Andrew Adams
Applications of Parallel Computers
Final Report
5-8-15

A Case Study in Parallel Scaling

In this class, we have dealt mostly with parallelization and scaling of single big data jobs; however, much of the work performed on computing clusters consists of smaller embarrassingly parallel problems. When dealing with problems of this nature, a different paradigm is required in order to achieve the highest utilization of resources. In this report, we will discuss how we determine suitable job sizes as well as look at a case study in scaling a GROMACS Molecular Dynamics simulation on the Xeon Phi cards of the Stampede computing cluster.

The Xeon Phi cards on Stampede contain 61 cores which are able to handle 4 openmp threads each. These cores live in a shared memory environment that allows them to run 61 MPI instances, 244 openmp threads (in one MPI instance), or anything in between. GROMACS supports hybrid openmp/MPI jobs with a maximum of 32 threads. We selected an example system from our research and performed strong scaling experiments with 100 ps of simulation time. The results are summarized in figure 1. As we can see, the system does not scale well beyond 10 cores and shows a preference for more MPI instances over threads.

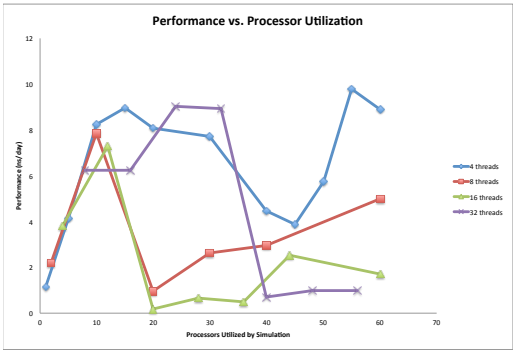

figure 1

A naive person would look at figure 1 and choose to run their jobs with 55 cores and 4 threads, however in an embarrassingly parallel application it might be beneficial to limit the size of the job to allow more throughput. While Stampede always gives jobs exclusive use of at least one node, our own Cowboy allows more than one job to run simultaneously on a single node. In this case, it would be beneficial to look at performance on a per-processor basis as in figure 2. This figure clearly shows that the most efficient way to run jobs is on a single core with 4 threads. This would potentially allow 61 jobs to be run at the same time allowing for a much larger throughput.

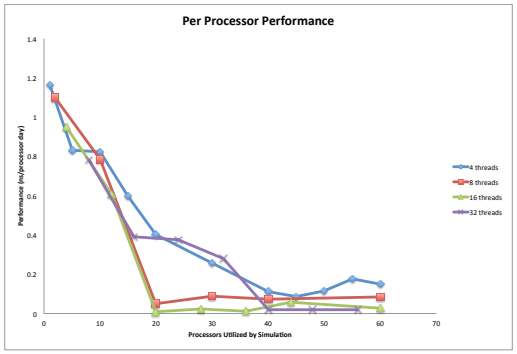

figure 2

In these simulations we have found every size of system to scale differently. This necessitates performing scaling experiments similar to the one presented here for each system of interest. Our philosophy in running these experiments is that we always want to take advantage of super-linear speedups (which can be attributed to an increase in available cache across more processors), and we can take advantage of linear speedups. When the system allows for it, it is always preferable to make jobs smaller to allow other jobs to be complete in parallel if the system is found to scale at less than a linear rate.