Homework 4 Report

1 Test Environment

I tested my code on department's ix server. It has two sockets with AMD Opteron 6376 on each socket. Each CPU has 8 cores, 16 threads. So, there are 32 hardware threads in total.

2 Test Results

2.1 Convergence Test Result for Tolerance of 1e-6, Using 32 Threads

Max Iteration	Time (seconds)	norm(b-A*x)	Converge
13,000	25.515200	5.743765e-05 > 1e-6	False
13,250	27.533186	3.585957e-05 > 1e-6	False
13,500	26.683842	6.690461e-06 > 1e-6	False
13,750	26.256778	4.147648e-06 > 1e-6	False
14,000	27.199833	9.928136e-07	True
15,000	26.453255	9.593049e-07	True

Table 1. Convergence Test

2.2 Performance vs. Threads, Using Tolerance of 1e-6, 14,000 Max Iterations

Threads	Time	norm(b-A*x)
2	251.994506	9.512838e-07
4	127.116177	9.076329e-07
8	67.801198	9.583944e-07
16	39.024688	9.372702e-07
32	26.754807	9.147343e-07

Table 2. Performance Scaling for number of threads

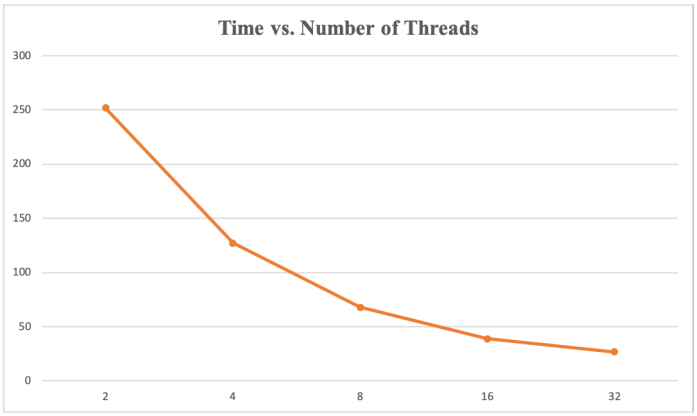

Figure 1. Performance Scaling for number of threads

2.3 Function cg() without omp tasks vs. Function cg() with omp tasks

Threads	Time	norm(b-A*x)
2	262.271384	1.205389e-06
4	137.757589	9.571862e-07
8	77.193165	9.879008e-07
16	46.175155	9.082830e-07
32	37.390880	9.578399e-07

Table 3. cg() with omp tasks

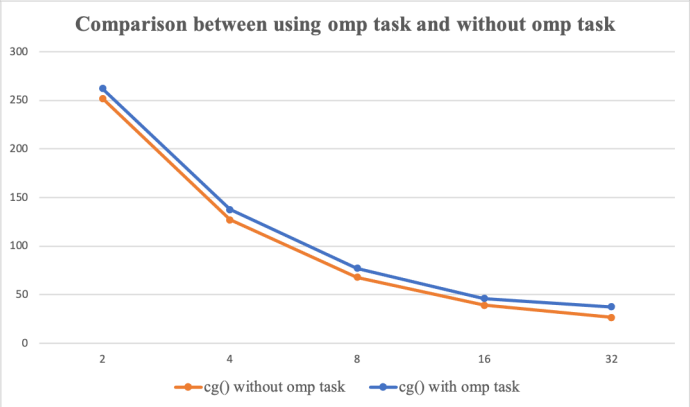

Figure 2. cg() without omp task vs. cg() with omp task

3 Findings

- For my implementation, CG does not converge for tolerance of 1e-6 when max iteration is below 14,000.
- The performance gain almost doubles when using double amount of threads.
- Since almost each step depends on the previous step in CG, there is not much opportunity for task-based parallelism. I attempted to add additional parallelization by using omp tasks to simultaneously calculate:
  1. x_(k+1) = x_(k) + alpha_(k) * p_(k)
  2. r_(k+1) = r_(k) - alpha_(k) * A * p_(k)

  However, by adding omp task will result approximately the same performance. This is most likely for two reasons. First, there is overhead for creating omp tasks. Second, most of the computational work has been done by vec_add() in parallel.