Assessing HPCS productivity with Purpose-Based Benchmarks at MHPCC

Motivation

High performance computing systems are being applied to an increasingly wide variety of domains, including nuclear physics, crash simulation, satellite data processing, fluid dynamics, climate modeling, bioinformatics, and financial modeling. They are also becoming radically less expensive: the recently announced $5.8 million dollar MACH-5 supercomputer based upon the Apple G-5 will execute 24 teraflops per second. In comparison, the current fastest supercomputer, Japan's Earth Simulator, executes 36 teraflops per second but costs over $350 million dollars.

Unfortunately, these numbers do not tell the whole story. The DARPA High Productivity Computing Systems Program [1] and the Workshop on the Roadmap for the Revitalization of High-End Computing [2] note that dramatic increases in low-level HPCS benchmarks of processor speed, memory access, and dollars/flop do not necessarily translate into increased development productivity. In other words, while the hardware is clearly getting faster and cheaper, the developer effort required to exploit these advances is becoming prohibitive. Software engineering, not hardware engineering, is becoming the limiting factor in the advance of HPCS.

To complicate matters further, the design, implementation, development, and maintenance of HPCS software systems can differ in significant ways from the systems and development processes more typically studied by the software engineering community:

- The requirements often include conformance to sophisticated mathematical models. Indeed, requirements may often take the form of an executable model in a system such as Mathematica, and the implementation involves porting to the HPCS.

- The software development process, or "workflow" for HPCS application development may differ profoundly from traditional software engineering processes. For example, one scientific computing workflow, dubbed the "lone researcher", involves a single scientist developing a system to test a hypothesis. Once the system runs correctly once and returns its results, the scientist has no further need of the system. This contrasts with standard software engineering lifecycle models, in which the useful life of the software is expected to begin, not end, after the first correct execution.

- "Usability" in the context of HPCS application development may revolve around optimization to the machine architecture so that computations complete in a reasonable amount of time. The effort and resources involved in such optimization may exceed those required for initial development of the algorithm.

To address these issues, I have begun collaborative research with SUN Microsystems over the past year with two broad goals: (1) The study of HPCS development from a software engineering perspective, with the goal of identifying key HPCS productivity bottlenecks and subsequent development of new tools or techniques to overcome them; and (2) The development of more comprehensive measures of HPCS productivity for use in evaluating new and existing HPCS architectures, which take into account not only the low-level hardware components, but the higher-level development costs associated with producing usable HPCS applications using the architecture. Our work together so far has included the development of a framework for HPCS productivity evaluation that uses "Purpose-Based Benchmarks" [3], and the organization of a workshop on HPCS and Software Engineering [4].

Assessing HPCS productivity with Purpose-Based Benchmarks at MHPCC

Proposed research and contributions

To accelerate the pace of this research, and to create new opportunities for UH and MHPCC participation, I request graduate assistant funding for Michael Paulding, an ICS Ph.D. student, for the 2004-2005 academic year. Michael enrolled in EE 603, Understanding and Utilizing Parallel Computation, taught by Professor David Yun during Spring 2004, which gave him a comprehensive introduction to HPCS software development using the Engineering Department's LIPS-240 cluster as well as experience with MPI programming in C and Fortran. As part of this class, Michael began implementation of a Purpose-Based Benchmark (PBB) called the "Truss Optimizer". Briefly, given a wall with three attachment points and a load at a distance from the wall, the Truss Optimizer finds the pin-connected structure that uses the least amount of steel and therefore has the least mass. Each PBB, such as the Truss Optimizer, is designed to be complex enough to exercise HPCS resources, yet simple enough that it can be implemented on multiple platforms for comparative purposes. Unlike traditional HPC benchmarks like Linpack, PBBs measure not only run-time performance but also development time measures such as the time required to implement the software.

During the 2004-2005 academic year, the proposed funding will support the following research activities:

- Implementation of the Truss Optimizer PBB on an MHPCC platform such as HuiNalu using C, Fortran, and MPI libraries.
- Acquisition of PBB measurements for HuiNalu, including traditional HPC execution time metrics like speedup, as well as traditional software engineering metrics like developer effort, defect density, code size, and code complexity.
- Extension of my research system, the Hackystat automated software measurement framework [5], to support HPCS measurement.
- Possible porting/reimplementation of the Truss Optimizer PBB on a different MHPCC platform (such as Squall) or to a different software language/distributed computing infrastructure (such as JavaParty).

This proposed research is designed to make technical contributions to our understanding of the software engineering of high performance computing systems. It will result in the first complete implementation of the Truss Optimizer PBB, which will provide valuable insight into the utility and effectiveness of PBBs in general, the Truss PBB in particular, and the kinds of measures that can and should be collected. The research is also designed to create new opportunities for UH and MHPCC. It will provide new visibility for MHPCC as the site in which these measures were created. The data collected should be helpful in improving the developer productivity of MHPCC users. Finally, the research could lead to new collaborations between UH, MHPCC, SUN Microsystems, and the DARPA High Productivity Computing Systems program.

References

[1] DARPA High Productivity Computing Systems Program, http://www.highproductivity.org/

[2] The Roadmap for the Revitalization of High-End Computing, http://www.cra.org/Activities/workshops/nitrd/

[3] S. Faulk, J. Gustafson, P. Johnson, A. Porter, W. Tichy, and L. Votta, *Measuring HPCS Productivity*, International Journal of High Performance Computing and Applications, vol. 18, no. 4, Winter 2004.

[4] International Workshop on Software Engineering for High Performance Computing System Applications, http://csdl.ics.hawaii.edu/se-hpcs/

[5] Hackystat, A Framework for Automated Software Engineering Measurement, http://hackydev.ics.hawaii.edu