Cluster and Cloud Computing Assignment 1 Report: HPC Instagram Geoprocessing

SHAJID MOHAMMAD

1. Introduction

The aim of this report is to implement and analyze the performance of a parallelized application which will search a large geocoded Instagram dataset to identify Instagram usage (aka posts) hotspots around Melbourne leveraging the University of Melbourne HPC facility SPARTAN.

2. Dataset

There are two input datasets used in the current parallel application, the first dataset is a large geocoded Instagram dataset which contains a list of posts in the form of JSON which is used to identify Instagram usage hotspots around Melbourne, the file name is bigInstagram.json and its present at the location "/data/projects/COMP90024/", we can access this file by creating a soft link from our home directory using the following command,

ln-s/data/projects/COMP90024/bigInstagram.json

The second file is a JSON-based Grid file called melbGrid.json, which has the reference coordinates i.e. latitudes and longitudes of Melbourne where each of the corners of the boxes is given in the file dividing the geography of Melbourne into small polygons, here rows are denoted by alphabets A, B, C, D and columns are denoted by numbers 1, 2, 3, 4, 5. The melbGrid.json file is present at the location "/data/projects/COMP90024/", we can access this file by creating a soft link from our home directory using the following command,

ln -s /data/projects/COMP90024/melbGrid.json

Hence, we use the two datasets here i.e. the bigInstagram.json and melbGrid.json and identify Instagram activity around Melbourne.

3. Environment

The application runs on, The University of Melbourne's HPC facility SPARTAN. The job scheduler used in Spartan is Slurm which is an open source Workload Manager. To schedule any application on Spartan to run using its HPC resources, we write Slurm scripts which will be discussed later.

The following libraries are required to be loaded in Spartan for our application to execute:

3.1 Slurm:

Slurm is pre-installed and configured in Spartan and it is imported by default, we use the SBATCH directive to invoke it.

3.2 Python:

The application is developed in Python; hence it requires python for execution, Python is pre-configured in Spartan, but we have to import it using the load module command,

module load Python/3.5.2-goolf-2015a

4. Methodology and Evaluation

The Application is written in a python script, HPCInstagramGeoProcessingUsingMPI.py,
The Processing of the Instagram dataset to formulate and identify Instagram usage is run on three different resources in Spartan

- 1 node and 1 core
- 1 node and 8 cores
- 2 nodes and 8 cores (with 4 cores per node).

4.1 1 Node and 1 Core

The Application to identify Instagram usage (aka posts) hotspots around Melbourne is run over then HPC system Spartan with 1 node and 1 core, this is achieved using a single thread. The data in the JSON file is processed one after the other sequentially by the single master thread, here we don't leverage Spartans multi cores but rely on a single core and node for the processing. On Spartan, To Run a job, we schedule it by preparing a Slurm script and submit that job to the Job Scheduler Slurm. We submit it by executing the command,

sbatch 1n1cpy.slurm

Here, 1n1cpy.slurm is the slurm script name and it is scripted as follows,

- Script used to submit job on Spartan:

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --time=0-0:10:00
#SBATCH --partition=physical

# Load required modules
module load Python/3.5.2-goolf-2015a

# Launch multiple process python code
echo "Cluster and Cloud Computing Assignment1
using 1 node and 1 core"

time mpiexec python3
HPCInstagramGeoProcessingUsingMPI.py
melbGrid.json bigInstagram.json
```

- **Performance:**
  When running the Application on a single node and single core, the Time Taken for the Application to process the entire geocoded Instagram dataset is

Resource	Time (Secs)
1 node and 1 core	96

Table 1: Time Taken to process the Geocoded Instagram Dataset on 1 node and 1 core.

4.2 1 node and 8 cores

The Application to identify Instagram usage (aka posts) hotspots around Melbourne is run over the HPC system Spartan with 1 node and 8 cores, this is achieved by spawning 8 processes, where each process runs on a separate core. The data in the JSON file is distributed (Mapper) by the master process (rank = 0), to the other processes, generally from rank 1 to rank 7, based on the value of (linecount % 8), and each core on the slaves is utilized in processing the JSON, and calculating whether it falls in the Melbourne grid and if yes in which cell, after formulating it, it returns the result back to the master where the individual results from the slaves are (Reducer) added and the total count of all the number of posts in a cell is returned, this data is further formulated to get the total number of posts in each row as well by adding the individual count of cells in that row and also to get the total number of posts in each column by adding the individual count of all the cells in that column.

On Spartan, To Run a job, we schedule it by preparing a Slurm script and submit that job to the Job Scheduler Slurm. We submit it by executing the command,

sbatch 1n8cpy.slurm

Here, 1n8cpy.slurm is the slurm script name and it is scripted as follows,

- Script used to submit job on Spartan:

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks=8
#SBATCH --time=0-0:10:00
#SBATCH --partition=physical

# Load required modules
module load Python/3.5.2-goolf-2015a

# Launch multiple process python code
echo "Cluster and Cloud Computing Assignment1
using 1 node and 8 cores"

time mpiexec python3
HPCInstagramGeoProcessingUsingMPI.py
melbGrid.json bigInstagram.json
```

- **Performance:**
  When running the Application on a single node and 8 cores, the Time Taken for the Application to process the entire geocoded Instagram dataset is:

Resource	Time (Secs)
1 node and 8 cores	50

Table 2: Time Taken to process the Geocoded Instagram Dataset on 1 node and 8 cores.

4.3 2 nodes and 8 cores

The Application is run over the HPC system Spartan with 2 nodes and 8 cores, this is achieved by spawning 8 processes, 4 on each node, where each process runs on a separate core. The data in the JSON file is distributed (Mapper) by the master process (rank = 0), to the other processes, generally from rank 1 to rank 7, based on the value of (linecount % 8), and each core on the slaves is utilized in processing the JSON, and calculating whether it falls in the Melbourne grid and if yes in which cell, after formulating it, it returns the result back to the master (Reducer) where the individual results from the slaves are added and the total count of all the number of posts in a cell is returned, this data is further formulated to get the total number of posts in each row as well by adding the individual count of cells in that row and also to get the total number of posts in each column by adding the individual count of all the cells in that column.

On Spartan, To Run a job, we schedule it by preparing a Slurm script and submit that job to the Job Scheduler Slurm. We submit it by executing the command,

sbatch 2n8cpy.slurm

Here, 2n8cpy.slurm is the slurm script name and it is scripted as follows,

- Script used to submit job on Spartan:

```
#!/bin/bash
#SBATCH --nodes=2
#SBATCH --ntasks=4
#SBATCH --time=0-0:10:00
#SBATCH --partition=physical

# Load required modules
module load Python/3.5.2-goolf-2015a

# Launch multiple process python code
echo "Cluster and Cloud Computing Assignment1
using 2 nodes and 8 cores"
time mpiexec python3
HPCInstagramGeoProcessingUsingMPI.py
melbGrid.json bigInstagram.json
```

- **Performance:**
  When running the Application on 2 nodes and 8 cores (4 on each), the Time Taken for the Application to process the entire geocoded Instagram dataset is:

Resource	Time (Secs)
2 nodes and 8 cores (4 each)	56

Table 2: Time Taken to process the Geocoded Instagram Dataset on 2 nodes and 8 cores.

5. Result:

We can check the status of our jobs in our queue by executing the command,

squeue -u shajid

If all the jobs are completed our queue will be empty and we can file's likes slurm-<jobid>.out.
We can check the output of our jobs by opening the file, for the current project the file created is *nano slurm-3384990.out* for the job which used 1 node and 8 cores and the result of the job is as follows:

Cluster and Cloud Computing Assignment1 using 1 node and 8 cores

The Ordered (ranked) Total number of Instagram posts made in each box is :
C2 : 175838 posts,
B2 : 22062 posts,
C3 : 17184 posts,
B3 : 5815 posts,
C4 : 4145 posts,
B1 : 3311 posts,
C5 : 2613 posts,
D3 : 2269 posts,
D4 : 1889 posts,
C1 : 1522 posts,
B4 : 900 posts,
D5 : 717 posts,
A2 : 479 posts,
A3 : 363 posts,
A1 : 262 posts,
A4 : 85 posts,

The Ordered (ranked) Total number of Instagram posts made in each row (row based) is :
C : 201302 posts,
B : 32088 posts,
D : 4875 posts,
A : 1189 posts,

The Ordered (ranked) Total number of Instagram posts made in each column (column based) is :
2 : 198379 posts,
3 : 25631 posts,
4 : 7019 posts,
1 : 5095 posts,
5 : 3330 posts,

real 0m50.290s
user 5m33.592s
sys 0m10.268s

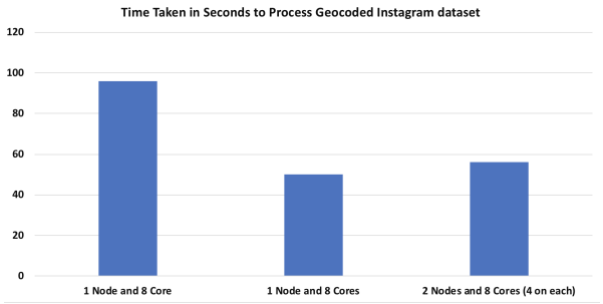
6. Evaluation:

The following table can be formulated, comparing the performance times on different resource configurations,

Resources Used on Spartan	Time Taken (Sec)
1Node and 1 Core	96
1Node and 8 Cores	50
2Nodes and 8Cores (4 on each)	56

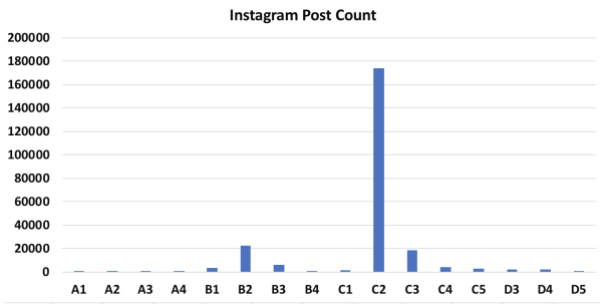
Table 5: Comparing the Results of the Performance in Time Taken per seconds to Execute.

It is observed that the performance on running the application on Spartan with 1 node and 8 cores is the highest with least time taken to execute, followed by the run which used 2 nodes and 8 cores and finally then the run which used just, 1 node and 1 core.
A Bar Graph can be plotted between the execution times for these runs as shown below,



Graph1: Comparing the Execution Times for the Application which ran on 3 different Configurations.

A Bar Graph can be plotted between the Total number of Instagram Posts in each cell, as shown below,



Graph2: Comparing the Total number of Instagram Posts in each cell.

7. Conclusion

This report established and presented the performance of a parallelized application which searches a large geocoded Instagram dataset to identify Instagram usage hotspots around Melbourne over the University of Melbourne's HPC facility SPARTAN, on three different resources i.e. with 1 node and 1 core, with 1 node and 8 cores, and 2 nodes and 8 cores and it is observed that the performance on running the application on 1 node and 8 cores is the highest and fastest followed by 2 nodes and 8 cores and then 1 node and 1 core as shown in Graph 1.