APC 524 Homework 4
Parallel Programming

Evan Leister

11/19/12

1 Problem Statement

Consider the heat diffusion equation:

$$\frac{\partial T}{\partial T} = \kappa \nabla^2 T \tag{1}$$

on a domain $0 \leq x, y \leq \pi$ with the following boundary conditions:

$$\begin{aligned} T(x,0) &= \cos^2 x \\ T(x,\pi) &= \sin^2 x \\ T(0,y) &= T(\pi,y) \end{aligned} \tag{2}$$

This equation can be solved as an initial value problem with an explicit timestepping scheme:

$$T_{i,j}^{n+1} = T_{i,j}^{n} + \Delta t\, \kappa \left(\frac{T_{i-1,j}^{n} + T_{i+1,j}^{n} + T_{i,j-1}^{n} + T_{i,i+j}^{n} - 4T_{i,j}^{n}}{\Delta x^2} \right) \tag{3}$$

If the end time desired is $t = 0.5\, \pi^2/\kappa$ and if the grid spacing is equally spaced, the following timestep criterion applies:

$$\Delta t < \frac{\Delta x^2}{4\kappa} \tag{4}$$

The above system was solved using three different implementations. One implementation was a simple initial value problem where all values were calculated in serial. A second implementation involved programming in OpenMP in order to parallelize some of the work. A third implementation used OpenMPI in order to pursue a different parallelization.

2 Results

Each of these solutions took the form of figure 1:

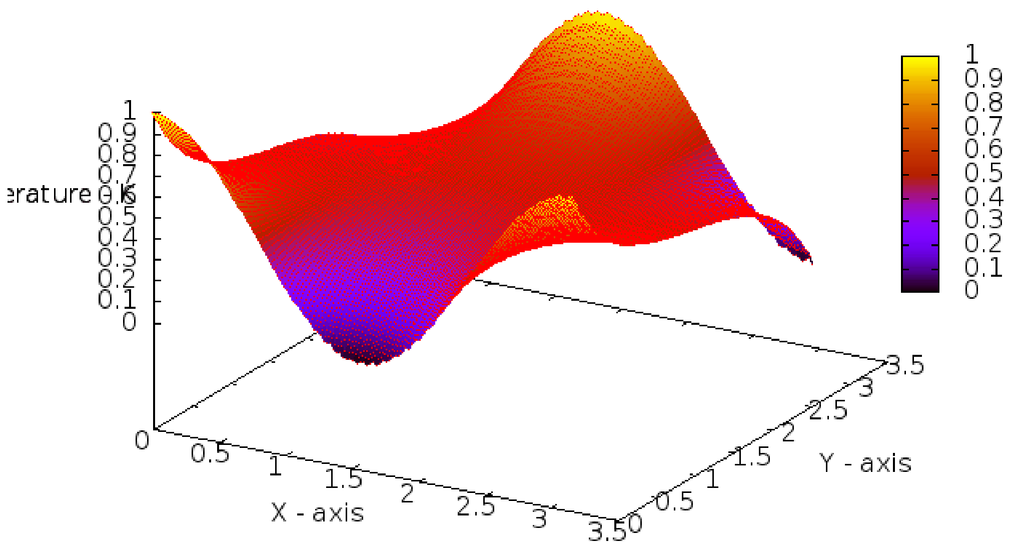

Figure 1: Contour plot of temperature

Each solution had an average temperature of 0.497.

These solutions were profiled using increasing amounts of grid cells on the Adroit computing cluster. The results are collated in table 1.

Run Type	128	256	512
Serial	7	128	3006
OMP: 1 thread	7	128	3440
OMP: 2 threads	4	64	1889
OMP 4 Threads	2	33	754
OMP 8 threads	1	17	363

Table 1: Adroit runs, time in seconds.

These results were not completed due to time and queue restrictions. The results in table 2 were computed on a 4-core Lenovo i7 with simulated hyperthreading. These results with 8 threads are not representative of what could be possible on a larger cluster with more processors. Note that the MPI implementation was completed later than was possible to implement a complete profile test.

Run Type	128	256	512	1024
Serial	2	20	338	5600
OMP 1 thread	1	21	352	5943
OMP 2 threads	1	11	186	3270
OMP 4 threads	0	7	100	3171
OMP 8 threads	1	7	137	3296
MPI 1 threads	1	18		
MPI 2 threads	1	11		
MPI 4 threads	1	6		

Table 2: Intel i7 runs, time in seconds

One immediate advantage of OpenMP is how simple it is to implement. Obtaining a factor of two speed gain with a small piece of parallelization is not insignificant. In contrast, MPI did not perform much better with slice decomposition but required a significant amount of extra programming. For this problem, MPI has potential to be much faster if a more intelligent domain decomposition were used. As was discussed in lecture, a square domain decomposition has a much smaller buffer for passing messages in between processes. The large buffer size significantly limited the speed and reliability of larger runs.