Evaluation of performance of Parallel Bucket Sort

Yonggang Li (18043158)

**Abstract**

This report evaluates the performance of a MPI program which implements parallel bucket sort. The program is running on a range of available nodes and processors. The testing result demonstrate Gustafson’s law well.

**Gustafson’s law**

“Gustafson estimated the speedup S gained by using N processors (instead of just one) for a task with a serial fraction s (which does not benefit from parallelism). Gustafson's law proposes that programmers tend to set the size of problems to fully exploit the computing power that becomes available as the resources improve. Therefore, if faster equipment is available, larger problems can be solved within the same time.” [1]

f is the fraction of parallelized part of the total work, if N is ONE, the speedup is 1, if increase N bigger, the speedup goes to *N*\**f*. If we can increase parallel part *f*, the speedup approaches N.

**Analysis of the program**

In the program, the parallel part is to throw numbers into correct buckets and perform quick sort on the big bucket. Serial part includes scatter the numbers, dump small buckets to big buckets and gather the result by root process.

**Testing result**

The first test is to increase the number and processors at the same time. We can see the parallel part keeps nearly steady, this makes sense as each processor handles a constant count of numbers. But the total time increase, this might result from the increase of communication cost.

|  |  |  |  |
| --- | --- | --- | --- |
| Numbers | # processors | Total time (seconds) | Parallel part time (seconds) |
| 400,000 | 2 | 0.073252 | 0.066843 |
| 800,000 | 4 | 0.087856 | 0.069855 |
| 1,600,000 | 8 | 0.128181 | 0.078841 |
| 3,200,000 | 16 | 0.300966 | 0.074652 |
| 6,400,000 | 32 | 0.715273 | 0.083101 |

Table 1 Total and parallel time for sorting different numbers on different processors

The second test is keep the numbers fixed (1,600, 000) while adding more processors. Table 2 shows the result. We can see the time of parallel part shrinks at nearly the same ratio of processor increasing. This is also reasonable as the size of the numbers decreasing. The total time decreases too but not the same ratio, this is also caused by inter-processes or inter-nodes communication.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | Total (seconds) | Parallel (seconds) | ***f*** (parallel/total) | Theoretical Speedups | Observed Speedups \* |
| 1 | 0.632468 | 0.606312 | 0.958645 | 1 | 1 |
| 4 | 0.182267 | 0.145682 | 0.799278 | 3.875934 | 3.470008 |
| 8 | 0.128181 | 0.078841 | 0.615076 | 6.594946 | 4.934179 |
| 16 | 0.159366 | 0.036158 | 0.226887 | 10.22613 | 3.968651 |
| 32 | 0.213822 | 0.019955 | 0.093325 | 8.033483 | 2.957918 |

Table 2: Time and speedups for sorting 1,600,000 numbers

\* Observed Speedups = Total time of one processor / Total time of N processors

**Conclusion**

The test demonstrates Gustafson’s Law that the more computing power the bigger problem we can solve without affecting the performance. On the other hand, the overhead cost must be considered in large distributed computing environment. This prevents our testing data fitting the formula well.

**Reference**

[1] Gustafson’s Law [online]. Last accessed August 24, 2018 at <https://en.wikipedia.org/wiki/Gustafson%27s_law>

[2] 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>