Cluster and Cloud Computing

Assignment 1 Report

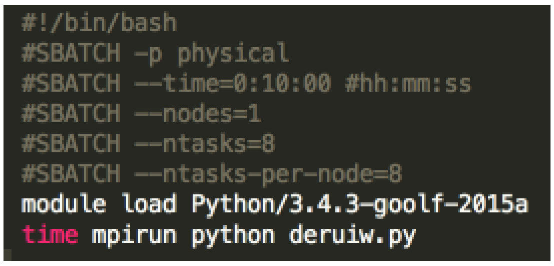
Derui Wang 679552

1. The invoke instructions

My python file responds directly to the scripts files. I have three script files which are 1n1c.sh, 1n8c.sh, 2n8c.sh. They stand for 1 node 1 core, 1 node 8 cores, and 2 nodes 4 cores for each node separately. The way to invoke three different script files is just simply run command: 'sbatch 1n1c.sh' (for example). The system will generate the output file correspondingly.

2. Example scripts:

Other scripts will be attached in the zip file. My script uses Python version 3.4.3 and sets the tolerance time to be 10 minutes. I tried to run the program in physical machine as it is more stable than virtual machine.



```
#!/bin/bash
#SBATCH -p physical
#SBATCH --time=0:10:00 #hh:mm:ss
#SBATCH --nodes=1
#SBATCH --ntasks=8
#SBATCH --ntasks-per-node=8
module load Python/3.4.3-goolf-2015a
time mpirun python deruiw.py
```

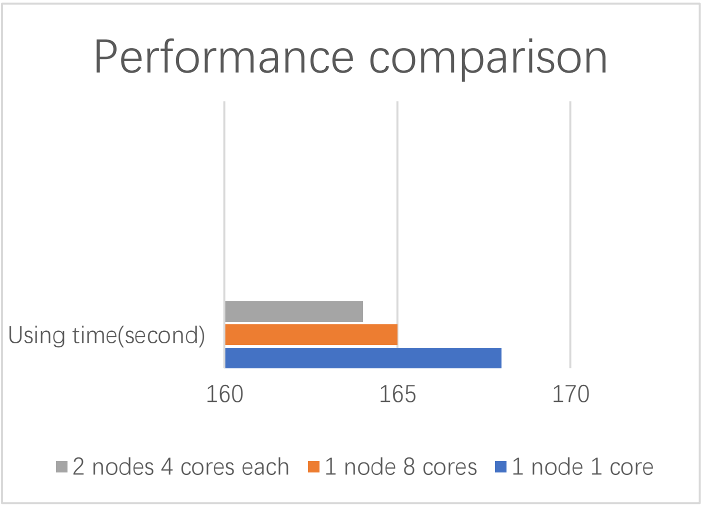
3. The way of parallelization

1) I tried to read the content of the 'bigTwitter.json' line by line in the rank 0 which is the first core. After that I extract the coordinates straight after parsing the JSON data. Then I use a filter function to get rid of the location which locates out of Melbourne. This way saves me about 2 minutes of running the program.

   I use 'numpy' library's function 'array_split' to split the above data into the size of using cores parts. Then I scatter the data to different cores.

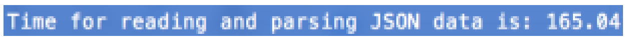
2) For each ranks(cores), I run the function 'count_each_section' separately to count the data in each section and use 'allgather' method to gather data from all ranks. After combing them, I will sort the result dictionary and print out the needed format result.

4. Performance comparison



As we can see from above, there is no much difference between each method.

As for this task, the reading and parsing time occupied the most in the total time usage. I try to time on the reading and parsing and give the following result.



```
Time for reading and parsing JSON data is: 165.04
```

According to the above graph, we can find that the time expense for reading and parsing is extremely large. There is a bit error between my timing program and the system timing method. But we can still find that the actual task (count each sections twitters) do not take much time to process. In this context, I will discuss the performance in next section.

5. Performance discussion

1) Based on the reading files and parsing JSON method I use, they are all displayed

in core 1, which means for different cores and nodes I used, they will all consume the same amount of time. But as for 2 nodes, after core 1 in one node read and parse the data, it needs to communicate with another node. In this case, the real time consuming is still debatable depending on real world cases.

2) By comparing 1 node 1 core and 1 node 8 cores, as shown above, 8 cores run faster than 1 core. This is because for different processors(cores), the whole task is split into 8 parts and run separately. Except from concerning fast scatter and gather process, 8 cores will take less time comparing with 1 core.
3) For the comparison between 1 node 8 cores and 2 nodes 4 cores each, I run it several times. Sometimes 2 nodes run faster and sometimes 1 node run faster. The difference would usually be 1 or 2 seconds. However, depending on networking and inter-process communication, the time consuming may vary. Spartan system sometimes decides to use the same node for both nodes, which could also affect the result.

   Theoretically, 2 nodes and 4 cores each should takes less time than 1 node 8 cores if we do not consider nodes' communication. Because 2 nodes stand for 2 computers which means more Ram. This gives more memory usage for one core to process the program. Nevertheless, for the certain task, it needs precise computation on time consuming as nodes communication could affect the result.

4) To sum up, based on Amdahl's law, multicore system should mean better performance. The speed improves base on the serial time/ parallel time.

6. Future Improvement

As shown above, the reading and parsing JSON process takes the majority time. In this case, the reading part can be run in different cores. For different cores, different rank starts to read the file at rank*(total lines/total ranks).

I use python JSON library to parse the given JSON data. However, the parsing process is quite slow. Comparing with other method such as readlines() and get rid of the useless parts by using regular expression. This could save more time than loads() method.