the offset is being set at this stage to determine the byte block to be read by each parallel task)

Master

Core

* 1. Broadcast the Melbourne Grid, Chunk byte range per task, and file source path

All

Cores

2

**Parallel Operation:**

* 1. Read Twitter file based on the assigned offset
  2. Identify valid tweets, and calculate count and hashtags per grid block

3

**Serial Operation #2**:

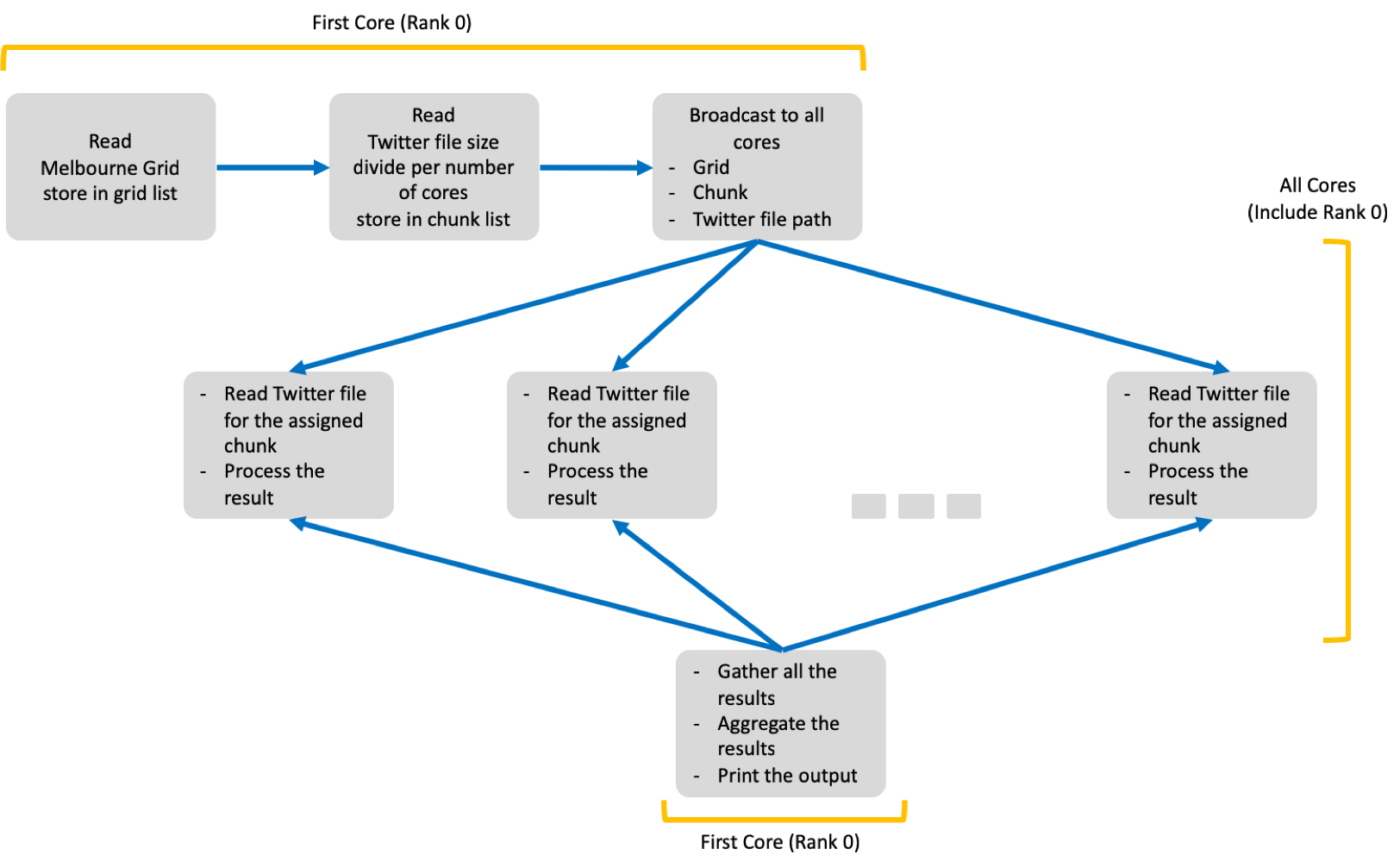
Master

Core

* 1. Gather and aggregate results from each parallel process
  2. Filter top 5 most tweeted hashtags per grid
  3. Print results

**02**

**Process** Illustration



This approach was adopted because it provides optimum performance improvement while increasing the number of parallel processes (nodes/cores) since the data is evenly split between processes. However, there are two main inevitable drawbacks with this approach; Firstly, the master process is a bottleneck. It is the main controller that should be up to split the tasks for each parallel process and later aggregate the results. Secondly, as explained in the findings section, the serial operation would slightly increase per core/node added (the increase is linearly proportional to the number of processes).

**Background** Work

Before comes to our final approach, we have tried several approaches. First, we try to read the whole twitter file in master core, scatter the data to all cores, and do aggregation in master core. This takes approximately 10 times slower than our final approach. Next is similar to the final approach, but data is split based on number of lines. This approach run slightly faster than the previous one. It would be significantly faster if number of lines is hard coded. However, the running time is still twice slower than the final approach. Thus, the final approach gives the best performance in terms of running time without any hard coding compared to other parallel approaches attempted.

**03**

**Important** Findings

**Scenario #1** : 1 Node 1 Core

**Scenario #2** : 1 Node 8 Cores

**Scenario #3** : 2 Nodes 8 Cores

Average run time per process

#3

#1

0.011

18.60

0.11

0.0009

140.93

0.10

0.010

19.30

0.11

#2

Serial 1 [0,015]

Parallel [0,141]

Serial 2 [0.98, 0.012]

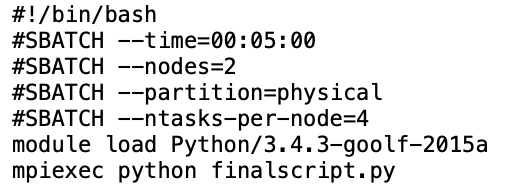
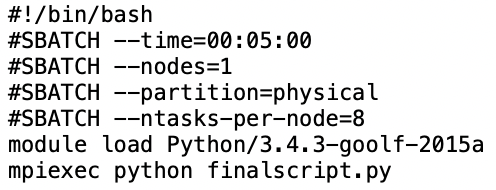
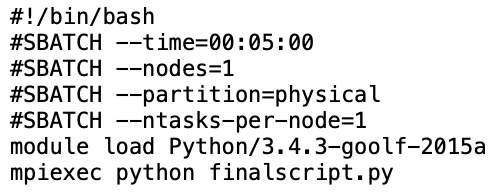
Scale (seconds)

Relative to running the job on SPARTAN using 1 node and 1 core (Scenario #1), we noticed a 5.37 times improvement while running the job using 1 node 8 eight cores (Scenario #2). Whereas there is 5.10 times improvement when running using 2 nodes and 4 cores (Scenario #3). The small variation of the performance between Scenario 2 and 3 denotes the extra network latency in communication between 2 nodes with physical partition[[2]](#footnote-2).

It is worth noting that we noticed when increasing the number of nodes/cores. Based on the task breakdown runtime of each operation (right side of the figure above), there is a noticeable increase in the first serial operation done by the Master process and a slight increase in the last serial operation. The first operation increase is due to the dynamic program logic to calculate the file read block size for each process; there is an extra overhead to read the last line per block based on the number of processes, to determine the end of line byte offset. Thus, adding more processes would lead to a linear increase of the serial line reads. The final serial operation increase is due to the gather process; with more nodes/cores added there will be an overhead for gathering and merging the processed data from each node. However, this increase is relatively considered as a small fraction comparing to the volume of the data which is insignificant.

**04**

**Invocation** Process

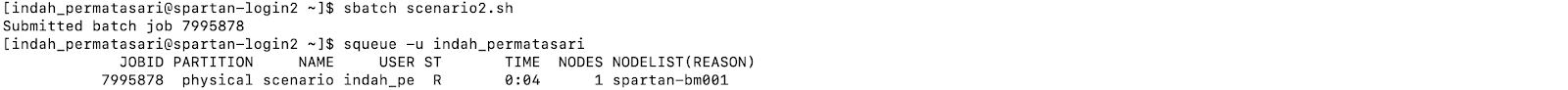


Scenario #1 : 1 Node 1 Core

Scenario #2 : 1 Node 8 Cores

Scenario #3 : 2 Nodes 8 Cores

The python script is stored as finalscript.py. We run on spartan using sbatch <scenario\_filename>, for example for scenario 2:



**Python Implementation** Decisions/Assumptions**:**

1. Based on the last assignment announcement. Only hashtags following (<space>#<string><space>) are considered.
2. The x and y coordinates for each twitter are retrieved from [doc-coordinates-coordinates] JSON element, and the hashtags are retrieved from doc-text.
3. Python seek function is used to set the offset position from which each process will read the file.
4. For calculating the offset position for reading the file we used (total file size/no of processes) as a starting boundary. We adjust the chunk boundaries to make sure each chunk begins at new line.
5. The bigTwitter.json file is opened as a binary file for faster processing. Each line read will be converted to a JSON object to retrieve the relevant tags for processing.

**Attachment** Description

|  |  |  |
| --- | --- | --- |
| No | File Name | Description |
| 1 | scenario1.sh | shell script file for scenario 1 |
| 2 | scenario2.sh | shell script file for scenario 2 |
| 3 | scenario3.sh | shell script file for scenario 3 |
| 4 | finalscript.py | python script for data processing |
| 5 | slurm-8031675-1x1Physical.out | output file from scenario 1 |
| 6 | slurm-8031631-1x8Physical.out | output file from scenario 2 |
| 7 | slurm-8031619-2x4Physical.out | output file from scenario 3 |
| 8 | scontrol show job 8031675.png | job status of scenario 1 |
| 9 | scontrol show job 8031631.png | job status of scenario 2 |
| 10 | scontrol show job 8031619.png | job status of scenario 3 |
| 11 | statistics.xlsx | runtime details per core |

1. The University of Melbourne, “Spartan Documentation,” *Spartan Documentation*. [Online]. Available: https://dashboard.hpc.unimelb.edu.au/. [↑](#footnote-ref-1)
2. The University of Melbourne, “Spartan Documentation,” *Spartan Documentation*. [Online]. Available: https://dashboard.hpc.unimelb.edu.au/faq/ [↑](#footnote-ref-2)