**Cluster and Cloud Computing**

**Bing Xie 741012**

# 1 Introduction

This project is implemented in two method to develop parallelized python based application. One method is that master loads json data and then scatter those data to slaves, each slave processes data and send processed data to master, master gathers processed data. Compare with the first method, the difference of the second method is that master does not have to scatter data, each slave picks up a block of data to process and send processed data to master, before gathering processed data, master play the same role as slaves. These two methods apply mpi library and map-reduce frame similar to the mechanism of Hadoop, Spark and Storm.

# 2 System Structure (Map-Reduce Mechanism)

MPI is used to manage the messages transfer between cores. Rank 0 are set to be master core, the others are slave cores.

## 2.1 Scatter-Gather Method

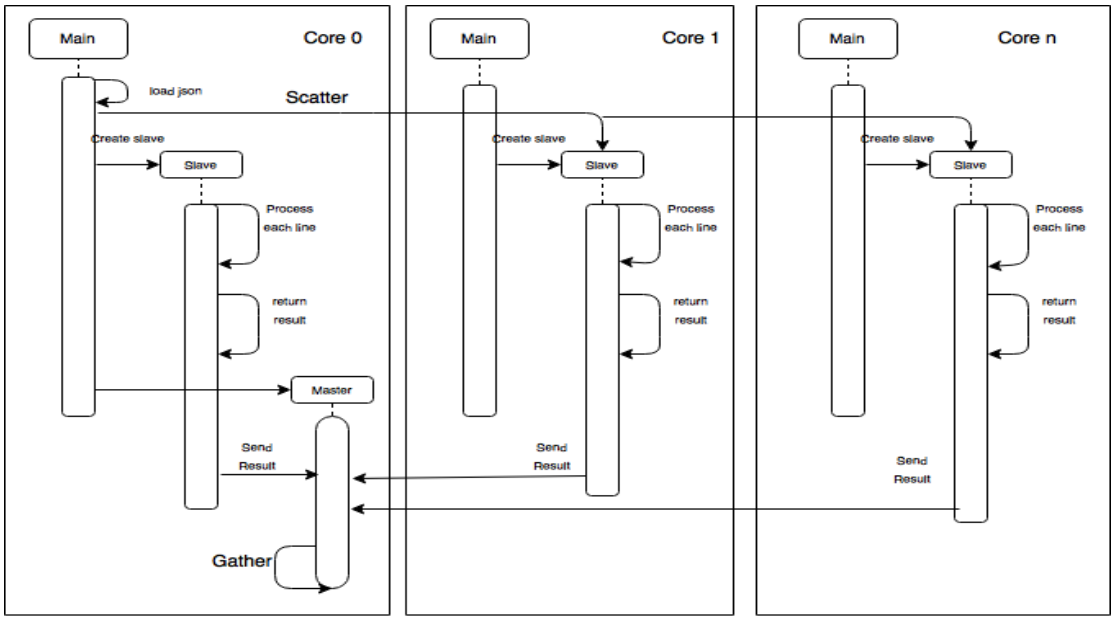


Figure 1. Processes between cores (Scatter-Gather method)

Figure 1 illustrates the workflow of first method. Master is responsible for loading json data line by line and dispatch data to slaves. The reason that loading json data line by line is to avoid memory overflow. Then split the data in the size of cores and **scatter** those data to slaves by using **data=comm.scatter(data, root =0).**

In order to make program efficiency proper, I process the meshGrid.json and store the information in a dictionary, which will be used by slaves to process data. After that, each slave will send result **data_send** to master.

For core of master, it does the job as a slave after data is scattered. Then it uses **comm.gather(data_send,root=0)** receive the messages from other cores. When all data is gathered at master in form of a list of list, master will reduce all data, then print the result as in outputs file.

### 2.2 Gather Method

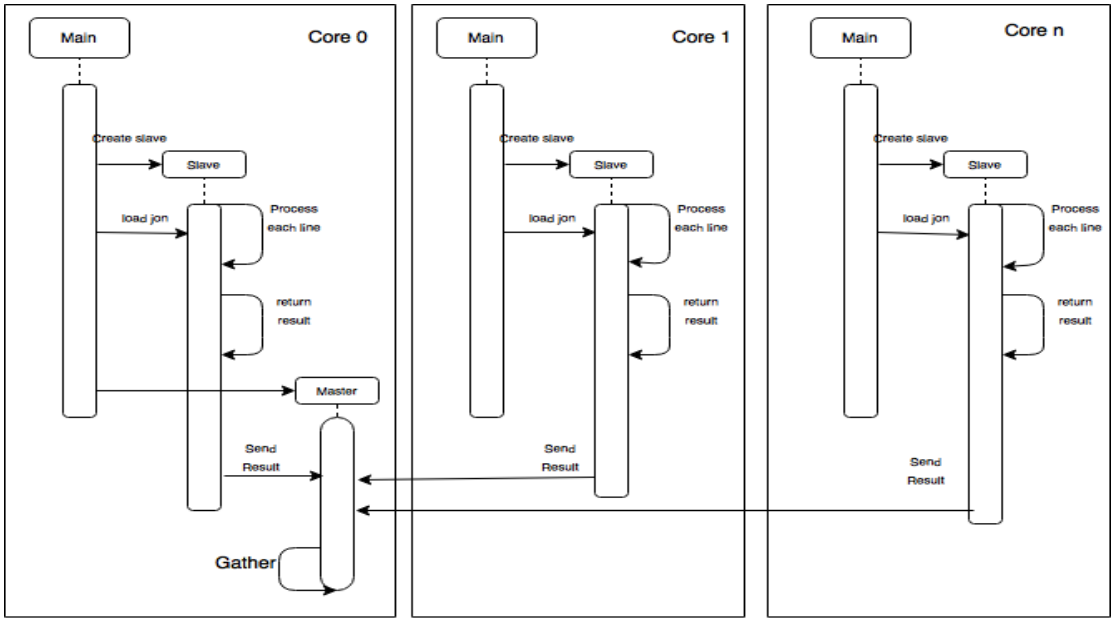


Figure 2. Processes between cores (Gather method)

Figure 2 illustrate the workflow of second method. For a core of slaves it process the lines **lineNumber%CoreSize==Rank**, which means master has not to dispatch data and slaves can processes the data refer to lineNumber and coreSize. The other procedure is same as Scatter-Gather method.

## 3 Settings in shell executables files

| Job_Script1n1c.sh | Job_Script1n8c.sh | Job_Script2n8c.sh |
|---|---|---|
| #!/bin/bash | | |
| #SBATCH -p physical | | |
| #SBATCH --time=01:00:00 | | |
| #SBATCH --nodes=1 | #SBATCH --nodes=1 | #SBATCH --nodes=2 |
| #SBATCH --ntasks=1 | #SBATCH –-ntasks-per-node=8 | #SBATCH –-ntasks-per-node=4 |
| #SBATCH –cpus-per-tasks=1 | #SBATCH –cpus-per-tasks=1 | #SBATCH –cpus-per-tasks=2 |

| #module load Python/3.4.3-goolf-2015a | | |
|---|---|---|
| time mpirun –np 1<br>python GeoProcess.py | time mpirun –np 8<br>python GeoProcess.py | time mpirun –np 8<br>python GeoProcess.py |

## 4 Executing Result and Analysis

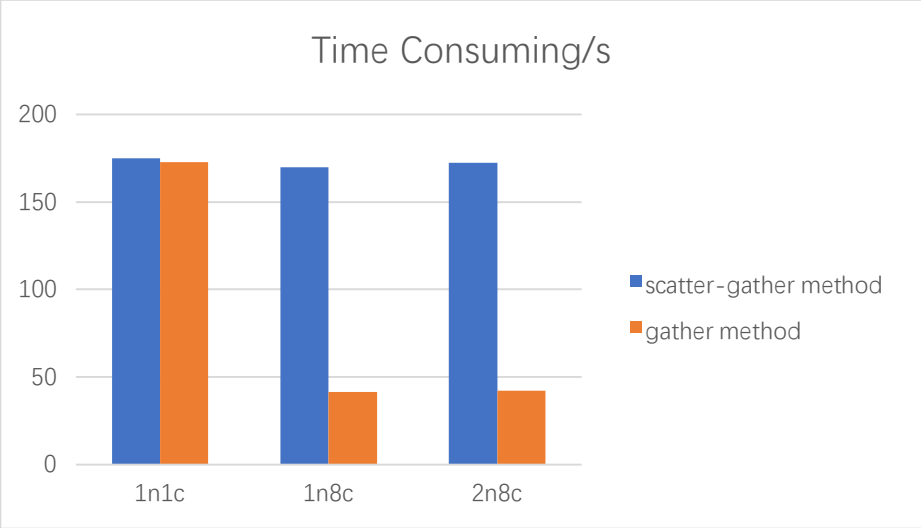


Figure 3 Time Consuming of two methods

Outputs files display the rank of total count of tweets in each box, row and column. Figure 3 displays the comparison of executing time between two methods, it can be concluded that:

1. Loading json data and scatter data from master to slaves consume much time since the consuming time of scatter-gather method is about three times than gather method with 1 node 8cores or 2nodes 8cores. With slaves picking up and loading data, parallelized program could achieve high performance.
2. The consuming time of 2 nodes 8 cores is slightly more than 1 node 8 cores, the possible reason is that it requires time for process communication between nodes.
3. With core and node increase, performance of scatter-gather method is not improved, it may caused by the time consuming of processes communication between cores and nodes.
4. For this project, Gather method has higher performance the efficiency has been improved more than three time when program is parallelized.