Assignment 1 Report

Vedant Chauhan
892758

Introduction

The assignment identifies Instagram activity around Melbourne. Assignment is programmed in Java. The two files used in the program are melbGrid.json and bigInstagram.json.

~ melbGrid.json :- JSON grid file.
~ bigInstagram.json :- main 10GB JSON file for analysis.

Using the melbGrid file, grid of Melbourne is created and coordinates from bigInstagram file are matched with the grid. If coordinates does not exist, then ignore the post. Otherwise, check if the coordinates lie inside the grid and increase the count of the particular grid. Print the grid with a particular order.

How to execute

As Spartan HPC facility has jar zip issue, following commands are used

$ module load Java/1.8.0_71

$ jar –xf HPC_GeoProcessing.jar

$ jar –xf commons-io-2.6.jar

$ jar –xf commons-io-2.6-javadoc.jar

$ jar –xf gson-2.8.2.jar

$ sbatch script.slurm

Program description and parallelization

The java library used in program are gson-2.8.2.jar for JSON parsing and commons-io-2.6 for stream handling. Due to problems encountered in Spartan, command line arguments are not used. The two files are hardcoded, and can be change in the program. ‘JsonReader’ parses the JSON and makes the grid, and commons-io ‘lineIterator’ iterates the Instagram posts and increases the count of grid if post lies in the grid.
Program is parallelized using mpi with rank 0 consider as Master and other as slaves. Whole JSON files are read by slaves and master (master acting as slave), they return the count of grid items to master. Master computes the order and print the results. I understand this is a poor technique to read the whole file by each slave instead of splitting the file between processes. It leads to performance and efficiency issues.

Scripts and Results

Nodes and Cores	1 node and 1 core	1 node and 8 core	2 node and 8 core
Script	#!/bin/sh #SBATCH --nodes=1 #SBATCH --ntasks=1 #SBATCH --time=01:00:00 #SBATCH --partition=physical module load Java/1.8.0_71 module load mpj/0.44 mpjrun.sh -np 1 HPC_GeoProcessing	#!/bin/sh #SBATCH --nodes=1 #SBATCH --ntasks=8 #SBATCH --time=01:00:00 #SBATCH --partition=physical module load Java/1.8.0_71 module load mpj/0.44 mpjrun.sh -np 8 HPC_GeoProcessing	#!/bin/sh #SBATCH --nodes=2 #SBATCH --ntasks-per-node=4 #SBATCH --cpus-per-task=1 #SBATCH --time=01:00:00 #SBATCH --partition=physical module load Java/1.8.0_71 module load mpj/0.44 mpjrun.sh -np 8 HPC_GeoProcessing
Results (Physical Partition)	MPJ Express (0.44) is started in the multicore configuration Hostname: spartan-bm014.hpc.unimelb.edu.au -----Grid Ordering----- C2: 175362 B2: 22817 C3: 17180 B3: 5708 C4: 4081 B1: 3312 C5: 2615 D3: 2336 D4: 1906 C1: 1596 B4: 951 D5: 717 A3: 497 A2: 479 A1: 262 A4: 103 -----Row Ordering----- C-Row: 200834 B-Row: 32788 D-Row: 4959 A-Row: 1341	MPJ Express (0.44) is started in the multicore configuration Hostname: spartan-bm003.hpc.unimelb.edu.au Hostname: spartan-bm003.hpc.unimelb.edu.au Hostname: spartan-bm003.hpc.unimelb.edu.au Hostname: spartan-bm003.hpc.unimelb.edu.au Hostname: spartan-bm003.hpc.unimelb.edu.au Hostname: spartan-bm003.hpc.unimelb.edu.au Hostname: spartan-bm003.hpc.unimelb.edu.au Hostname: spartan-bm003.hpc.unimelb.edu.au -----Grid Ordering----- // Same as left side -----Row Ordering----- // Same as left side -----Column Ordering----- // Same as left side Time usage = 39.0 s	MPJ Express (0.44) is started in the multicore configuration Hostname: spartan-bm019.hpc.unimelb.edu.au Hostname: spartan-bm019.hpc.unimelb.edu.au Hostname: spartan-bm019.hpc.unimelb.edu.au Hostname: spartan-bm019.hpc.unimelb.edu.au Hostname: spartan-bm019.hpc.unimelb.edu.au Hostname: spartan-bm019.hpc.unimelb.edu.au Hostname: spartan-bm019.hpc.unimelb.edu.au Hostname: spartan-bm019.hpc.unimelb.edu.au -----Grid Ordering----- // Same as left side -----Row Ordering----- // Same as left side -----Column Ordering----- // Same as left side Time usage = 63.0 s

	-----Column Ordering----- Column 2: 198658 Column 3: 25721 Column 4: 7041 Column 1: 5170 Column 5: 3332 Time usage = 32.0 s		

Table 1. Scripts and results of program

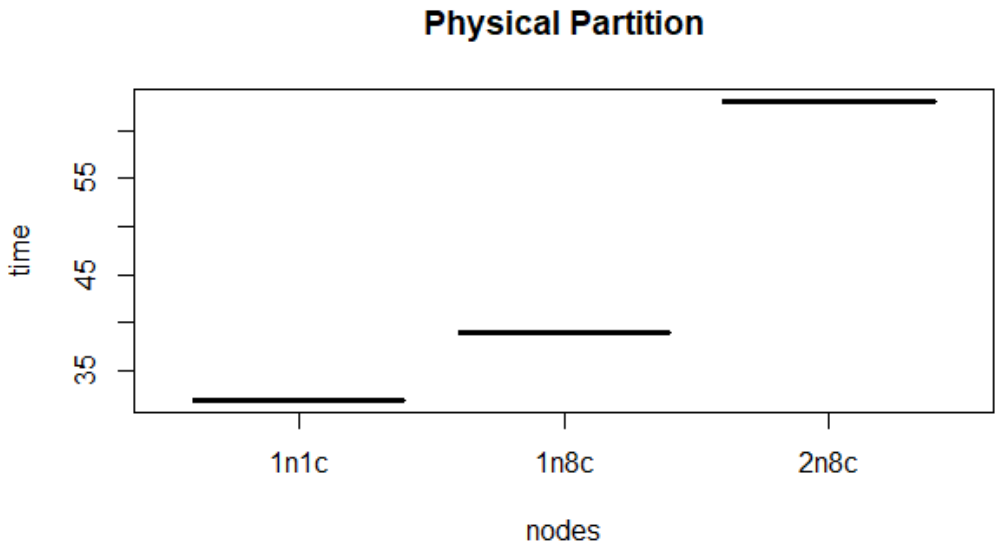

Fig. 1. Plot of time (in seconds) vs nodes based on partition

Analysis

1. After allocating multiple nodes in script, Spartan scheduler allocates only one node for all processes. But, after checking further cluster configuration of mpj is not working properly in Spartan. Another reason can be, first Spartan scheduler allocates 1 node and 4 process to one node (spartan-bm019) and then checks for free node and happens to find the same node (spartan-bm019).
2. I have executed the program once (in physical partition). There can be issue with consistency. However, results are unexpected. In physical partition, as shown in Fig.1, time got increased which is not ideal. It can be due to the nature of implementation. In this case, slave and master are reading the JSON files, leading the processes to take time. As, same node is allocated by Spartan. When a process finishes a task, another task will be given to the same process. Therefore, time taken by slaves to read the file gets increased and overall performance gets affected. Needless to say, a poor approach to handle parallel tasks.