Final Project Suggestions

The final programming project is of your choosing, and is ideally an application you are actually interested in parallelizing. It can use any of the programming models we will look at in class. Group projects of 2 to 4 people are encouraged. If you already have a project and are looking for another team member, or if you are looking for an idea and are interested in working with someone from class, send us email with a description.

If you have nothing in mind, we list possible projects below. You can also talk to other faculty at Courant with numerical interests (e.g. Mike Shelley, Denis Zorin, Charlie Peskin). If your interests really lie elsewhere, come talk to us to try to find a suitable project.

Preliminary ideas for project proposals should be discussed or emailed to us by Oct. 7, to give us time to iterate. The final proposals are due Oct. 21. Here are some ideas:

1. Parallelization of AMRCLAW (adaptive version of Clawpack which solves hyperbolic conservation laws in 2 or 3 dimensions using a wave propagation algorithm) using OpenMP. Try two approaches to parallelization - one based on a finer loop level granularity and a higher level patch-based approach. (Should have enough people on the team to try both approaches).

2. Use tools (e.g. profile-based optimization, PAPI) to characterize performance and optimize an already parallel code that solves the Euler equations in complex geometry.

3. Build a parallel $O(n^2)$ fast direct solver for linear systems of equations coming from discretizations of integral equations.

4. Using CUDA, implement a mixed-precision linear solver for solving dense linear systems by using a single-precision solver as a preconditioner for FGMRES (see the tech report “Using FGMRES to obtain backward stability in mixed precision” by Arioli and Duff).

5. A PETSc-based project (parallelize an existing solver or a new one).

6. A GPU-based project (talk to Denis Zorin).

7. A Parallel Language project: compare and contrast a simple solver in UPC or Titanium with an MPI implementation.