Homework 9

Josh Bradt

April 18, 2016

1 Models

Multiplying an $n \times m$ matrix by a vector of length m should take

$$\begin{aligned} \mathcal{M} &= mr + nm(2r + 2c) + mw \\ &= \frac{2m^2}{p}(r + c) + 2mr \end{aligned} \tag{1}$$

where r is the memory access time, p is the number of processes, and c is the time per floating point operation. To simplify the first line, I took advantage of the fact that $n = m/p$, and I assumed that memory reads and writes took the same amount of time. The Blue Waters documentation[1] lists a per-core performance of 19.6 GFLOPS and a peak memory access speed of 102 GB/s. This suggests that the values of the two parameters here are $c = 1/(19.6\,\mathrm{GFLOPS}) = 5.1 \times 10^{-11}\,\mathrm{s}$ and $r = (8\,\mathrm{B})/(102\,\mathrm{GB/s}) = 7.8 \times 10^{-11}\,\mathrm{s}$. (Note that this is probably an unrealistically optimistic estimate...).

The communication time per vector chunk is given by

$$\mathcal{C} = Rn = \frac{Rm}{p}. \tag{2}$$

Here, R is the communication rate, which I found to be $6.8 \times 10^{-10}\,\mathrm{s/B}$ on the last assignment.

This means that the total time for the method using `MPI_Allgather` should be

$$T_{\mathrm{AG}} = p\mathcal{C} + \mathcal{M} \tag{3}$$

since each process must send one chunk and receive the remaining $p - 1$ chunks before performing the multiplication.

The total time for the ring method should be

$$T_{\mathrm{ring}} = p \max\left(\mathcal{C}, \frac{\mathcal{M}}{p}\right). \tag{4}$$

That is, the total time taken is the time taken by the slower of the two overlapping procedures.

[1] `https://bluewaters.ncsa.illinois.edu/node_core_comparison`

Matrix Size	Num. proc.	Chunk size	Allgather time [s]	Ring time [s]
1024	128	8	6.5088×10^{-5}	4.1914×10^{-4}
16 384	128	128	1.0182×10^{-2}	1.0066×10^{-2}
102 400	128	800	4.3615×10^{-1}	3.6520×10^{-1}
1024	256	4	4.0102×10^{-4}	8.3899×10^{-4}
16 384	256	64	5.8219×10^{-3}	5.7230×10^{-3}
102 400	256	400	2.2142×10^{-1}	1.7744×10^{-1}
1024	512	2	9.5201×10^{-4}	1.8370×10^{-3}
16 384	512	32	3.6731×10^{-3}	3.7570×10^{-3}
102 400	512	200	1.1294×10^{-1}	8.9725×10^{-2}
1024	1024	1	1.4019×10^{-4}	3.8991×10^{-3}
16 384	1024	16	2.2910×10^{-3}	4.2529×10^{-3}
102 400	1024	100	5.7398×10^{-2}	4.7342×10^{-2}

Table 1: Results

2 Results

I wrote my own code to perform a matrix-vector multiplication using MPI with the two implementations described in the assignment. I compiled the code on Blue Waters with the default compiler and MPI implementation, and I ran all of the tests in one job. The job requested 64 XE nodes. I ran using the command `aprun -n $NPROC -N 16 -e MPICH_NEMESIS_ASYNC_PROGRESS=1 -e MPICH_MAX_THREAD_SAFETY=multiple ...` where `$NPROC` represents the total number of processes to use in a given run. The data is listed in Table 1 and plotted along with the models in Figure 1.

The results were not entirely what I expected. Both the measurements and the models show little difference between the two methods, while I had assumed that the ring communication method would be significantly faster since it overlaps communication and computation. The models, however, show that the computation time grows with matrix size m as $O(m^2)$ while the communication time only grows as $O(m)$, which supports the idea that the communication time is insignificant compared to the computation time for large m.

Based on these results, I would probably choose the method based on `MPI_Allgather` in most cases since it is much simpler and less error-prone.

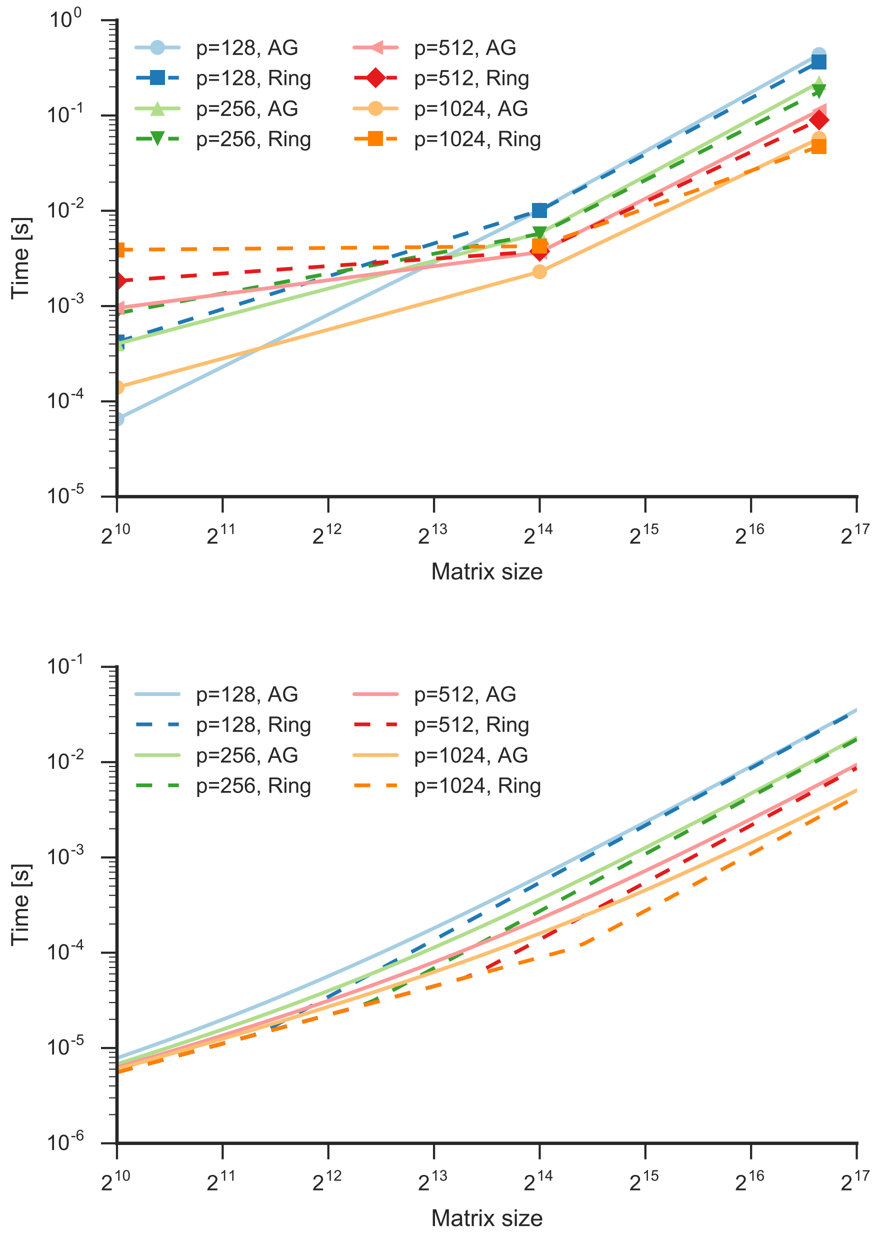

Figure 1: The measured results from the experiment (top) and the models of the expected performance (bottom).