HW 1: Parallel Programming with MPI

Mian Qin, UIN: 725006574

1. Plot execution time versus p to demonstrate how time varies with the number of processes.

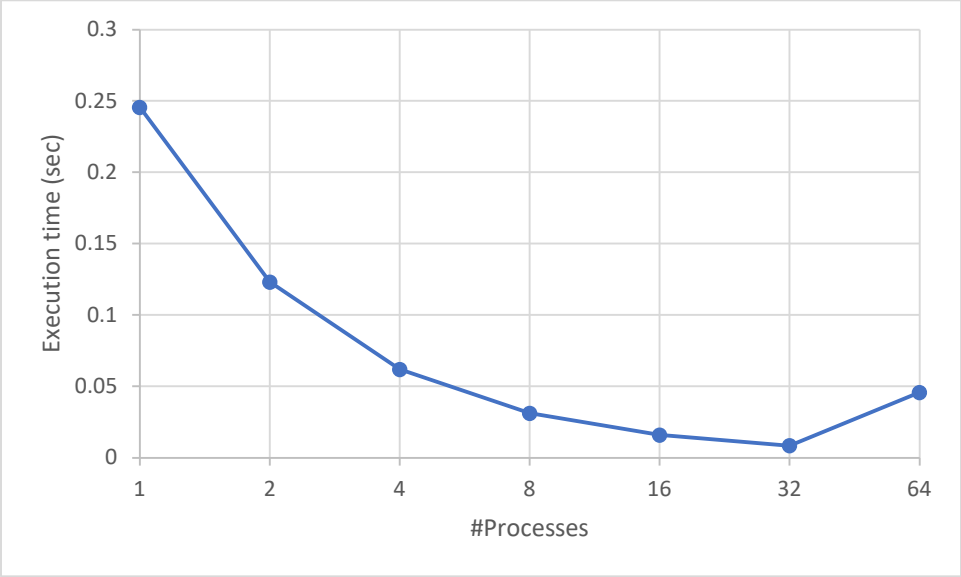

Figure 1 Execution time versus number of processes

The results are collected through averaging three independent running results (variance is very small). From Fig. 1 we can see that the minimum execution time is achieved at 32 processes. 64 processes result is even higher than 8,16,32 processes. I think this is due to communication overhead across nodes, since we have fixed number of processes per node. (64 processes has 16 nodes compared to 8 nodes for 32 processes)

2. Plot speedup versus p to demonstrate the change in speedup with p

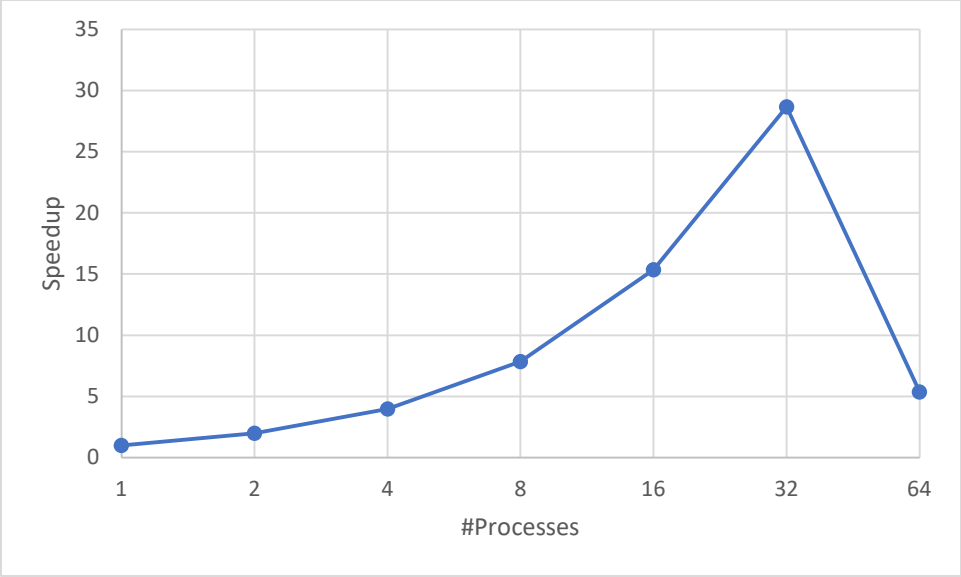

Figure 2 Speedup vs number of processes

From Fig. 2 we can see that before 32 processes, the speedup is almost linear with the increase of number of processes.

3. Plot efficiency versus p to demonstrate how efficiency changes as the number of processes is increased.

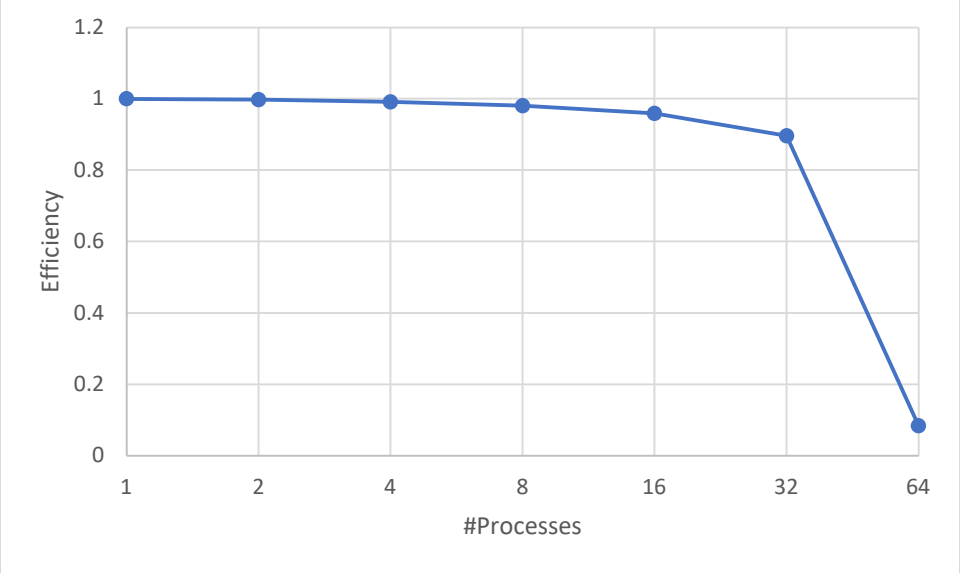

Figure 3 Efficiency versus number of processes

4. What value of p minimizes the parallel runtime?

As discussed under Q1, when p = 32, it minimizes the parallel runtime.

5. Plot time versus ptile to illustrate your experimental results for this question.

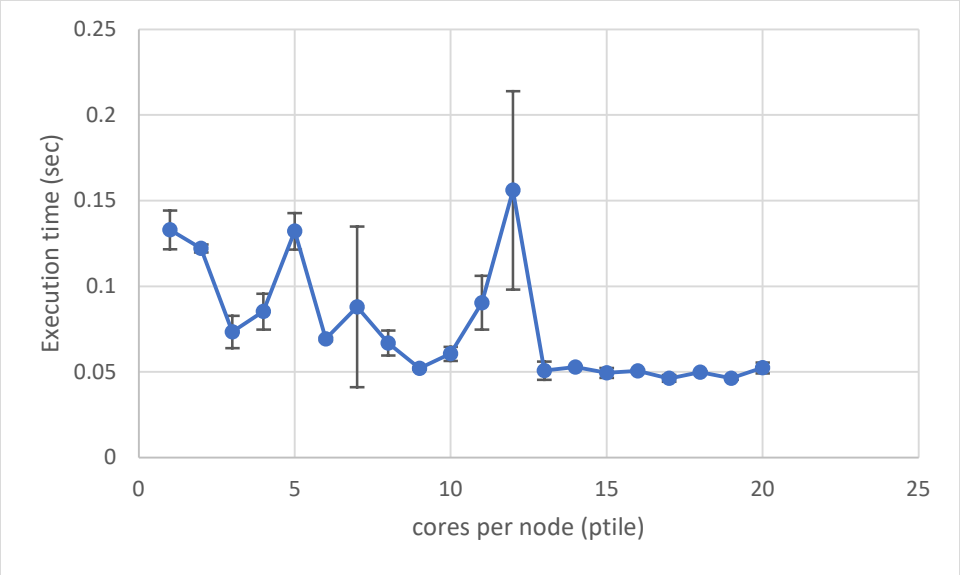

Figure 4 Execution time versus ptile under 64 processes

Before running any experiment, considering the architecture of the ada cluster (the node we are using has maximum of 20 cores per node, hyper-thread not enabled). Since the between nodes communication (network) is the main overhead (within a node, the communication can be down through share memory), I presume larger ptile (maximum of 20) should achieve minimum execution time.

For conducting this experiment, I wrote a script file to automatically change the ***ptile*** parameters (from 1 to 20) in the job file and submit the job file to get the results. Since the maximum number of cores in the node we are using in ada is 20, so we stop at 20 for the ptile scaling. (If the ptile exceeds 20, the batch won't execute). I ran the experiment 5 times to get the results as shown in Fig. 4. The minimum execution time is achieved when ptile=17 (across 13 to 20 the execution time is similar since they use similar amount of nodes overall which is around 4). This highly depends on how the processes are mapped to the physical location of the nodes which will determine the communication overhead. The large variance for some data points is caused by different node mappings for each runs.

6. Repeat the experiments with p=64 for n=10^2, 10^4, 10^6 and 10^8.
    a. Plot the speedup observed w.r.t. p=1 versus n.

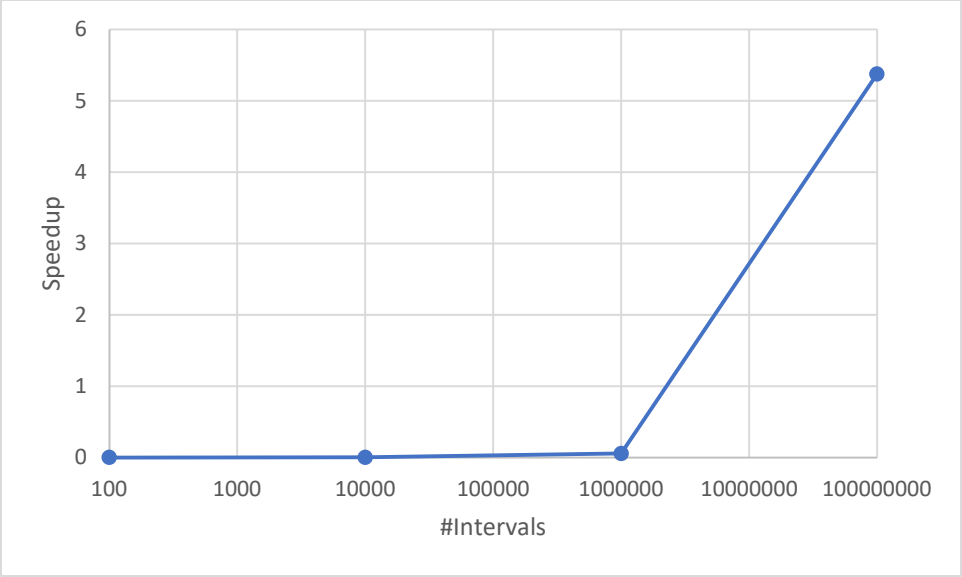

Figure 5 Speedup versus number of intervals under p=64

b. Plot the relative error versus n to illustrate the accuracy of the algorithm as a function of n.

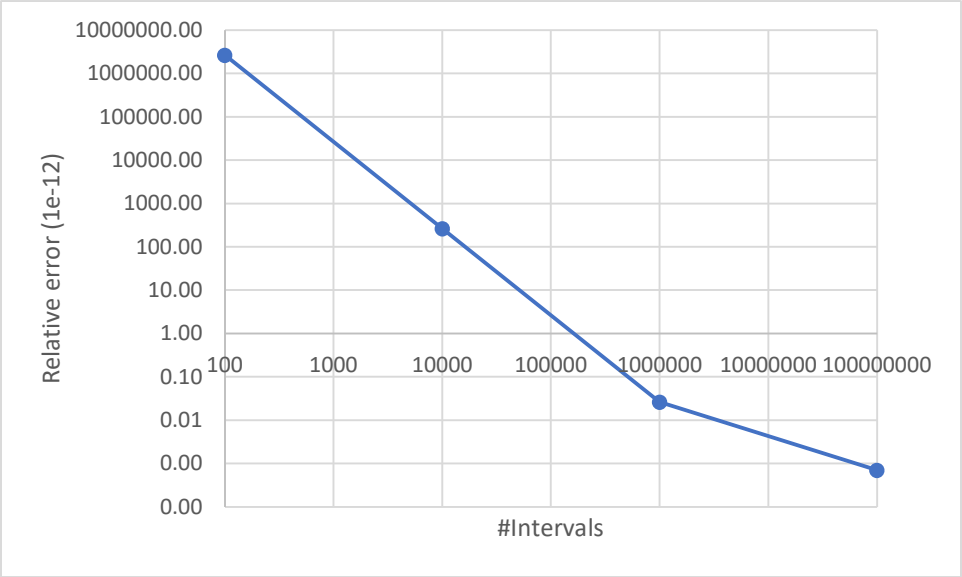

Figure 6 Relative error versus number of intervals under p=64

Noted that both number of intervals and relative error is logarithmic scale. From Fig. 6 we can see that the relative error goes up with the increase of intervals. This is because with the increase of intervals, more computation is conducted and more precise is the results.