

## Tasks Completed This Week

This week, I was able to get the direct solver for PETSc installed and running, and performed some tests with the sample file yielding.xml. This file is a sample model file provided by Gale that can be used to test the functionality of your installation, as sample output is available. Before the direct solver, I was having issues with getting the program to scale well to multiple processors, and was seeing a slow down instead of a speed up when running on more than 4 processors. The direct solver, as mentioned in last weeks report, can be used instead of the default, GMRES solver that Gale uses. The GMRES, or generalized minimal residue method, is a form of iterative solver. As a result, it can take much longer to converge on smaller sets of data. When the program is split among more processors, each processor has to deal with a smaller set of data, so it is likely that the slow down can be attributed to the weakness of the GMRES. In the figures below, the results from the direct solver versus the GMRES method that is used by default by Gale are shown side by side, with the direct solver on the right.

## Speedup Analysis

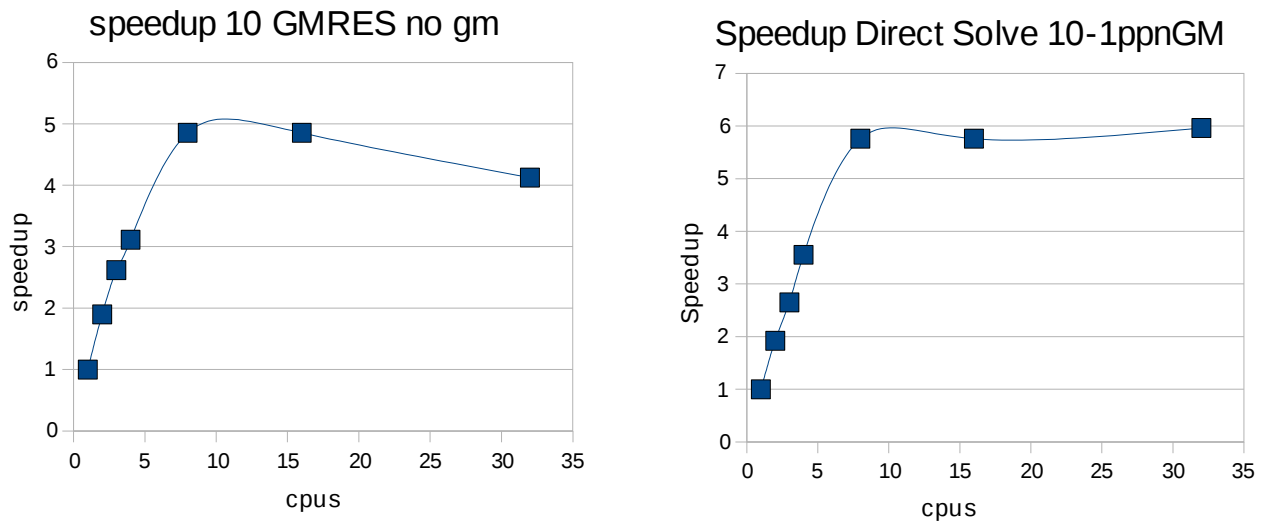


Figure 1. Speedup of 10 time step simulation, no output, myrinet

At 10 time steps, which is a relatively very short simulation, the difference in speedup between the direct solver and the GMRES method is slightly evident in a few ways. First off, the over all speedup for the direct solver is higher, peaking at about 6, while the GMRES method peaks at around 5. There is also a slight decline in speed up from 8 to 32 processors for the GMRES solver.

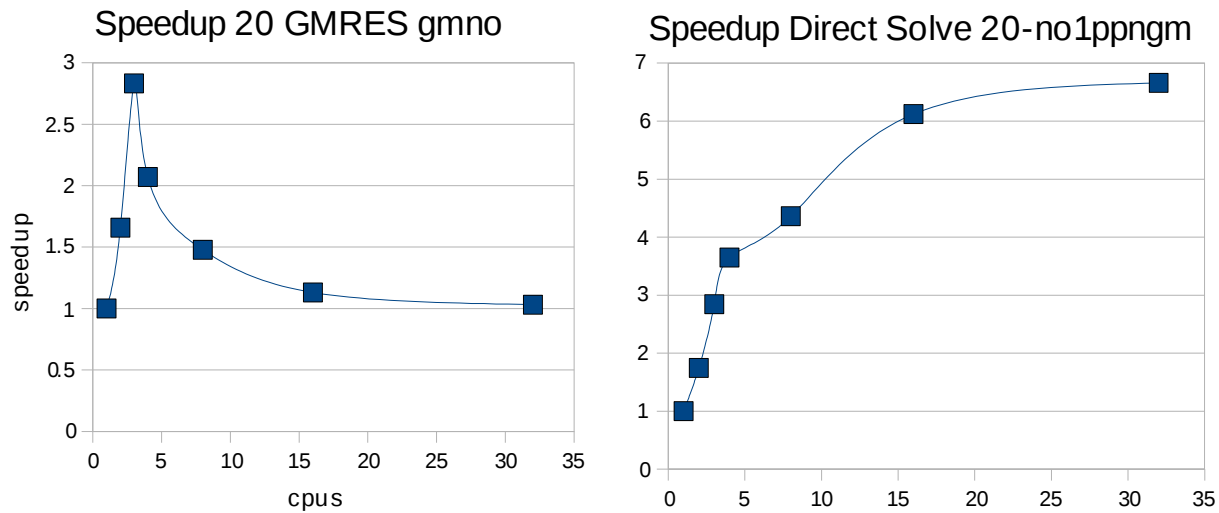


Figure 2. Speedup of 20 time step simulation, no output, myrinet

In figure 2, the difference between the GMRES method and the direct solver becomes much more apparent. The speedup is about the same until it reaches 4 processors, at which point the GMRES method slows down a lot. By the time the curve reaches 32 processors, the speedup is 1.03. The direct solver however, scales up and keeps getting faster and faster as more processors are used, reaching a maximum speedup of around 6.6.

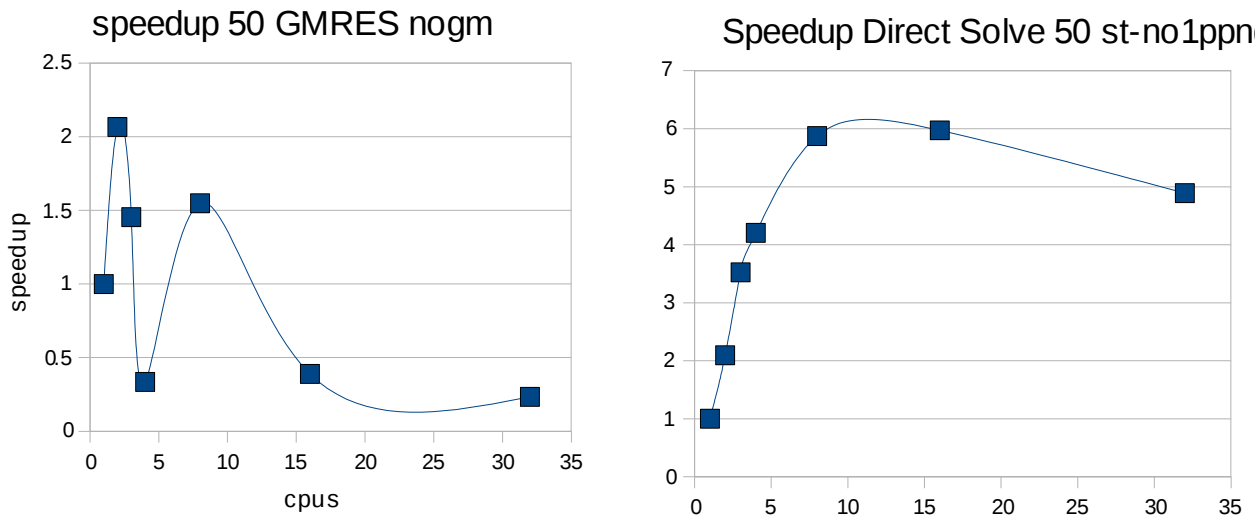


Figure 3. Speedup of 50 time step simulation, no output, myrinet

The speedup for 50 time steps and the GMRES method, as shown on the left in figure 3, is very peculiar. It drops off as soon as more than 2 processors are used, but jumps up once it gets to 8 processors. At first I thought that this was a mistake, but after 15 runs with the same results, I assumed that it must be correct. The only explanation that I can come up with is that for 3 and 4 processors, a very large amount of data has to be passed to each processor over the network, creating a bottleneck. This could explain why when it runs on 8 processors it sees a minor speedup, as the data set being passed is not as large per processor. Using the direct solver results in a speedup that is much more reasonable, however it does tail off after 16 processors. This is almost certainly limited by the network

speed, as with 32 processors, there is much more inter-processor communication, and would explain the slight decline in speedup. More investigation is needed on higher numbers of processors to determine if this is true.

### Efficiency Analysis

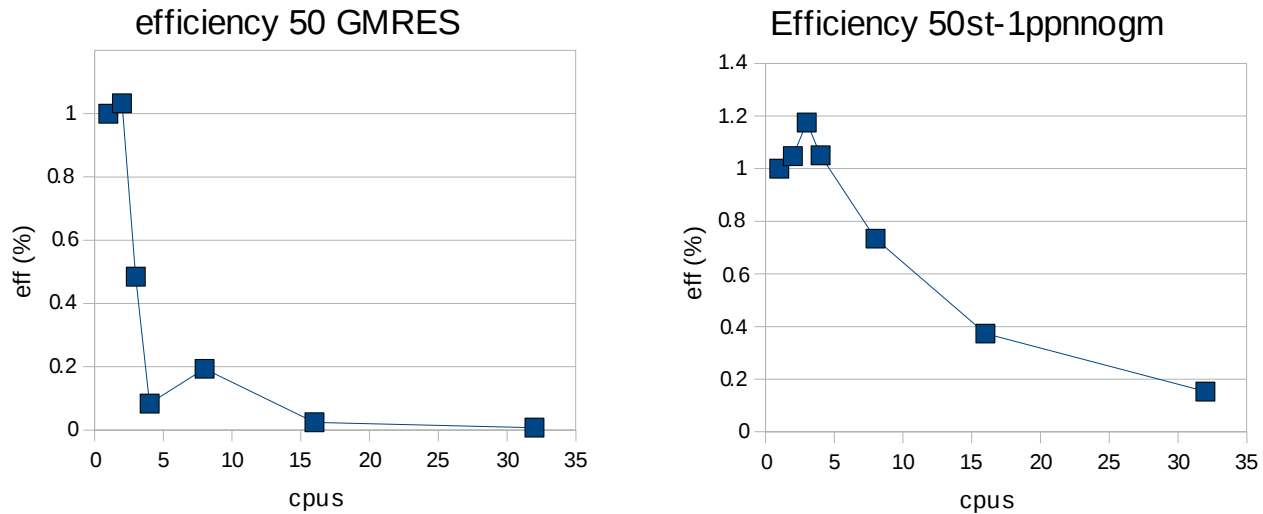


Figure 4. Efficiency for 50 time step simulation, no output, myrinet

Figure 4 compares the efficiency of the two methods running at 50 time steps, which is a more realistic number of simulations for a real scientific application. The GMRES method has very poor efficiency, with all of the jobs that ran on 4 or more processors seeing less than 20% efficiency. The direct solver, on the other hand, saw some very slight super-linear speedup on 2, 3, and 4 processors. The efficiency declines when running on higher processor counts, but this is to be expected due to the program not being 100% parallel. The super-linear speedup is probably due to the extra cache and less memory access providing more of a benefit than the repeated sequential tasks being performed multiple times. While an efficiency of 15% on 32 processors is not too spectacular, it is much better than the .7% efficiency exhibited by the GMRES method.

### Tasks for Next Week

While I have run the simulations provided to me by Ben for both the 2D and 3D examples, I have not collected all of the data required for a complete analysis yet of it's performance on Gale. The supercomputer is currently running a 1000 time step 2D simulation on 8 processors right now, and by next week I hope to have run successfully both the 2D and 3D simulations. Also, I would like to have completed analysis of the program running on various processor counts to see where it is fastest while maintaining a good efficiency.

Also, hopefully I can start on the next part of my project, which involves figuring out an automated method of visualizing data output from FLAC3D.

