Cluster and Cloud Computing Assignment 1
HPC Instagram GeoProcessing

Yan Jiang 816920

1. Introduction:

With the development of big data, the size of the data to be processed is larger and larger, which makes higher demands on the speed of file processing. The speed of single-core processing was not enough to meet the current needs. Therefore, the parallel multicore method is used to improve the processing speed of IO tasks. The assignment is required to design a parallel method to handle millions of Instagram posts. The experimenter is asked to collect statistics on the number of Instagram post in different blocks of Melbourne. The report will introduce the dataset in this project. Then, the report will provide more details on demonstrating the method used in this project. Next, the report will analyze the result of different configure.

2. Dataset

The project provides a file named melbGrid to separate Melbourne area in 16 grids based on longitude and latitude. The id, the maximum and minimum values of latitude and longitude of each gird are encapsulated in the json format. The project provides different sizes of Instagram posts including tinyInstagram, smallInstagram and bigInsntagram.

3. Methodology

3.1 operating environment

The program will be uploaded to Spartan to test its performance. Spartan is a high-performance computing platform, which is able to provide a Linux hpc environment for researchers to run their programs.

3.2 Tools:

The program is written in python. Package mpi4py is used to pass python data structure between processes (or multiple cpus).

3.3 Slurm files:

```
#!/bin/bash
#SBATCH -p physical
#SBATCH --output=1n8c.txt
#SBATCH --nodes=1
#SBATCH --ntasks=8
#SBATCH --time=00:15:00

module load Python/3.4.3-goolf-2015a
mpiexec python cloudpal.py bigInstagram.json
```

Figure 1: The figure of slurm file

The slurm file is used to set the configuration and distribute the assignment to a set of assigned nodes for execution. This project used three kind of configuration to execute the program. We only append the slurm file of 1 node 8 cores as an instance.

3.4 Procedure:

3.4.1 Load the melbGrid file

Firstly, the program creates a class to set the attributes of each grid such as id, minimum and maximum value of latitude and longitude and posts count. Then the program creates a list to set the initial value according to the loaded melbGrid data.

3.4.2 Build the MPI architecture:

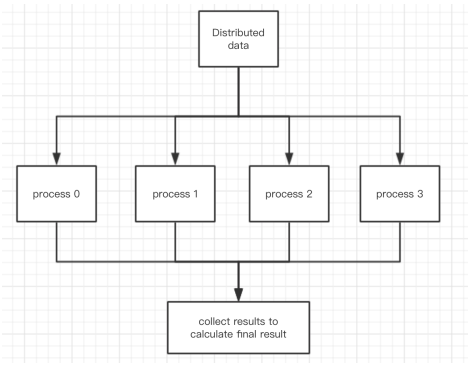

Figure 2: The figure of architecture

The parallel model chosen by this project is single-program multiple-data model. In this model, each process executes the same code and read all of data. However different process executes different part of data. The program will do a simple division of workload. The remainder of the number of rows divided by the number of processes is marked as the label. If the rank number equals this label, this process will process the row of data. The designer set rank 0 process as root process. After

processing all of data, all processes return their results to the root process. The root process gather all results and calculate the final result.

3.4.3 data processing

Data processing task is done on all process. Each process has a list to count the number of posts on each grid. Determining whether a post is in the grid is simple. Just determine if its coordinates are within the latitude and longitude of the grid. Iterate through each grid in the list, and if there is a grid that meets the requirements, then its count plus one.

4 Result and Discussion:

Rank the Grid	Rank the row	Rank the columns
C2: 174128	C-Row: 200653	Column2: 196817
B2: 22212	B-Row: 32733	Column3: 27151
C3: 18184	D-Row: 4953	Column4: 7152
B3: 6187	A-Row: 1337	Column1: 5168
C4: 4141		Column5: 3388
B1: 3311		
C5: 2605		
D3: 2313		
D4: 1857		
C1: 1595		
B4: 1023		
D5: 783		
A2: 477		
A3: 467		
A1: 262		
A4: 131		

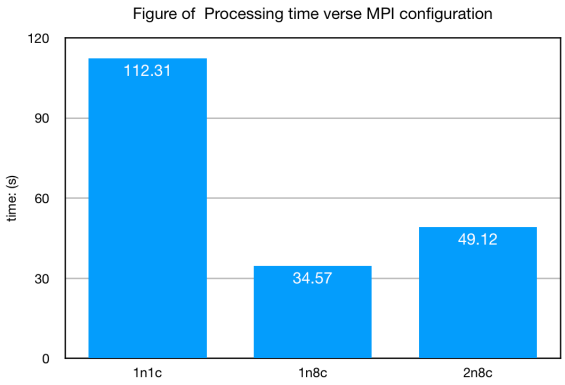

Figure 3: The figure of results

After implementing the program, the result is shown as above. The figure points out that the running speed of 1 node 8 cores is fastest. The running speed of 2 nodes 8 cores is faster than the speed of 1 node 1 core. It indicates that parallel programing is more efficient than single process. Multi-core processes different parts of the data can effectively reduce the idle time of each process. Therefore, it can maximize the use of the CPU resources. However, the figure points out that the speed of 2 nodes 8 cores is lower than the speed of 1 node 8 cores. There is a presumption that this result is caused by the time between nodes communicating. Different processes on the same node can exchange data by accessing memory. Processes on different nodes can only be exchanged through network communication. The speed of network communication data exchange is much slower than that of direct access to memory. This is the reason that the speed of 2 nodes 8 cores is slower than 1 node 8 cores.