Sputnik: Automated Decomposition on Heterogeneous Clusters of Multiprocessors

Sean P. Peisert
Professor Scott B. Baden, Advisor

Goal 1: To improve machine utilization and software performance on heterogeneous clusters shared-memory multiprocessors.

Goal 2: To enable a user to write software for a heterogeneous cluster as if the machine was homogeneously. In a word: easier.

Approach: Provide an application programmer interface (API) that enables the programmer to treat a heterogeneous cluster of nodes as if it is homogeneous. An application enters two stages of execution: Discovery and Optimization.

The Sputnik Model:

Cluster Discovery Stage: Runs the kernel inside a shell to test the program on each individual node to determine the characteristics and "power" of each node.

Cluster Optimizer Stage: Uses the results from Cluster Discovery to optimize running the program on the cluster.

Sputnik Implementation:

Cluster Discovery obtains the times for a run of the kernel on each individual node. It also determines the optimal number of OpenMP threads per node to run with.

Cluster Optimizer uses the times for each node to repartition the problem to run optimally on the cluster, balancing out the running times and more fully utilizing the cluster.

Results:

• Obtained on a pair of SGI Origin2000 machines, at NCSA, one with 256 processors and the other with 128 processors.
• Benchmark is a red-black iterative solver of Poisson’s equation using Gauss-Seidel’s method.
• Slower of the unoptimized times shows speedup close to the theoretical peak, when compared to the slower of the optimized computation times.
• Indicates that adjusting the number of threads on each node and repartitioning the data can be successful with this particular benchmark, though other optimizations could be implemented in the future to expand the breadth of problems that the method Sputnik uses can solve.
• Disparity between times of unoptimized computation indicates how much time is wasted.

Future work:

• Run on a Sun HPC SMP cluster
• Run on teraFLOP P-scale machines, including Blue Horizon
• Other applications
• Other kinds of optimizations (e.g. cache)
• Dynamic optimization for Grids and Metacomputers

Implementation: A C++ class library implemented using the KeLP Infrastructure plus some OpenMP code.

This work is supported in part by the Cal Micro Program, Sun Microsystems, the San Diego Supercomputer Center, The National Partnership for Advanced Computational Infrastructure, and the National Computational Science Alliance. Special thanks to Faisal Saied (NCSA), Louis Hoyenga (NCSA), Emil Sarpa (Sun Microsystems), and Mike Vildibill (SDSC).

http://www-cse.ucsd.edu/groups/hpcl/scg/kelp/ § http://www.sdsc.edu/~peisert/research.html