NIMROD Processor-Layer Scaling on Seaborg

Carl Sovinec, University of Wisconsin, October 4, 2004

In August, I distributed an e-mail recommending that NIMROD users test the speed of their computations with different numbers of grid blocks. Scaling tests performed for the FY05 NERSC ERCAP request showed that the finite element computations run faster with more (smaller) grid blocks and that cache usage is the likely culprit. However, I did not attempt to get maximum scaling from toroidal decomposition, having been disappointed with the IBM machines after being accustomed to very good scaling on the T3E. With the resolution milestones for the ELM study, toroidal decomposition scaling is worth another look. In addition, Chris Carey has been investigating different options for performing parallel communication over NIMROD's processor-layers, and this has brought the associated latency to our attention.

While increasing the number of grid blocks with a fixed finite-element mesh can be very good for the finite element scatter operations, the number of layer-communication calls increases proportionately. On machines where this type of communication is dominated by bandwidth (as on our cluster), there is essentially no penalty, since the total amount of data is the same. However, this is not the case on Seaborg for at least some problems of interest. Consider the results shown in Fig. 1. The plots show the FFT time, the finite element computation time (which is inclusive of nearly all FFTs), and the total run time for a nonlinear problem with an 18×24 mesh of biquartic elements. (Actually, the FFT timing calls were modified to record just the associated layer-communication time—the serial FFTs are trivial in comparison.) Matrix-free solves are used in the velocity and temperature advances in these tests (continuity='full' and p_model='aniso1', respectively) to represent FFT-intensive simulations. It's worth noting that the number of cg iterations (with FFT calls during each iteration) for the temperature advance increase from 6 to 8 to 17 per step (rough averages); this is an application-relevant scaling but not a computer-science scaling. With 11 Fourier components and 11 processors, the layer-communication (FFT) time is trivial, so the 6-block computation (dashed traces) wins by better cache usage. However, with 43 Fourier components and 43 processors, the layer-communication time is a significant fraction of the total time. The extra latency adds significant computation costs to the 6-block computation, and the 1-block computation is faster.

We are therefore led to investigate whether it is possible to improve the finite element computations with fewer large grid-blocks. What dominates the FE compute time (exclusive of FFTs) with large grid-blocks is scatter operations that take integrand information from the numerical quadrature points and accumulates it in the rblock matrix data structures. The rearranging that occurs here is pretty messy, so it seems reasonable that small array sizes help avoid thrashing cache. A new approach is to do the accumulation in a temporary array that has the same structure as the data coming out of the integrand routines and then going through the messy rearranging only once. For biquartic elements, there are 25 numerical quadrature points per element, so the messy bit is done in 1/25 of the time. Timing results with this approach are shown in Figure 2. The FFT times are essentially the same, but the finite element computations are faster. The total time for the 43-layer calculation with 1 block is 1157 s with the new routines, 1371 s with the old routines, and 1711 s with the old routines and 6 blocks.

As an aside, the old routines run faster on some small problems, so the latest implementation contains both and does CPU timing of both approaches 'on the fly' to apply the best of the two for each matrix and for each grid block.

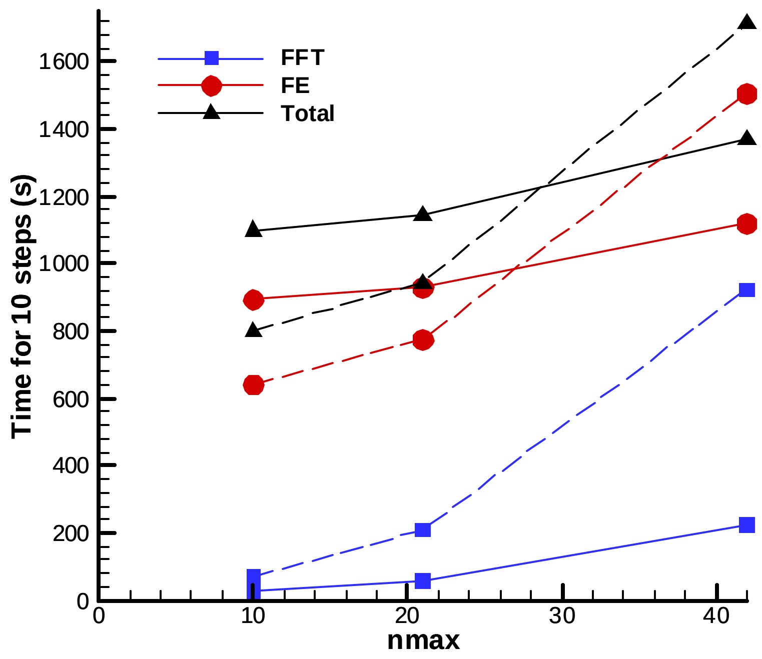

Figure 1. Weak parallel scaling for toroidal resolution (nprocs=nmax+1), where the computations include Fourier components 0≤n≤nmax. The solid lines show results from computations with one grid block containing the entire 18×24 (radial×poloidal angle) mesh. Dashed lines show results from simulations with the mesh divided into six grid blocks. Each element is biquartic.

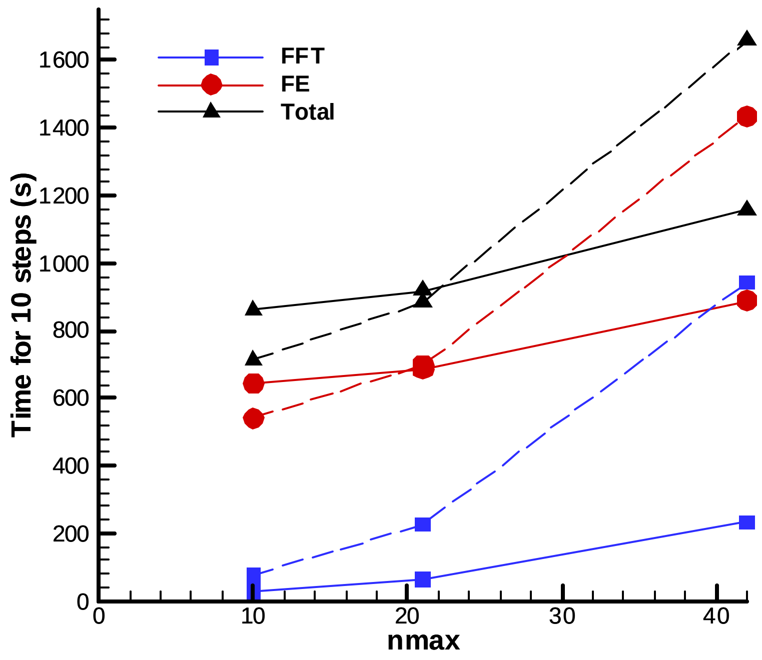

Figure 2. Weak scaling results with revised numerical integration routines for rblock matrices.

It is, of course, possible to apply NIMROD with larger numbers of Fourier components. At the next power of 2 for the number of toroidal cells, there are 86 Fourier components after dealiasing for quadratic nonlinearities. In the test case considered here, the average number of iterations during the temperature advance increases to about 80, velocity iterations increase, and the FFT, FE+FFT and total times increase to 1613 s, 2538 s, and 2964 s, respectively, with 86 Fourier components. Since the layer decomposition works by further decomposing the poloidal plane (beyond any poloidal decomposition, which isn't used here), the layers are running out of space after the mpi_alltoallv calls. There are only about 5 poloidal cells per processor layer ($18\times24\div86$). With 3 grid blocks, the CPU times are yet much larger! The present algorithm needs to have a larger poloidal mesh in order to scale to 86 layers. Alternatively, one can put more than one Fourier component per layer.

In summary, we have found that scaling the number of processor layers with Fourier components is practical on Seaborg, provided that one is conscious of latency and the number of elements left on each layer after the mpi_alltoallv calls. Revisions to the finite element scatter operations help reduce the number of cache-unfriendly data rearrangement loops. This improves performance with large grid-blocks, where the impact of latency associated with layer communication is smallest. Ongoing work includes developing alternatives to the mpi_alltoallv-based layer decomposition and further reduction of the impact of communication latency.