

PERC Highlights

10 June 2004

Performance Modeling

1. *An improved performance modeling framework has predicted performance within 15% on two different sets of scientific application codes.* A performance model is an encapsulation of the performance characteristics of a given scientific application on a given computer system, which then can be used to predict the performance of the code on a new system. An accurate and easy-to-use performance modeling capability has many potential applications, including more effective system design, streamlined system procurements, and even easier tuning of programs by applications scientists. However, until recently accurate performance modeling has required highly expert, in-depth analysis of both the computer program and the computer system.

We have been developing a methodology for performance modeling that significantly simplifies this process, replacing in-depth analysis by automated tools. We have previously reported that our MetaSim performance modeling framework produced “blind” performance predictions to within 15% accuracy on two DOE Office of Science codes: an unstructured grid code from the TOPS program and the POP ocean modeling application. Recently we applied a new, faster and more accurate MetaSim framework to a completely different set of application codes, namely some DoD HPCMO mission-partner codes, and also achieved “blind” performance predictions to within 15%. In particular, the predictions for HYCOM and Cobalt60 were performed at various problem sizes for processor counts ranging from 64 to 256 on three different architectures. This work will appear in a forthcoming paper accepted for UGC. [Lead institution: SDSC].

2. *A performance convolver used in performance modeling has been improved, resulting in more accurate performance modeling predictions.* The MetaSim Convolver is one component of the MetaSim performance modeling framework. It maps the memory access pattern of an application, which is obtained by the MetaSim Tracer tool, to a particular computer system’s memory performance characteristics, which are measured by the Memory Access Pattern Signature (MAPS) benchmark. Previously, the MetaSim Convolver used only two memory performance curves that captured upper and lower performance bounds. We recently improved these upper or lower bounds, and also produced a family of parametric curves that span the space between them. Initial comparison of predictions made with the old and new methods show improvements in accuracy for two applications on three machines: HP SC45, IBM Power3, and IBM Power4. This work will appear in a forthcoming paper submitted to SC2004. [Lead institution: SDSC]

3. *Two tracing tools, one for performance modeling and the other performance simulation, have been substantially accelerated using time and space sampling techniques, resulting in up to a 100-fold reduction in resources needed to gather trace data.*

At the start of the PERC project, when the MetaSim Tracer (which is used to sample a running application to obtain performance data) was first built, the slowdown due to tracing a large application program was over 1000-fold. Improvements to the MetaSim Tracer allow the user to

specify the interval at which to sample the address stream of an application. This and some other sampling techniques have reduced the slowdown to roughly 40-fold without significant reductions in tracing accuracy. [Lead institution: SDSC]

In a related activity, by using the SIGMA tracing library with the Dyninst instrumentation infrastructure, we have been able to accurately predict the performance of memory intensive programs, yet reduce the instrumentation overhead by a factor of 100. The key to this technique is a new approach to trace sampling that gathers detailed memory access information for only a small fraction of the time steps in a program. Using this technique produces results similar in accuracy to using a full memory trace of the program. [Lead institution: UMD]

4. *We have developed metrics of memory access regularity and a tool set for measuring the metrics in high performance applications.* In a SC2003 paper, one of six best paper nominees at the conference, we demonstrated that high-level memory can models provide useful guidance to significantly improve memory performance, even with applications that already are running at fairly high performance rates. In particular, we defined a metric of spatial reference regularity (regularity in the patterns of memory access), and then used the results of measurements with this metric as a guide to tuning applications for improved performance. Reductions in cache misses and memory stalls ranging from 5% to 95% were achieved on various test codes. [Lead institution: LLNL]

5. *We have developed a tool that helps a programmer identify the parts of a scientific program that have a performance gap.* Some scientific application codes perform well, and others perform poorly, when measured as a fraction of the theoretical peak performance rate of a given computer system. But the theoretical peak speed is not necessarily a good reflection of the performance potential of a particular code—a given application code may already be saturating some system resource, with little headroom for improvement. We have developed a tool that helps a scientific programmer identify parts of a program that have a performance gap, meaning that there is potential for improvement using the right tuning tools. The tool provides a quantitative estimate of the imbalance between memory demands and computations performed in those portions of a program. By comparing these numbers for different algorithms and/or different computer hardware, the programmer can select a combination that has the best price-performance balance. [Lead institution: ANL]

Performance Measurement and Analysis Tools

We completed and demonstrated at SC2003 a major new version of the SvPablo toolkit. This version, which includes a scalability analysis module, is installed on major sites across the country and has been used to characterize performance of various scientific application codes. SvPablo is a tool that integrates a number of performance analysis features into a graphical user interface tied to the user's source code. A recent version, which was officially released and demonstrated at SC2003, has been used to characterize performance of various scientific application codes, including TPM, a Tree-Particle-Mesh program for cosmological simulations, and MILC, a quantum chromo-dynamics application. The latest version of SvPablo version is now installed and accessible on major computational facilities across the country, including the IA-64 cluster at NCSA, the IBM-SP system at NERSC and the IA-32 cluster at FermiLab. One of the most useful features in this latest SvPablo version is the scalability analysis module. This

terpstra 6/9/04 6:19 PM

Deleted: 6

dhbailey 6/10/04 7:35 AM

Formatted: Space After: 0 pt, Don't adjust space between Latin and Asian text, Don't adjust space between Asian text and

dhbailey 6/10/04 7:35 AM

Formatted: Font:Times New Roman, 12 pt, Italic

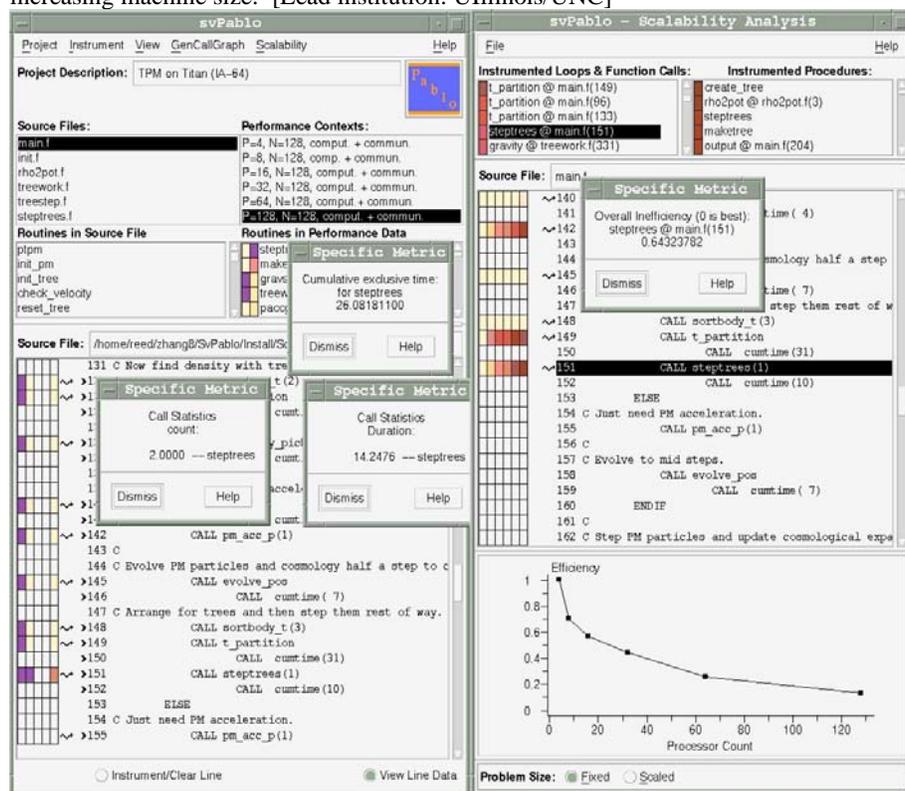
dhbailey 6/10/04 7:34 AM

Formatted: Font:Times New Roman, 12

dhbailey 6/10/04 7:34 AM

Deleted: We have completed a major new version of the SvPablo toolkit.

module combines performance data from multiple executions to provide a display, in graphical form, with information on how the various code sections scale as the machine size grows. As an example, Figure 1 depicts such information for a particular fragment of the TPM application, for executions using from 4 to 128 Itanium processors. With this kind of information, users can browse the various code parts, and quickly locate sections that present unacceptable scaling with increasing machine size. [Lead institution: UIllinois/UNC]



Two beta releases of PAPI 3.0, a major rewrite of the Performance Applications Programming Interface (PAPI) library, have been made available and a third will be released shortly. PAPI is a uniform software interface to the hardware performance counters present on many present-day computer processors. These counters provide access to very useful performance data, such as the number of floating-point operations performed, the number of cache misses at various levels of the cache hierarchy, and other information. PAPI 3 provides a streamlined interface with significantly reduced overheads. It supports multiple simultaneous event overflows and statistical profiling, and enhanced support of performance events native to specific architectures. PAPI 3 now supports most computer architectures of significance to the DOE Office of Science, including IBM POWER 3 and 4; Itanium 1 and 2; Pentium III and 4; and Opteron. We are porting PAPI to additional architectures, including the IBM BlueGene/L and the Cray T3E and X1. We are also encouraging the Linux kernel developers to incorporate performance counter

terpstra 6/9/04 6:20 PM
Deleted: 7

access directly into the kernel, which would more readily provide this functionality for the emerging Linux cluster community. [Lead institution: UTK]

SciDAC Application Impacts

Working together with scientists of other SciDAC projects, we have significantly improved the performance of several SciDAC application codes:

The performance of the astrophysics codes AGILE-BOLTZTRAN and ZEUS-MP were improved on IBM systems by 55% and 90%, respectively. The AGILE-BOLTZTRAN code improvement came from quantifying the impact of, and then eliminating, redundant computations. [Lead institution: ORNL]

The performance of the POP ocean modeling code was improved by 300% on the Cray X1 by optimizing communication algorithms and by quantifying operating system performance problems that were subsequently corrected. See separate PDF graphic file. [Lead institution: ORNL]

The performance of the CAM atmospheric model was improved by 30% on the IBM systems by quantifying load imbalance and developing new load balancing schemes optimized for clusters of shared memory systems. See separate PDF graphic file. [Lead institution: ORNL]

The performance of the fusion code GS2 was improved by 300% on IBM systems by changing the data decomposition. This optimization was discovered using the Active Harmony runtime optimization tool. See also item 11 below. [Lead institution: UMD]

Performance Optimization Tools

9. We implemented a wide range of source-to-source loop optimizations (loop fusion, fission, tiling, unrolling) in the ROSE source-to-source translator infrastructure. One approach to improving the performance of large scientific codes is to transform certain loops, at the source code level, to loops that can be compiled to run at higher rates. In a tool that we have developed, we have pursued four loop transformation techniques: loop fusion (combination of two loops), loop fission (separation of certain loops into two or more independent lops), tiling (rearrangement of loops to facilitate improved streaming of data from memory) and loop unrolling (expanding a loop so each new iteration contains several old iterations). Tests with our tool have demonstrated that these optimizations (especially loop fusion) yield a two- to five-times performance improvement on multigrid kernels written in C. Even greater performance improvements are achieved with high-level, object-oriented kernels written in C++ that facilitate application development. [Lead institution: LLNL]

10. Using Active Harmony techniques, we changed the default layout configuration within the GS2 program. Active Harmony is a software architecture that supports distributed execution of computational objects with dynamic adaptation of resources. We recently applied these tools to GS2, a SciDAC application code that studies low-frequency turbulence in magnetized plasma. When we run on 128 processors on NERSC SP-2, the program performance (average execution time per processor per time step) was reduced from 55 seconds to 16 seconds (without collision mode) and from 71 seconds to 32 seconds (with collision mode). This is about 3.4 times faster

terpstra 6/9/04 6:23 PM

Deleted: _____

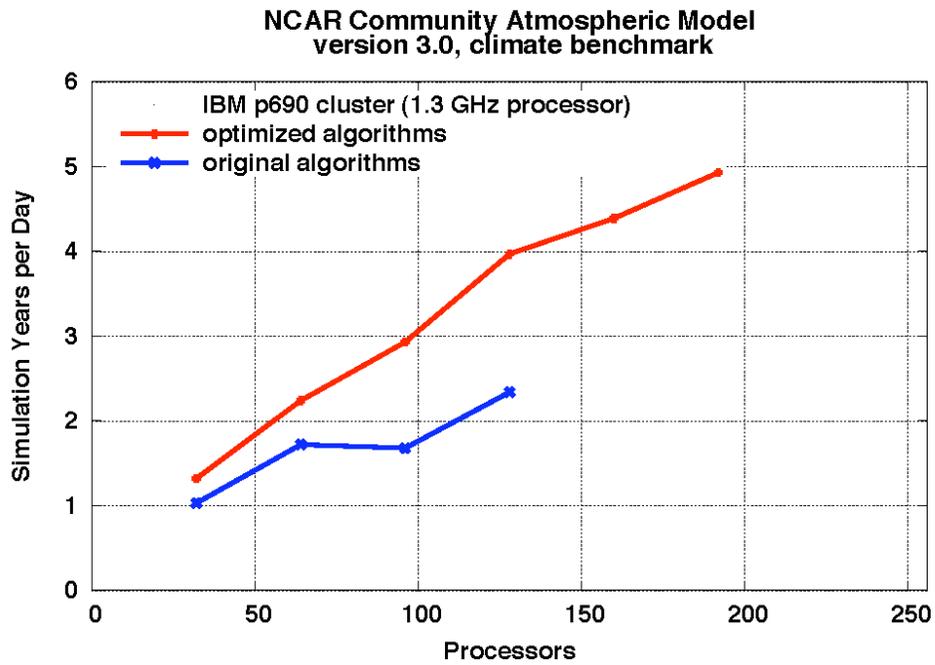
terpstra 6/9/04 6:20 PM

Deleted: 8

terpstra 6/9/04 6:21 PM

Deleted: _____

without collision mode and 2.3 times faster with collision mode. The resulting program is typically run for thousands of time steps when run in production mode. [Lead institution: UMD]



LANL Parallel Ocean Program
POP 1.4.3, climate benchmark

