Cluster and Cloud Computing Assignment 1

HPC Twitter GeoProcessing

Shaochuan Luo Hong Zhang

Introduction

Microblogging provides a continuously-updated stream of real-time data which are valuable for public status monitoring and analysis, group behaviors predictions and geographic phenomena modeling. Incorporating twitter content with particular geolocation from which it originated has become a powerful source for modeling population activities that could bridge the gaps between social media contents and real world activities. This area has garnered immense interest in recent years and the speed and performance of twitter data geoprocessing and analysis is crucial in real-time decision-making with large-scale, wide-spread of topics involved.

In this project, the preprocessed large Twitter dataset is utilized to identify Twitter activities around Melbourne and the most frequently used hashtags in each of the grid cell. The geotagged content with the specific locations are the cardinal elements for this experiment. Two main tasks should be accomplished:

Order the grid boxes based on the total number of Tweets posted in each box and return the total count of posts in each box.

Rank the top 5 hashtags (not only 5 hashtag) in each Grid cells based on the number of occurrences of those hashtags.

Facility and tool used in this project are Spartan HPC facility and Python Mpi4py library. Datasets used are bigTwitter.json which include 25,0000 json-format tweets records and melbGrid.json dataset including the grid location information.

The application is given different number of nodes and cores to be utilized and performance to be compared among 3 resource conditions: 1 node 1 core, 1 node 8 cores and 2 node 4 cores. Our goal of this experience is to learn about how basic benchmarking of applications on an HPC facility can be achieved and how a shared resource could have influence and implication on the implementation. Next part is hypotheses. Then we will describe the method and strategy we apply, demonstrate our result. Finally, make a conclusion and evaluation.

Hypotheses

Prior to implementation, basic hypotheses are proposed in this part based on Amdahl's Law. By increasing the number of processors, we hope to get more work done in less time. We speculate the speedup is up to around 5 to 7 from 1 node 1 core to a 1 node 8 cores or 2 node 4 cores system. Another hypothesis is that 1 node 8 cores will be slightly faster than 2 node 4 cores system.

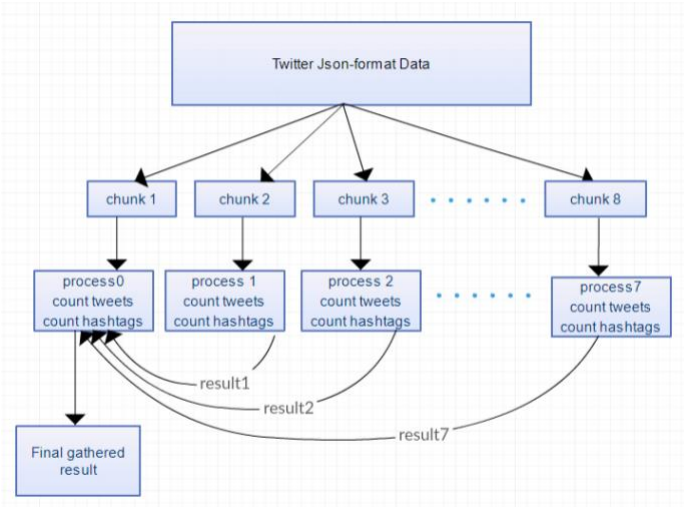
Describe method

```
#!/bin/bash
#SBATCH -J assignment_1_1x8
#SBATCH -p physical
#SBATCH -t 0-00:10:00
#SBATCH -N 1
#SBATCH --account=COMP90024
#SBATCH --ntasks=8
#SBATCH --ntasks-per-node=8
#SBATCH --cpus-per-task=1
#SBATCH --output=1_8.txt

module load Python/3.6.4-intel-2017.u2

time mpirun python3 Assignment1_beta1.py bigTwitter.json melbGrid.json
```

Above screenshot is our slurm file for submitting the job of 1 node 8 cores to SPARTAN. We need 8 tasks (processes) and allocate each task one core. All of task run in one node, so "-N" is equal to one. For 2 node 8 cores experiment, we assign the "-N" with 2 and change "--ntasks-per-node" to 4. For 1 node 1 core, we just need one task. Wall time we set is 10 minutes according to the time (less than 1 second under 1 core) we spend on testing small file. The module we load is python3.6.4 and the blue "time" in bottom is to record the time consumed in this job.

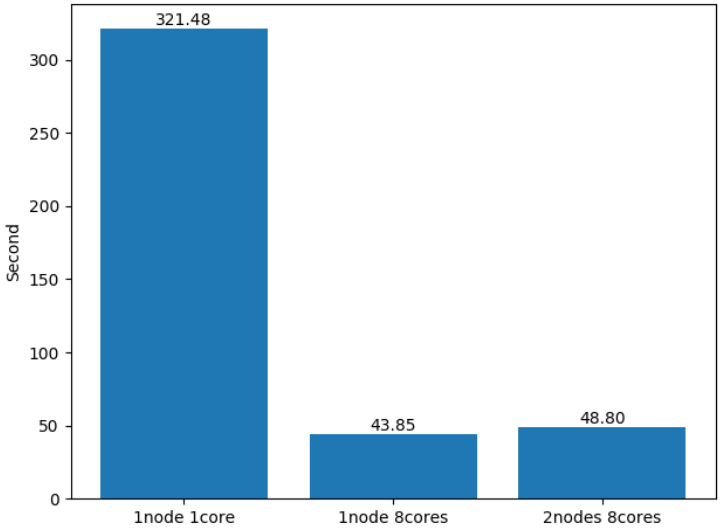
According requirements, we did three experiments under different resources configuration. Resource of the first experiment is one process using one core in one node. All work of reading and processing the file are finished by this process.

For the experiment with one node eight cores, we divide the file into 8 chunks with similar bytes and generate 8 processes to read and process different chunks. Each process owns one core and they are in the same node. After each process finishes its job, the master process will gather the result of all processes and handle it (the details of dealing with this in comments of code). For example, calculate the total of each grid. For this strategy, the parallel job is reading and processing file, serial job is properly handling statistics from different processes and getting final result. We describe the idea of this method by the following graph:



The method applied in experiment with two nodes of four cores are same. The difference is that four processes are allocated to another node. Before end, results of these four processes will be collected by master process.

Result



From above bar chart we can see that times taken by 1 node 8 cores and 2 nodes 8cores are almost same. And time spent in 1 node 1 core is about 7 times as much as 1 node 8 cores and 2 nodes 8 cores. The statistical results of the three experiments are the same. The top grid is grid C2. The texts with details and the scripts used for submitting the job to SPARTAN are included in the zip file.

Conclusion and Evaluation

Since this project is a fixed-size problem, we apply Amdahl's Law to analyze our result. The speed up between 1 node 1 core and 1 node 8 cores is 7.3304, which is a little bit faster than 2 node 8 cores (6.587). The ideal speed up is 8, and both of them are very close to it. That means the serial part of our code accounts for about 10% to 15% of the whole code. That is accessible and consistent with Amdahl's Law.

In some ways, under same number of cores, job executed by single node is faster the multiple nodes. Because cores in a same node share the same address space and there is communication overhead between different nodes. On the other hand, probably, due to SPMD (single program, multiple data, a subcategory of MIMD) technique employed to achieve parallelism, processes can run on multiple CPUs with different input in order to obtain data faster. In the result, time of 2 nodes is little bit slower than 1 node, this is also acceptable considering above causes the complex operating environment.