Cluster and Cloud Computing Assignment 1
HPC Instagram GeoProcessing

Ruifengl

686141

Objective:

The assignment is to identify Instagram posts around Melbourne from large Instagram dataset. We implement parallelized application on high performance computer, Spartan, to identify those posts. Based on different resources (nodes and cores), I have made analysis through approach and variation in performance below.

How to invoke application using script

To invoke the application, I set up resources using Slurm options. For each different setting, 1 node 1 core, 1 node 8 cores and 2 nodes 8 cores, I run a single Slurm job three separate times. The script has been given below.

```
1  #!/bin/bash
2  #SBATCH -p physical
3  #SABTCH --time=0-12:00:00
4  #SBATCH --nodes=1
5  #SBATCH --ntasks-per-node=1
6  #SBATCH --output=bigInstagram_1n1c.out
7
8  module load Python/3.4.3-goolf-2015a
9  time mpiexec python asgn_code_2.py
10
```

Figure 1. Bash script

In HPC cluster system, node is an individual computing system. Each node is equipped with CPUs, RAMs and hard disks. Message passing interface(MPI) is a system, which helps to exchange data between different nodes through communication network. Each node works on portion of computing problem so that it is to perform distributed memory parallel programming.

In the following script**,** #!/bin/bash used to invoke bash. Using #SBATCH -p physical**,** it specifies physical partition for storage space allocation. Physical partition equipped with 21 nodes, each with 12 cores and 251 GB of RAM, and high-speed 56GigE networking connected with nodes. Comparing to cloud partition, it has better performance of communications between nodes. Then it specifies the number of nodes required (#SBATCH –-nodes= #), cores required for each node (#SBATCH –ntasks-per-node=#). #SBATCH –time= 0-12:00:00 restricted wall time up to 12 hours, which provides maximum amount time that application can run. #SBATCH --output=bigInstagram_#n#c.out gives the results after it's done the job.

Here I programmed with python3 loading module Python/3.4.3-goolf-2015a. It runs application using mpiexec command, along with execution time using time command.

Approach used to parallelize your code

I used python 3 as the development tool, along with mpi4py library. Among those methods of mpi4py, I choose collective communication. The main idea of collective communication is sending different chunks of dataset to different processes.

Firstly, I utilize regular expression to read all coordinates then store them in a new array. The root process takes the new array, then distribute chunks of array to each different rank using MPI_Scatter. In this case, I used Numpy library to split array to even sized chunks. At the end, root process gathers and combines all results from ranks using MPI_Gather. Each rank takes different data to run in parallel. By doing so, it improves the efficiency using parallel computing. The concepts of MPI_Scatter and MPI_Gather pictures have been given below.

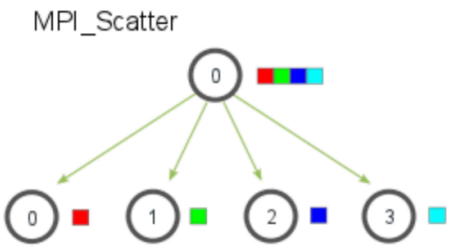

Figure 2. MPI_Scatter from mpi4py

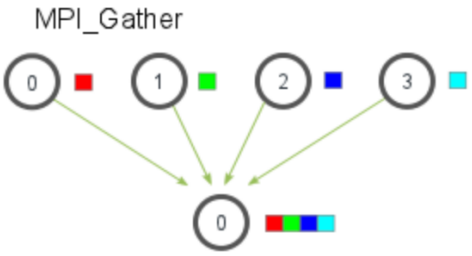

Figure 3. MPI_Gather from mpi4py

Variations in its performance

As can be seen from 3 different settings below, it is slower running on single node and single core comparing to single node and octal cores. This is because we take advantage of running parallel code in the multiple processors. More specific, from the table below, running with 8 cores is 10% faster than running with single core.

Within one node, multiple processors are connected to a single main memory and are controlled by a single operating system. Comparing to running job on two nodes, single node allows faster processor memory accessing. And also, communication between two nodes take network overhead and latency of Ethernet network into account. Multiple nodes utilize synchronization and atomic operations, which lead to low-level performance and communication latency. As we can see from the table below, running job with two nodes is about 3 second slower than that with single node and octal cores.