HW 3: OpenMP Programming

In this assignment you will parallelize an iterative solver for the Poisson equation for a shared memory machine using OpenMP.

The problem is:

\[ \Delta u = u_{xx} + u_{yy} + u_{zz} = f \text{ in domain } \Omega \]
\[ u = g \text{ on } \partial \Omega \]

You are given an initial serial version of a 3D solver using Jacobi’s method. The solver uses a second-order finite difference scheme to approximate the derivatives. The domain \( \Omega \) is a cube from \([x_{lo},y_{lo},z_{lo}]\) to \([x_{hi},y_{hi},z_{hi}]\). Use loop level parallelism to speed up the run time of the code as much as possible.

You should time your code for domain sizes that include 20, 40, 80, 160 on a side. Note that you cannot directly compare runs with different grid sizes because it will take more iterations to converge on a finer grid. (Also, for the largest sizes you probably don’t want to run as many iterations). You should make speedup plots using between 1 and 16 threads. (Note: the queue for the class on Gauss has a limit of 16 nodes. While debugging you can use one node of the Dell cluster which supports up to 8-way shared memory parallelism). As always feel free to optimize in any other ways you find, including compiler flags, and improvements to the algorithms. Experiment with using either the 2 norm or the max norm to determine convergence, and report the results. If you have time you can also experiment with using Gauss Seidel (red-black variation) and report on that. Hand in your compile line along with your discussion of the things you tried (both successful and unsuccessful) and graphs. Please explicitly include your timing results along with your speedup plot for easy comparison.

Using Gibbs/Gauss

Gibbs is the front-end to the shared memory machine Gauss, an SGI Altix with 64 nodes. To run interactively you must use the following qsub command, which is slightly different than on the Dell.

```
qsub -I -q class -W group_list=class -l nodes=1:ppn=16,walltime=00:30:00
```

Note that to invoke the OpenMP compiler the option -openmp is needed on both the compile line and the link line. You will have to remember to load the compiler module (cc10) to get the icc compiler.