SciDAC PERC-2 Final Report

September 2004 – September 2006
University of North Carolina at Chapel Hill (UNC)

During two years of SciDAC PERC-2, our activities had centered largely on development of new performance analysis techniques to enable efficient use on systems containing thousands or tens of thousands of processors. In addition, we continued our application engagement efforts and utilized our tools to study the performance of various SciDAC applications on a variety of HPC platforms.

1. Performance tool research and development

Stratified Population Sampling

On systems containing tens of thousands of processors, task-level performance monitoring can be invasive and adversely affect the performance of the applications being measured. Moreover, they can produce prodigious amounts of performance data. To retain the benefits of detailed measurement while reducing data volumes, we developed an adaptive performance monitoring system using stratified population sampling techniques. This prototype (a) reduces the number of nodes sampled based on the balance of the accuracy and sampling size depending on the variability of the data being monitored and (b) supports the division of node populations into separate classes, because large scale applications often have task equivalence classes (e.g., task zero and all other tasks). Monitoring these subclasses independently enables us to focus on the most relevant performance data.

We have developed the Adaptive Monitoring and Profiling Library (AMPL), a general-purpose toolkit that reduces collected data using clustered population sampling techniques. It also provides facilities for stratification of clusters into groups of monitored nodes, which can further reduce monitoring overhead if the variance of monitored metrics in these groups is low.

Using AMPL, we conducted experiments with sPPM code on a Xeon Linux cluster of 256 nodes at National Center for Supercomputing Applications (NCSA) to collect common PAPI hardware performance counter data. The results show that on this cluster, only 5% to 14% of the total number of nodes need to be monitored to guarantee 90% confidence and 8% error in measuring the mean value.

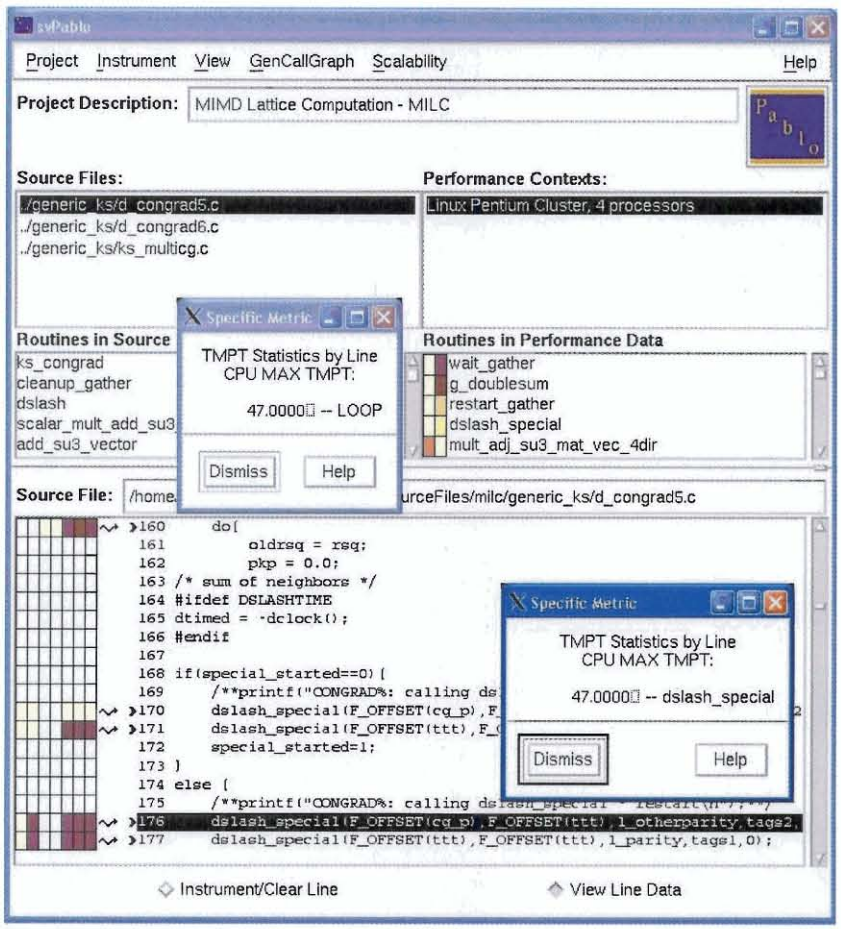
Reliability and Power Management

As the processor count in large systems grows to tens of thousands, system reliability and continued operation in the face of component failures is an increasing challenge. With partial funding from the Los Alamos Computer Science Institute (LACSI), we developed a system health monitoring library and a Health Application Programming Interface (HAPI). HAPI acquires fault indicator data via (a) vendor provided Self-Monitoring Analysis and Reporting Technology (SMART) for disk warnings such as seek and

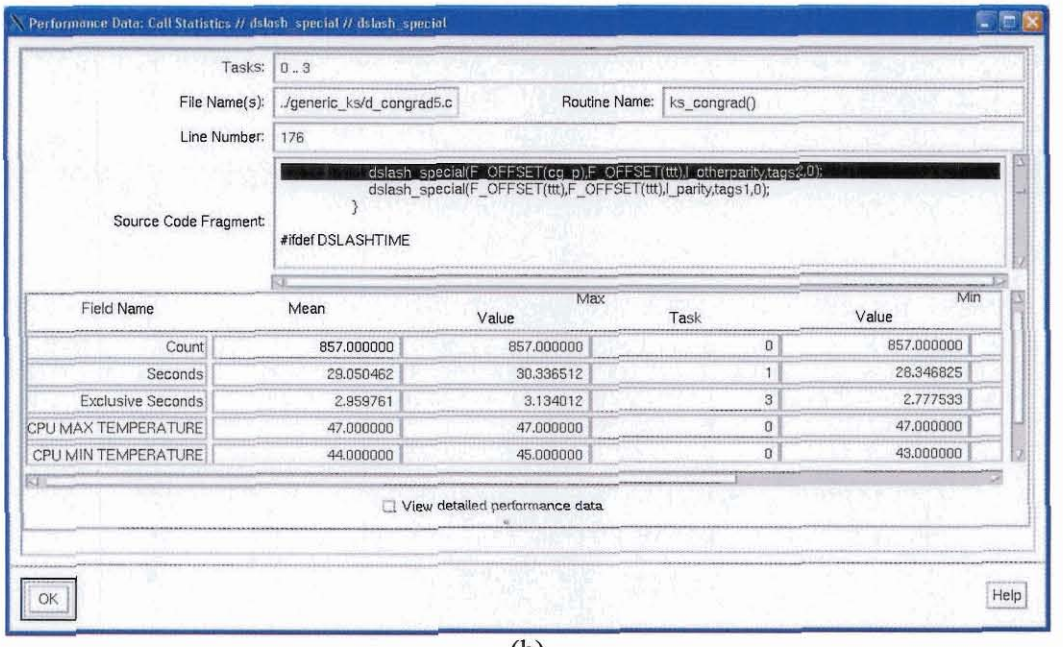
read/write retries; (b) Advanced Configuration and Power Interface (ACPI) for temperatures and CPU throttle rates and (c) low-level hardware sensors for motherboard temperatures and power supply voltages.

With PERC and other funding, we have integrated the HAPI library into our SvPablo performance analysis toolkit. The goal of this integration is to enable temperature and power consumption profiles of application execution, key indicators of potential failure modes and indicators of code sites for hybrid power-performance optimization. The temperature data is displayed associating with application source code in SvPablo GUI, as shown in the following figures. This data is from the MILC code, a part of the SciDAC Quantum Chromodynamics (QCD) project.

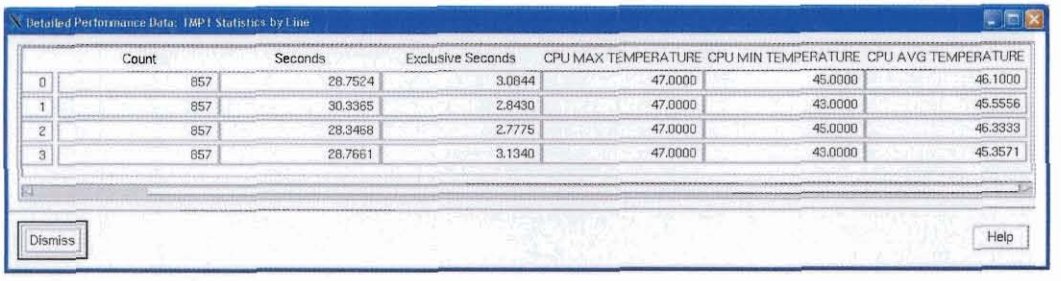
In Figure 1 (a), the CPU temperature data was captured via the HAPI library, now integrated with SvPablo data capture library, and is displayed along with other performance data by the source code. Statistical summaries for the CPU temperature over the processors (Figure 1 (b)) and detailed temperature readings for each processor (Figure 1 (c)) are also displayed for the instrumented constructs. By browsing the temperature data in SvPablo, users can easily detect the "hot spots" and identify the code sections for power-performance optimization.



(a)



(b)



(c)

Figure 1: SvPablo display of temperature measurement for MILC

SvPablo tool extension

Besides HAPI integration, we also extended SvPablo to support application scalability analysis. This extension 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 2 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.

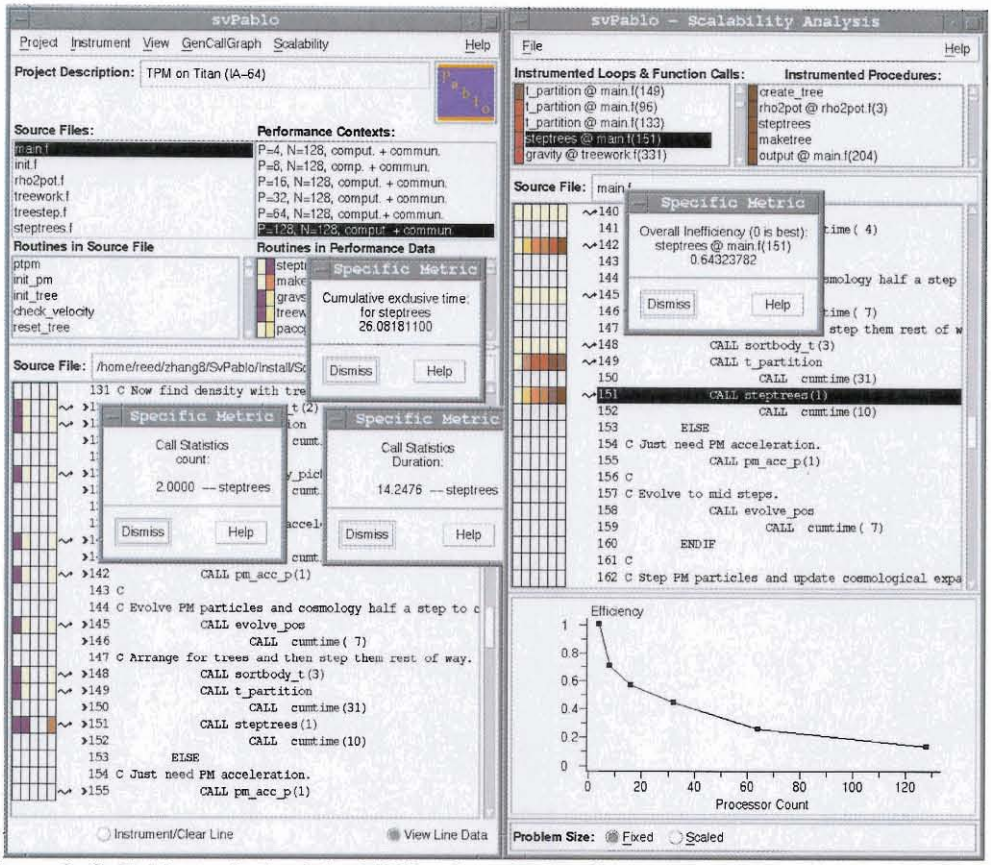


Figure 2: SvPablo analysis of the TPM code, with the Scalability GUI on the right half side

2. Application engagement performance studies

MILC (MIMD Lattice Computation):
During PERC-2, we collaborated closely with the U.S. Lattice Gauge Theory Community via a SciDAC program for a National Computational Infrastructure for Lattice Gauge Theory. Based on a MILC performance model created by MILC developers, we conducted performance analyses communicate-bound and compute-bound cases on Pentium-3 Linux clusters. We used SvPablo to collect detailed performance data on different aspects of the code on a variety of HPC platforms including QCDOC, various Linux clusters, and Cray X1 and IBM BlueGene/L systems, and identified the key fragments that affect the code performance. The performance study results have been presented at SciDAC QCD meetings and have helped the QCD community to understand MILC's performance behavior on different execution platforms and configurations.

In addition, we developed tools to help the QMP (Message-passing API for QCD) developers validate the QMP implementation on various QCD platforms and to study the communication patterns in MILC code.

CAM (Community Atmosphere Model):
Using SvPablo, we conducted performance analysis for CAM code with emphasis on load balancing and scalability. We instrumented critical sections in CAM code and did a

series of executions of the instrumented code on seaborg – a IBM SP system at NERSC. For the scalability, the code scaled well up to 64 MPI tasks for the specific data set – T85, and then it deteriorates on 128 MPI tasks. For load balancing, we experimented with four algorithms. Our results indicated that the algorithm which allows all-to-all communications had the best load balance on seaborg, although expensive.

GYRO (a fusion code):
We also participated in PERC-2 application engagement group efforts on performance evaluation for GYRO code. We conducted performance analysis for GYRO on IBM P3 and p690 systems, SGI Altix, and Cray X1, and examined its scaling behavior as well as the impact of task and memory affinity.

3. Publications

Daniel A. Reed, Celso Mendes, "Intelligent Adaptation in Grid Applications," *Proceedings of the IEEE,* Vol. 93, No.2, 2005

Daniel A. Reed, C. Lu and C .L. Mendes. "Reliability Challenges in Large Systems," *Future Generation Computer Systems*, Spring 2005.

Charng-da Lu, Daniel Reed, "Assessing Fault Sensitivity in MPI Applications," *SC2004 Best Technical Paper, Proceedings of Supercomputing 2004,* Pittsburgh, PA, November, 2004.

Karthik Pattabiraman. "Design and Evaluation of a Power-Aware Parallel I/O System," *Master's Thesis in Computer Science*, University of Illinois at Urbana-Champaign, 2004.