Evaluation of performance of MPI program

Yonggang Li (18043158)

**Abstract**

This report evaluates the performance of a MPI program which implements Monte Carlo methods to compute of the value of π. The program is running on a range of available nodes and processors. The testing result demonstrate Amdahl’s law well. The inter-process communication time is measured as it is an important factor in distributed computing.

**Amdahl’s law**

f is the non-parallelized serial fraction of the total work, if p increases bigger and bigger, the speedup goes to . In other words, the limit is set by the serial part of the task and giving more processors won’t help if p is big enough. To achieve higher speed-up the only way is to make the serial part smaller. Figure 2 below clearly shows this.

**Analysis of the program**

In the program, the serial part is to generate the initial random numbers and send to all worker nodes and waiting to receive result from all worker nodes to calculate π. The parallel part is that the worker nodes generate random numbers and test if they fall in the unit circle.

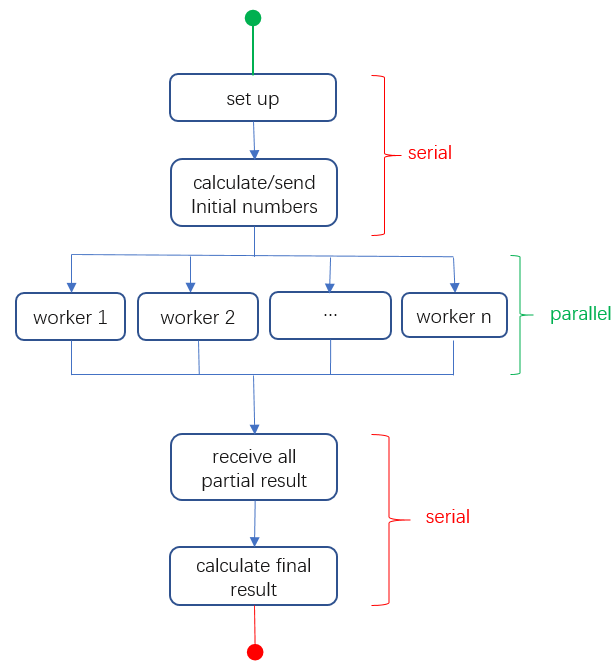


Figure 1. Serial and parallel parts of the program

**Testing result**

Observed speed-ups is the value of total time of one processor divided by total time of N processors.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| processors | total time | time of serial parts | f | speed-ups | |
| theoretical | observed |
| 1 | 0.092889 | 0 | 0 | 1 |  |
| 2 | 0.047515 | 0.00004 | 0.000841839 | 1.99992 | 1.95 |
| 4 | 0.023604 | 0.000211 | 0.008939163 | 3.99747 | 3.94 |
| 8 | 0.011778 | 0.001634 | 0.138733231 | 7.90953 | 7.89 |
| 16 | 0.009624 | 0.004537 | 0.471425603 | 14.98050 | 9.65 |
| 32 | 0.013906 | 0.011364 | 0.817201208 | 23.66367 | 6.68 |

Table 1. Performance testing result

**Inter-process communication time**

As the code below, master sends to and receives from workers one by one, the latter the worker the more time it waits for. Therefore, the time is measured in master process just before sending and after receiving.

*for (i=1; i<numproc; i++)*

*{*

*time\_start = MPI\_Wtime();*

*MPI\_Send(&data\_send, 1, MPI\_LONG, i, 0, MPI\_COMM\_WORLD);*

*MPI\_Recv(&data\_recv, 1, MPI\_LONG, i, 0, MPI\_COMM\_WORLD, &Stat);*

*Time\_elapsed = MPI\_Wtime() – time\_start;*

*fprintf(stdout, "It takes %f to communicate with worker-%d.\n", time\_elapsed, i);*

*}*

Figure 2 shows the communication time between master and workers, the average time is about 0.0005876 seconds. We can see the time increased dramatically from about 0.000030 to 0.00743 in the eighth sending and receiving.

Figure 2. communication time

If the master calculates each random number, sends to worker for testing and receive response, averagely it will take 2350 (0. 0005876 x 4,000,000) seconds for four million times communication solely.

**Conclusion**

In this test, the observed speed-ups are close to the theoretical values when the number of process is small. When the processors are added to 16, the observed speed-up reaches a peak at 9.65 but is just 64.55% of the theoretical value. Adding to 32 processors, the speed-up lower down to 6.59. It takes nearly 0.0006 seconds to communicate between master and worker processors.

**Reference**

[1] Amdahl's Law vs. Gustafson-Barsis' Law [online]. Last accessed August 3, 2018 at <http://www.drdobbs.com/parallel/amdahls-law-vs-gustafson-barsis-law/240162980?pgno=2>