COMP90024 Cluster and Cloud Computing

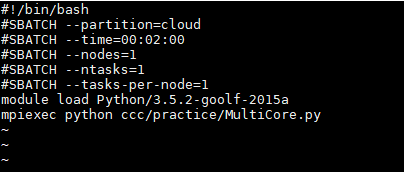
Assignment 1 Report

**Introduction:**

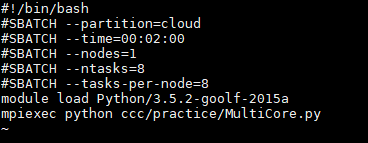
This report gives an introduction of the application used for twitter data analysis. The first section shows the scripts used for job submission, then, the second section introduces the code implementation of the parallel computing and the last section analysis the performances of the program.

**Section 1: Scripts used for job submission**

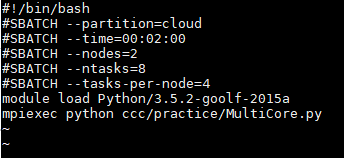
**The scripts used for submitting jobs to Spartan can be seen from figure 1.1 to figure 1.3 below.**



**Figure 1.1:** script for job on 1 node with 1 core



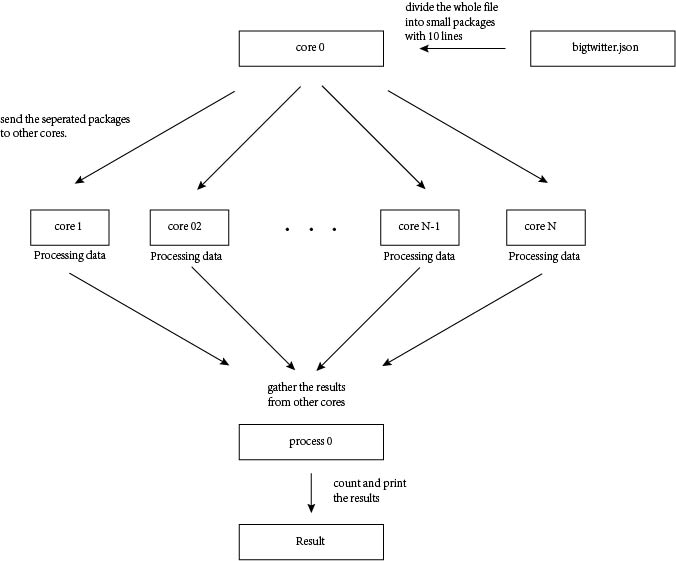
**Figure 1.2:** script for job on 1 node with 8 core



**Figure 1.3:** script for job on 2 nodes with 8 core

**Section 2: Approaches for Parallelism**

The approach used in this application for the implementation of the parallel computing is SPMD (Single-Program Multiple-Data) model, which can be seen in figure 2 below.



**Figure 2:** Approach used for the implementation of parallel computing

Considering the large size of the *bigTwitter.json*, it cannot be directly loaded. Thus, we use one core acting as the main core to separate the whole *bigTwitter.json* into small packages. Each package contains 10 lines of data. The number of lines in each package is decided according to the ratio of its transmission time to its processing time. These packages will be sent to other cores for data processing by the main core using comm.send method. If the last part of the *bigTwitter.json* contains less than 10 lines, it will also be sent to core 1.

For each package received from the main core, other cores will load the file, extract the geographical coordinates, classify it into corresponding regions, and record its tags.

After the whole *bigTwitter.json* file was assigned to other cores and finished processing, the main core will send a flag to other cores to stop them from waiting new packages. By using comm.gather() method to get the results from other cores, main core will count and print the total twitter numbers in each region, calculate the top 5 tags used in each region, and print them.

This approach has been improved for multiple times in order to achieve HPC (High-performance Computing). For example, the number of the lines in each package was changed from 100 to 10 as we found the processing time is longer than our prediction; the json.loads method was moved from the main core to other cores as its time consuming, and so on.

Other approaches like using all cores to read different lines in *bigTwitter.json* has also been considered. However, the current approach was found to be more efficient during several experiments.

**Section 3: Performances analysis**

The performances of this application on 1 node 1 core, 1 node 8 cores, and 2 nodes 8 cores can be seen in chart 3 below.

**Chart 3:** The time for executions on 1 node 1 core, 1 node 8 cores, and 2 nodes 8 cores

From chart 3, we can find out that the application on 1 node 8 cores has the best performance, which is about 3.3 times faster than the application on 1 node 1 cores. Then comes the application on 2 nodes 8 cores, about 7 seconds slower than the application on 1 node 8 cores, and about 3 times faster than the application on 1 node 1 core.

The reason why the application on 1 node 8 cores is not 8 times faster than the application on 1 node 1 core is that not every part of the code can be executed paralleled. For this particular approach, the distribution and gather of the data, the count of the total numbers of the twitters in each region, and the calculation of the top 5 tags are all procedures only executed on the main core, which consume a large amount of the total time and influence the efficiency.

The reason for the 7 seconds gap between the application on 1 node 8 cores and the application on 2 nodes 8 cores might cause by the longer time needed for the communications between cores in different nodes compared with the cores in the same node, as they have longer delays.