Comp90024 Cluster and Cloud Computing Assignment 1
HPC Instagram GeoProcessing Report

Jiyu Chen 908066

Question Overview

Given a pre-defined Melbourne-grid dividing the metropolitan area of Melbourne in 16 squares on latitude and longitude based on https://epsg.io/3857 [1]. Implement a parallelized application leveraging the University of Melbourne HPC facility SPARTAN [2] and summarize the Instagram post hotspots in the grid.

Data

File <bigInstagram.json> is a large geocoded Instagram dataset in json format which is treated as sample collection. Samples of locations and coordinates are extracted and studied. However, locations are not truly compatible with coordinates which means in some samples, coordinates may indicate somewhere else whereas the location tag indicates Melbourne. The location can be fake. Therefore, we only take coordinates into account and ignore other geo-information. An ill-json format in few samples is another problem which may cause a parse error to our json-relied system. A third problem is there are a few samples fit right on the edge of the grid which need to be considered which blocks they fit in. The solution is to fit these samples in left-upper grid. Considering the number of samples are enormous enough in generating a statistical hotspots overview, biases caused by ignoring samples that are ill-json formatted or ignoring samples fit right on the left most and up most edge as they will exit the grid according to the left-upper rule is tolerable.

Method

Spartan

Spartan is a IaaS cluster and computing system. Allocation of computation resources and requirement of services is interacted via slurm scripts. In this project, three slurm scripts requesting resources of 1 node 1 core, 2 nodes 8 core and 4 cores per node, 1 node 8 core. This project will compare and evaluate the performance of application in these three computational situations and analysis their results.

Code

Python and mpi4py is used in our application. Considering the dataset is enormous so the application loads each line from sample instead of loading the whole dataset into memory which will cause an overflow.

Parallel Approach

Two methods included in mpi4py is cited. The scatter(para1, para2) method is used to divide an array of objects stored in para1 and the process (para2)will scatter it into other processes. The gather() method is used to gather objects from other process to one typical process and form into a list. In application, master process is implemented to do the scatter and gather job. One of the problem is the number of objects that waiting to be scattered have to equals to the number of processes. Therefore, we implement a linear approach. We will let master process to get the number of running process and to read a same number of samples and scatter them to all the other processes. E.g. if the number of running process is 4, master will read 4 samples each time and scatter the sample to other process include itself. Then master continually read another 4 lines, scatter them and process. Therefore, the dataset will only be traversed once. After receiving data from master, each process will call json.loads() and extract the coordinate information to fit the grid. A while loop is implemented for this linear

reading and scattering approach, and the break condition in one of the processes is when a sample is None. The trick is master process will append None object when it has finished reading the dataset but there are still having slaves running and waiting for data. Hence, a none object padding will cause a break to all the running processes. And then, the master can call gather() method to get all fitted grids and move forward to reduce.

Reduce Approach

Before introducing the reduce method which is used to transform the disperse gathered data to a unified result, I will introduce the data structure of the gathered data and grid. The origin Melbourne grid data is a json file which takes O(n) traversing time complexity each time as we have to traverse through A1 block to D5 block while processing each sample. To improve this, we transform the grid into a python dictionary format as {coordinates tuple: block}. Another design is each process will create their own {block: count} dictionary storing total count of Instagram in different blocks which were scattered to them by master. The time complexity is optimised to a constant. Master gathers a list of dictionaries {block: count} from all processes and have to transform the result into a unified result which process is usually called reduce in most cluster computing systems. The reduce algorithm is simply traverse through the result lists and add up the number of counts under each same block.
[{block: count1}, {block: count2}] -> [{block: count1+count2}]

Result

A processing result is showed in figure 1.

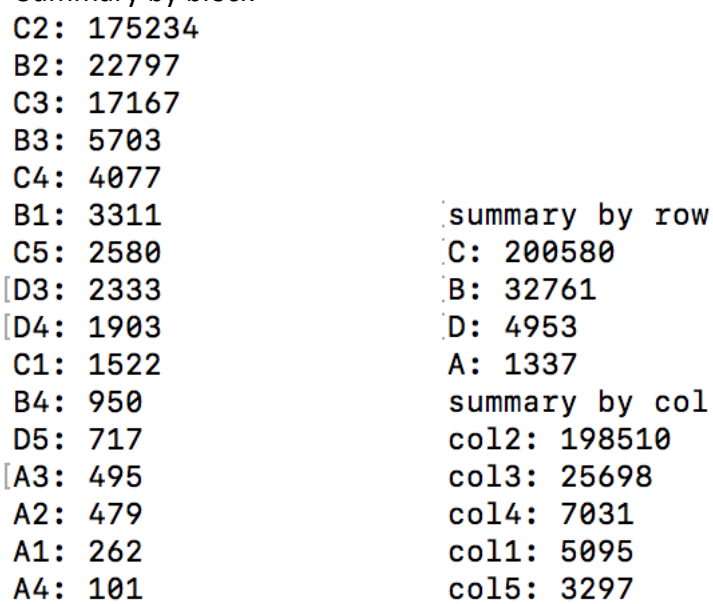

Figure 1

A compare of performance on 1 node 1 core, 1 node 8 core, 2 nodes 8 cores (4 cores per node) based on processing duration is show in Figure 2.

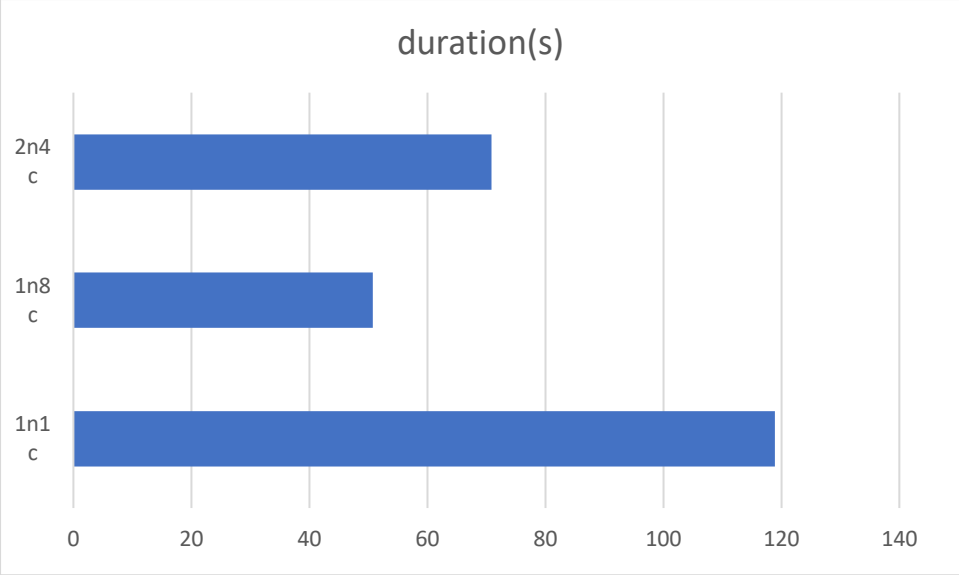

Figure 2

Slurm scripts

```
#!/bin/bash
#SBATCH --time=00:05:00
#SBATCH -p physical
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH -o 1n1c.txt
module load Python/3.4.3-goolf-2015a
mpiexec python cake.py
```

Figure 3

```
#!/bin/bash
#SBATCH --time=00:05:00
#SBATCH -p physical
#SBATCH --nodes=1
#SBATCH --ntasks=8
#SBATCH --cpus-per-task=1
#SBATCH -o 1n8c.txt
module load Python/3.4.3-goolf-2015a
mpiexec python cake.py
```

Figure 4

```
#!/bin/bash
#SBATCH --time=00:05:00
#SBATCH --partition=physical
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=4
#SBATCH -o 2n8c.txt
module load Python/3.4.3-goolf-2015a
mpiexec python cake.py
```

Figure 5

```
Figure 3: 1 node 1 core
Figure 4: 1 node 8 cores
Figure 5: 2 node 8 cores
          with 4 cores per node
```

Analysis

Figure 2 shows the performance of program on different cluster computation resources. An obvious problem is the performance on 2 nodes 8 core situation is lower than 1 node 8 core. The reason is, message passing between two nodes takes more time than between two cores on the same node. In implementation, data is linearly read and scattered to other processes instead of scatter in 8 fixing chunks and scattered at once. Therefore, the performance is worse on 2 nodes situation.

Reference

[1] projected coordinate system used for rendering maps in Google Maps, OpenStreetMap, https://epsg.io/3857.

[2] University of Melbourne (2017) Spartan HPC-Cloud Hybrid: Delivering Performance and Flexibility. https://doi.org/10.4225/49/58ead90dceaaa.