Report for HPC Homework 5

Zhen Li

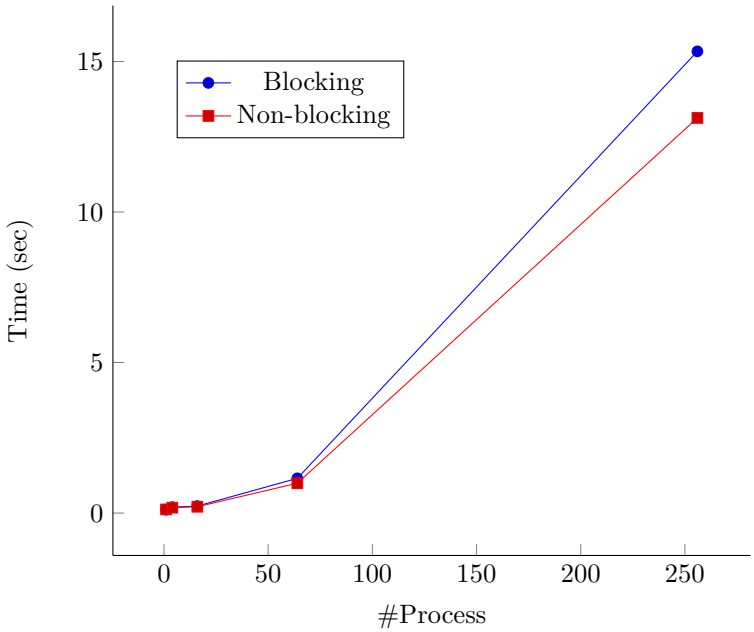
May 1, 2020

1. MPI-parallel two-dimensional Jacobi smoother

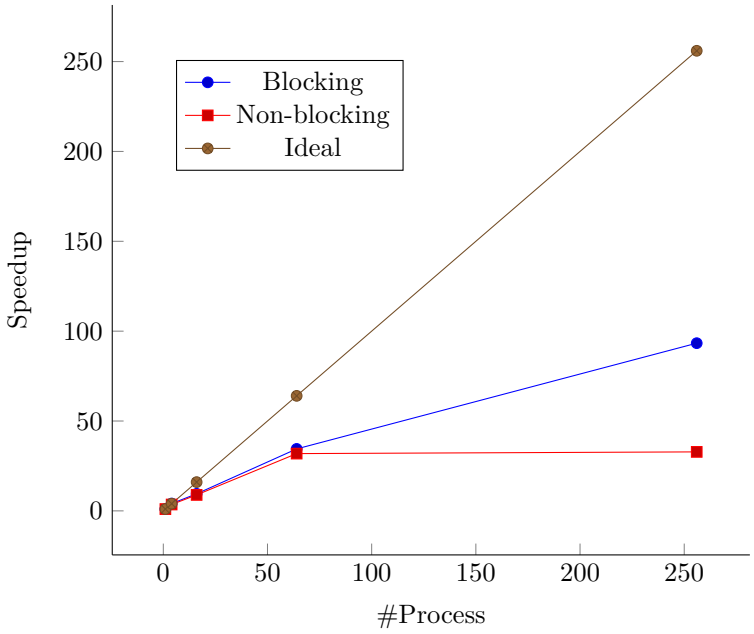
   - Every points reside in the $N_l \times N_l$ matrix should be updated by current process.
   - The leftmost column, rightmost column, topmost row and the bottommost row should be communicated to the adjacent left, right, up, down process.
   - See `jacobi.cpp` for blocking implementation and `jacobi-nb.cpp` for non-blocking implementation.
   - See `jobs/weak/` and `jobs/strong/` for job scripts and raw results generated on Prince cluster. In the file names, bl stands for blocking version, nb stands for non-blocking version, n stands for N, ln stands for N_l and np stands for number of processes in the file names.
   - For convenience, results are shown in the following table:

N	N_l	#Iteration	#Process	Time (blocking)	Time (non-blocking)
100	100	10000	1	0.114321 s	0.120982 s
200	100	10000	4	0.195516 s	0.17968 s
400	100	10000	16	0.23157 s	0.210041 s
800	100	10000	64	1.154025 s	0.991806 s
1600	100	10000	256	15.336503 s	13.124876 s
25600	25600	100	1	163.796906 s	152.059074 s
25600	12800	100	4	40.288567 s	42.902798 s
25600	6400	100	16	17.246715 s	17.122933 s
25600	3200	100	64	4.750117 s	4.771576 s
25600	1600	100	256	1.75543 s	4.632839 s

     where the upper half is for weak scalability comparison and the lower half is for strong scalability comparison.
   - For weak scalability, the timings are plotted below. As you may see, this algorithm is not weakly scalable since the running time significantly increases as N and #Process grows.



- For strong scalability, the speedups are plotted below. The series of blocking version some how resembles linear speedup, but it is still not ideal. For the non-blocking version, the running time of #Process=256 is almost the same with that of #Process=64, which might be caused by instability of the system. I have to mention that a previous buggy test of non-blocking version with #Process=256 does suggest the running time is within 2 sec, which is what I expect. Since the nodes required for #Process=256 are down now, I am not able to run the test again.



- For comparison between blocking and non-blocking version, actually their timings do not differ much but the non-blocking version still outperforms the blocking one when #Process and N both getting bigger.

2. Parallel sample sort

- See `ssort.cpp` for implementation.
- See `jobs/ssort/` for job scripts and raw results.
- The results are listed below. They are obtained with the configuration of #Node=12 and #TaskPerNode=14, that is #Process=168.

N	P	Time
10000	168	0.068062 s
100000	168	0.718512 s
1000000	168	20.496098 s

- You may notice that for $N = 1000000$ the running time increases a a lot compared to smaller N's. I guess it is also because of instability of the system.
- Note: the output of all bucket is removed from the repository because they are too large (about 2 gigabytes). You may generate them by running the program.