Ran Pang

Parallel and Distributed Systems Program Assignment #3 Report

Gaussian Elimination with MPI, Pthreads and OpenMP

This program assignment means to implement Gaussian Elimination using MPI, OpenMP, and compare the performances of programs using MPI, OpenMP and Pthreads. Please execute *pthread_test.sh*, *mpi_test.sh* and *openmp_test.sh* to compile and run the programs respectively. The printed results are shown as pairs of number of threads/processors vs. execution time in seconds.

Both the Pthreads and MPI programs use cyclic assignment instead of block assignment to distribute sub matrices among threads/processors. The MPI programs use pipe-lining instead of broadcasting to send message among processors. All three types of programs do Gaussian Elimination on a 4000x4000 randomly generated matrix. In order to check the correctness of the programs, each matrix among a sample of randomly generated matrices are processed by each program plus the sample program */u/cs458/apps/gauss/seq/gauss*.

The settings of the two platforms are shown in the tables below.

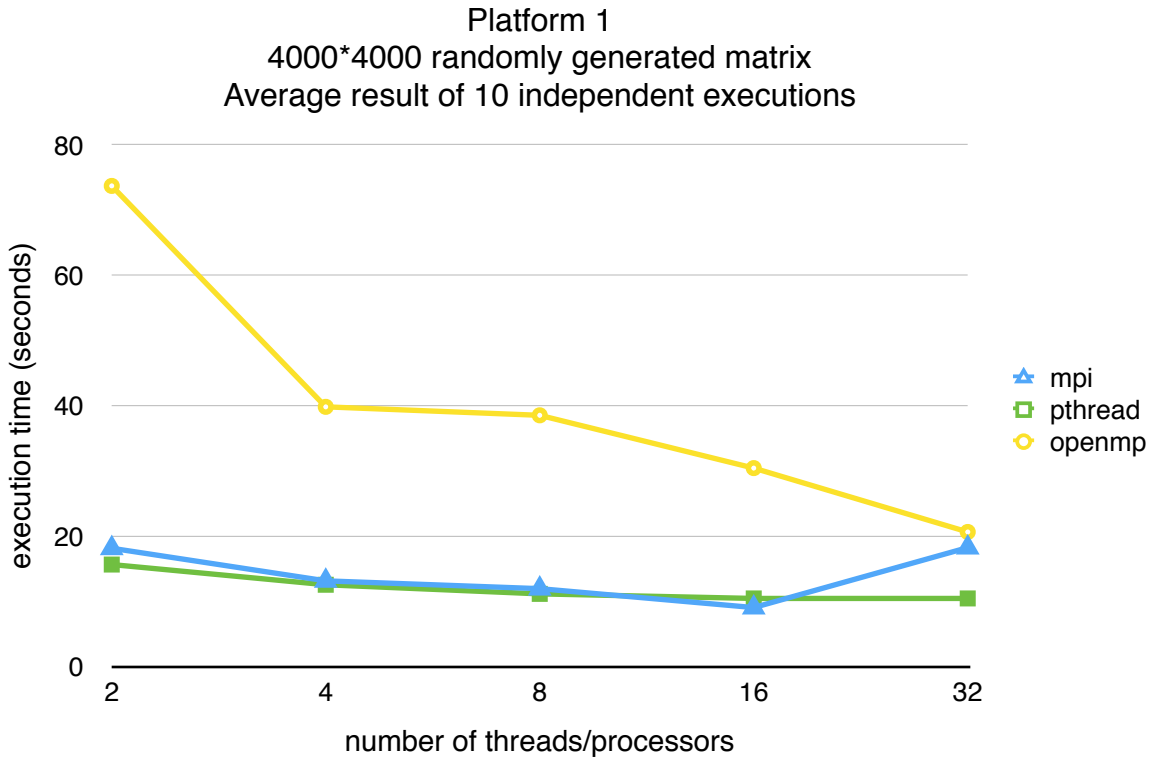
Platform 1

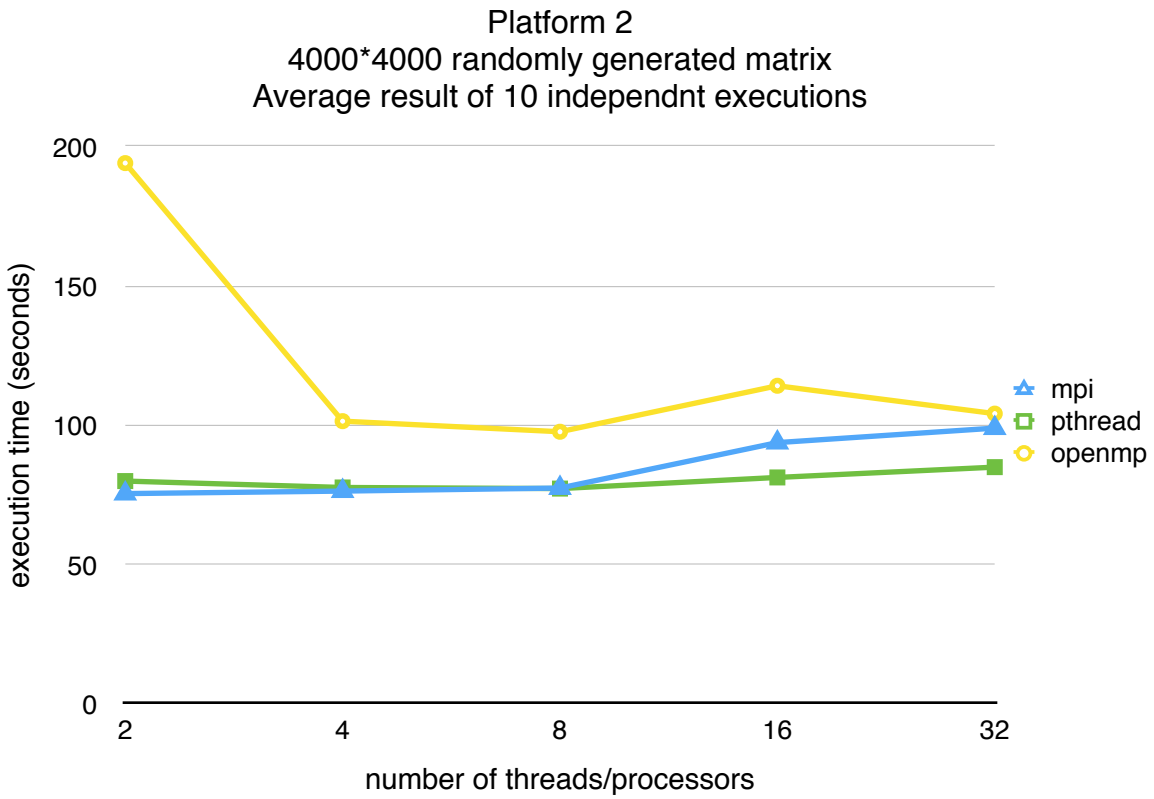
machine	cycle3.cs.rochester.edu
OS	Linux 3.18.7 x86_64
GCC	4.8.3
processor model name	Intel Xeon CPU E5-2430 2.20GHz
number of processors	24
number of cores per processor	6
cache size	15360 KB
memory available	63127064 KB

Platform 2

machine	cycle1.cs.rochester.edu
OS	Linux 3.18.7 x86_64
GCC	4.8.3
processor model name	Intel Xeon CPU 3.00GHz
number of processors	8
number of cores per processor	2
cache size	2048 KB
memory available	4047100 KB

The performances of three programs on the two platforms are shown in the figures below.





As shown in the figures, performances in the two platforms indicate some similarities. When the number of threads/processors is very small (e.g. 2), Pthreads programs and MPI programs perform no significant difference and remarkably better than OpenMP programs. As the number of threads/processors grows, the performance of Pthreads programs and MPI programs increase slightly and make a turn at a point where the number of threads/processors exceed the maximum threads/processors available in the platform. On the contrary, OpenMP programs increase performance dramatically as the number of threads/processors grow so that it reaches near the performance of Pthreads programs and MPI programs.

Why OpenMP runs much slower than the other two methods when the number of threads/processors is small? My guess is that it has more things to do with the implementation of the Gaussian Elimination algorithm than with the parallel schemes that differs among the programs. OpenMP does work because it is observed that the sequential Gaussian Elimination program, which the OpenMP program is based on, takes just about two times longer to execute than the OpenMP program on 2 threads. Moreover, the performance of OpenMP program shows reasonable trend in the figures above. It is also observed that on a single thread, the Pthreads program still runs significantly faster than the sequential program that form the base of the OpenMP program. Unfortunately I fail to detect the cause of these weird behaviors so far. Anyway all three programs shows correct result on test matrices.