Performance Engineering Research Institute
SciDAC-2 Enabling Technologies Institute
Final Report

20 April 2013

I. Introduction

This document is the final report for cooperative agreement DE-FC02-06ER25765, the Performance Engineering Research Institute (PERI), an Enabling Technologies Institute of the Scientific Discovery through Advanced Computing (SciDAC-2) program of the Department of Energy's Office of Science (DOE SC) Advanced Scientific Computing Research (ASCR) program.

Enhancing the performance of SciDAC applications on petascale systems had high priority within DOE SC at the start of the second phase of the SciDAC program, SciDAC-2, and it continues to do so today. Achieving expected levels of performance on high-end computing (HEC) systems is growing ever more challenging due to enormous scale, increasing architectural complexity, and increasing application complexity. To address these challenges, PERI implemented a unified, tripartite research plan encompassing: (1) performance modeling and prediction; (2) automatic performance tuning; and (3) performance engineering of high profile applications. The PERI performance modeling and prediction activity developed and refined performance models, significantly reducing the cost of collecting the data upon which the models are based, and increasing model fidelity, speed and generality. PERI's primary research activity was automatic tuning *(autotuning)* of scientific software. This activity was spurred by the strong user preference for automatic tools and was based on previous successful activities such as ATLAS, which automatically tuned components of the LAPACK linear algebra library, and other recent work on autotuning domain-specific libraries. Our third major component was application engagement, to which we devoted approximately 30% of our effort to work directly with SciDAC-2 applications. This last activity not only helped DOE scientists meet their near-term performance goals, but also helped keep PERI research focused on the real challenges facing DOE computational scientists as they entered the petascale era.

The University of Southern California's Information Sciences Institute organized PERI and managed its progress. Over the five-year course of SciDAC-2, the original PERI consortium of ten institutions was augmented with additional ones, either via subcontracts or because the researchers changed institutions, and their funding followed. This report is a summary of the overall results of the USC PERI effort. We direct interested readers to the final reports of each PERI institution for additional details.

II. Application Performance Modeling

During the course of SciDAC-2, and under the direction of USC, PERI researchers established a performance modeling methodology to predict application run-times accurately and with error bounds as the application's input parameters are varied. In addition, we demonstrated techniques that provide significant cost savings for computer architecture sensitivity studies, which vary machine parameters, such as cache sizes or prefetch and branch prediction strategies. PERI researchers extended this work to enhance scalability, flexibility and the level of automation needed for

exascale extrapolation. Allan Snavely of UCSD was the leader of the PERI application performance modeling activity, and USC did not participate directly in this research.

III. Automatic Performance Tuning

In the area of automatic performance tuning, PERI researchers developed whole program analyses that facilitate optimizations spanning multiple kernels, files, and procedure boundaries. This work utilized autotuning techniques on significantly larger scales than individual kernels, and was targeted at full-scale SciDAC applications. The principal research focus involved scaling and generalizing the techniques developed for kernel-level optimization and leveraging work elsewhere within PERI to identify opportunities for improvement and then generate viable search strategies. As an outgrowth of collaborations with SciDAC application teams, we defined additional analysis and transformation support required to meet the needs of SciDAC application developers. Architectural targets of these transformations included SIMD compute engines such as SSE-3, prefetch into cache, and managing multiple cores. In addition, PERI personnel developed transformations for specific application classes such as stencil computations, and specialization for known problem sizes. A number of transformation mechanisms were explored to simplify the development of application-specific transformations.

Kathy Yelick from the Lawrence Berkeley National Laboratory (LBNL) initially directed this aspect of PERI's research. When she was assigned to be the director of the National Energy Research Scientific Computing Center (NERSC), USC's Mary Hall took over. When Mary moved to the University of Utah, to accept a faculty position in Computer Science, USC subcontracted to Utah so that Mary could remain in the leader of the autotuning effort.

Autotuning Tool Integration

Within PERI, several different research groups were developing autotuning *tools* to address the challenges of automatic performance tuning. These projects, many of which had benefitted from years of DOE investment, had complementary strengths and could, therefore, be brought together to develop an integrated autotuning *system*. Towards that end, PERI researchers worked to develop a common framework to allow autotuning tools to share information and facilitate composition into the most appropriate set of tools for a particular application. Through common application programming interfaces (APIs), they created an autotuning system that brings together the best capabilities of each of these tools.

PERI researchers focused their development of interfaces on two portions of the autotuning process. Any compiler-based approach will apply code transformations to rewrite application code from its original form to a form that more effectively exploits architectural features such as registers, caches, SIMD compute engines, and multiple cores. Commonly used code transformations include loop unrolling, blocking for cache, and software pipelining. Thus, researchers designed a *transformation API* that is input to the Transformation box in Figure 1. This API provides a *transformation recipe* that described how to transform original source into an optimized source representation.

By accepting a common transformation recipe, the PERI autotuning system permits code transformation strategies derived by PERI compilers and tools (or users) to be implemented using any transformation and code generation tool, such as CHiLL (USC/ISI, Utah and Argonne), the ROSE LoopProcessor (LLNL), and POET (UTSA). The API supports the specification of unbound transformation parameters that are then tuned using search algorithms. The initial API includes a naming convention for specifying language constructs in source code and code transformations available in CHiLL and ROSE/POET. PERI researchers published two papers describing the transformation recipes [Hall2009]. The first paper, a collaboration between University of Utah, USC/ISI and Argonne researchers, described an API that attempts to generalize con-

cepts derived from the PERI team into a common recipe that can be the input/output of different PERI tools. The second paper, produced in collaboration between University of Utah and USC/ISI, raised the level of abstraction for this framework and offered a programming language interface for developing libraries of optimization strategies [Rudy2010]. Such optimization libraries can be programmed by compiler experts, and then made available to application developers, thus providing a high-level interface to the compiler code generation and transformation and the autotuning framework.

A *search API* was also created to provide input into the empirical optimization process by running experiments on actual hardware to determine the best optimized implementation. The search API allows the autotuning tools to exchange information about their available tuning options and constraints on the search space, and to plug-in different search algorithms. The common framework supports both autotuning using training runs (and re-compilation) along with continuous optimization during production runs. The search API, led by the UMD team, permits autotuning systems to exchange information about their available tuning options, constraints on the search space, and to plug-in different search algorithms. The common framework supports both automatic tuning using training runs along with continuous optimization during production runs.

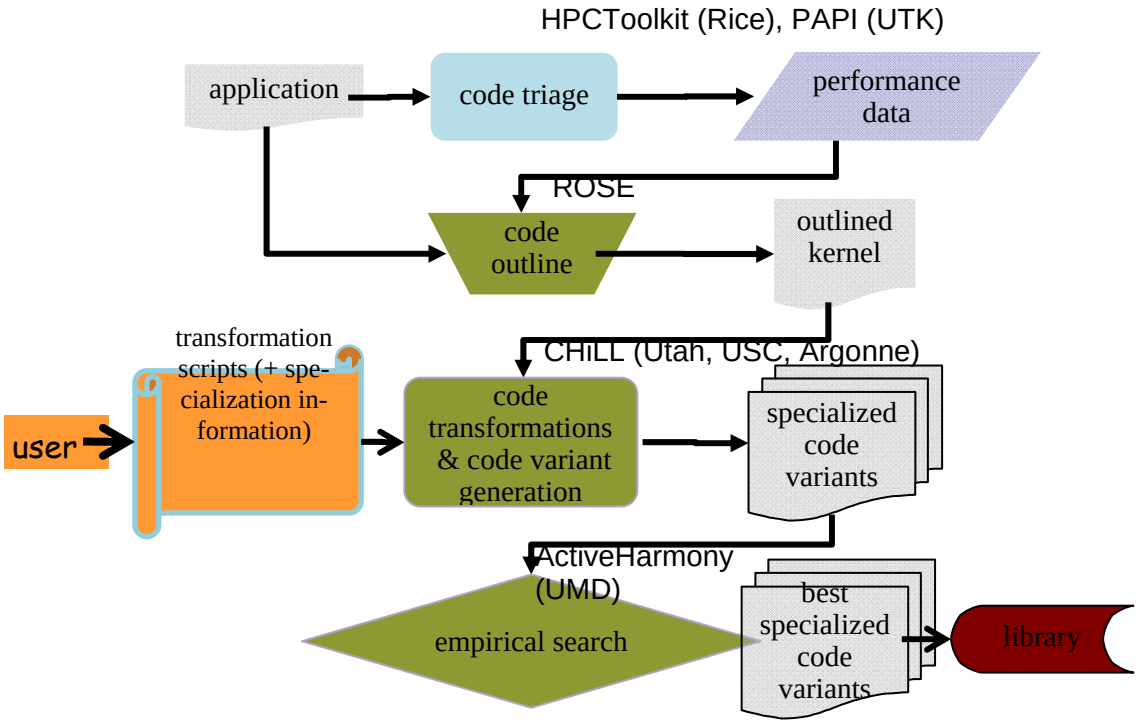

Figure 1: PERI automatic tuning workflow.

As stated above, PERI work in autotuning focused on the integration of several PERI tools. An initial integration of UMD's Active Harmony system and the CHiLL transformation framework developed and supported by researchers at USC/ISI, Utah and ANL provided experience in how to integrate these separate tools effectively into an autotuning system. Through the APIs and substantial coordination activities, PERI researchers then focused on more extensive integration of end-to-end tools applied to complete applications. In 2009, PERI researchers first applied this integrated set of tools to SMG2000. In 2010, they applied these tools to PFLOTRAN as part of the Tiger Team activity. Most of the remainder of this section describes in greater detail the work on SMG2000 and PFLOTRAN.

Autotuning of SMG2000

The primary focus of PERI's research goal was autotuning of a representative DOE application. As a group, we chose to focus on SMG2000, which computes a semi-coarsening multigrid on structured grids. This code, which is based on the TOPS-developed hypre library, is representative of a wide variety of SciDAC applications. As shown in Figure 2, we integrated tools that included the work of seven PERI institutions to perform autotuning of a key kernel of SMG2000. A *code triage* phase examines the original SMG2000 code to identify the key computations in SMG2000. We use HPCToolkit, which in turn uses PAPI, to collect performance monitoring information and map it back to the application code structures. HPCToolkit discovers that the key performance bottleneck in SMG2000 is a residual computation which implements a 2-dimensional 6-point stencil using a sparse matrix-vector multiplication. We then use the ROSE compiler to *outline* the residual computation into a standalone program. The outlined kernel is a very small piece of code with a much smaller execution time than the full application, is therefore better suited for autotuning experiments. We describe a strategy for transforming the outlined kernel to improve its performance, and provide this script to CHiLL for automatic code generation of a set of variants of the outlined kernel. CHiLL is invoked repeatedly by ActiveHarmony, using a parallel rank ordering algorithm to evaluate the set of variants generated by CHiLL. The parallel rank order search finds the best solution of 581M possible solutions in 20 steps, and yields a 2.37X speedup on residual computation and 27% performance improvement on the full application. Figure 2 below depicts the improvement of SMG20000 as the search process unfolds.

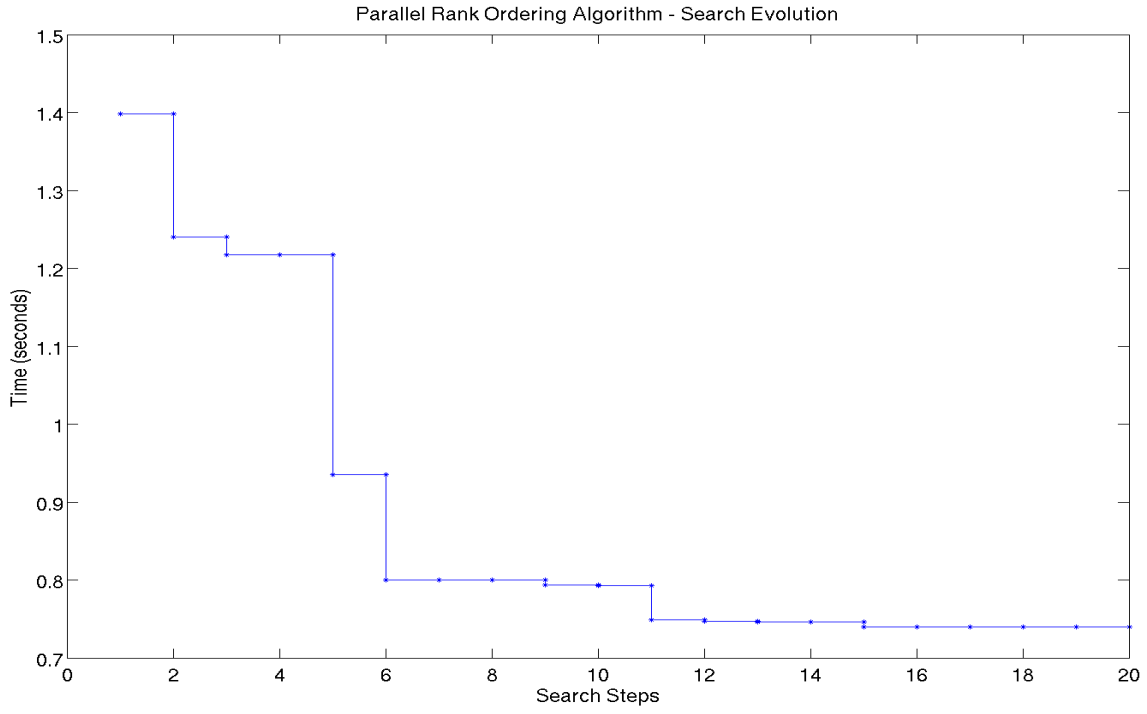

Figure 2: Improvement of SMG2000 runtime vs. depth in search

Autotuning of PFLOTRAN

PFLOTRAN is a DOE application developed at LANL that models multiscale-multiphase-multicomponent subsurface reactive flows. PFLOTRAN uses the PETSc library as the basis of

its parallel framework. As shown in Figure 1, PERI researchers integrated tools that included the work of several PERI institutions to perform autotuning of key computations in PFLOTRAN. A *code triage* phase examined the original PFLOTRAN code to identify the key computations. We then used HPCToolkit, which in turn uses PAPI, to collect performance monitoring information and map it back to the application code structures.

Using HPC Toolkit (Rice) and the Cray PAT tool, researchers at North Carolina State, Rice and ORNL identified three main computations in PFLOTRAN as candidates for optimization: a PETSc routine that computes a matrix-vector multiplication (MatMul_SeqBAIJ_N); a PETSc function that solves the system A x = b, given a factored matrix A, (MatSolve_SeqBAIJ_N); and a routine that calculates the contribution of aqueous equilibrium complexity to the residual and Jacobian functions for Newton-Raphson (RTOTAL). These two PETSc functions comprise 17% of execution time, and 6-7% of the computation is spent in a single loop nest computation of RTOTAL; all functions achieve only between 4 and 5% of peak on a node on Jaguar.

In a collaboration between USC/ISI, Utah and UMD, PERI researchers performed autotuning of MatSolve_SeqBAIJ_N, which performs a forward and a backward solve, and uses a blocked compressed row representation for matrix A. The block size N is a parameter, but instrumentation data shows that during production runs the block sizes are small numbers, up to 15. As previous work by the Utah, USC and ANL has shown [Shin2010], the performance of matrix computations with small matrix sizes is very sensitive to optimization parameters such as loop unroll factors and tile sizes. When combined with specialization, a compiler technique for generating highly optimized code for known problem sizes, we achieved better performance than most high performance libraries. The autotuning team used CHiLL to generate specialized code versions for a particular block size, and ActiveHarmony to search for the best performing version. Our experiment used four different compilers available on Jaguar, and evaluated performance as a function of unroll factors for the inner loops. PERI researchers compared results using Active Harmony and an exhaustive search of a search space consisting of more than 1100 points. The results are shown in Table 1. The original performance varies significantly by compiler, but the best speedup and best overall performance was obtained with the Pathscale compiler, demonstrating a 1.8x performance gain with Active Harmony and a 1.9x performance gain using exhaustive search. UMD and Utah worked on replacing this code in the full PFLOTRAN application. USC/ISI is currently working on the performance optimization of the backward solve targeting the Cray XT5 at ORNL (Jaguar).

In a collaboration between Argonne, Utah and UMD, USC also performed autotuning for the PETSc routine MatMul_SeqBAIJ_N. Like the previous example, the code was specialized for block sizes of N=15 or a multiple of 15 (15x105,15x90,15x75, 15x60). Performance gains were up to 1.5X on a single node of Jaguar using the PGI compiler.

In addition to these two PETSc kernels, we also optimized the RTOTAL function from PFLOTRAN. Early pathfinding and hand optimization by ORNL researchers revealed a collection of optimizations that would improve the performance of the loop from this code in which the application spent 6% of its time. The loop consisted of inner loop nests with a variable *ncomp* used to establish the number of loop iterations. By analyzing instrumentation data, we discovered that the value of *ncomp* was usually very small, between 2 and 4. By specializing the code for specific values of ncomp and aggressively unrolling the inner loop nests, we can achieve speedups ranging from 1.32X to 1.52X using CHiLL, or up to 1.8X with additional manual optimizations to remove recurrences between accumulations. These results are in Table 2.

Compiler	Original	Active Harmony			Exhaustive		
	Time	Time	(u1,u2)	Speedup	Time	(u1,u2)	Speedup
pathscale	0.58	0.32	(3,11)	1.81	0.30	(3,15)	1.93
gnu	0.71	0.47	(5,13)	1.51	0.46	(5,7)	1.54
pgi	0.90	0.53	(5,3)	1.70	0.53	(5,3)	1.70
cray	1.13	0.70	(15,5)	1.61	0.69	(15,15)	1.63

Table 1. Autotuning results for forward solve from triangular solve library, using CHiLL and Active Harmony.

Ncomp	Original	Chill		Hand Tuned	
	Time	Time	Speedup	Time	Speedup
1	0.09	0.06	1.52	0.05	1.80
2	0.13	0.096	1.32	0.087	1.46
3	0.18	0.12	1.49	0.11	1.45
4	0.21	0.16	1.32	0.15	1.45

Table 2. Results of optimizing RTOTAL function in PFLOTRAN.

At SciDAC 2011, a poster and short paper reported on overall performance improvement for PFLOTRAN [Chame 2011]. After autotuning, each of these routines was sped up by nearly a factor of two. Additional I/O tuning performed by our PERI collaborators resulted in a final overall 40-fold speedup of the initialization phase, 4-fold improvement in the a write stage, and a 5-fold improvement of total simulation time on 90,000 cores.

Autotuning Technology Software Development

PERI researchers at Utah, USC/ISI and ANL developed CHiLL, a framework for composing high-level loop transformations designed to generate efficient code for complex loop nests. It supports an extensive set of loop transformations for perfect and imperfect loop nests, including tiling, permutation and unroll-and-jam, thus lifting the burden of generating multiple intermediate

steps from compilers or optimization tools. CHiLL uses an improved version of Omega (OmegaPlus) to manipulate integer arithmetic and relies on polyhedral scanning provided by Omega's code generator. Utah, USC/ISI and UMD researchers integrated the UMD Active Harmony system with the CHiLL framework. Users can customize the CHiLL optimizer via a process called recipes. In the final year of PERI, we performed experiments applying CHiLL to two SciDAC-e codes, MGDC and the QR factorization of. An integration of CHiLL with the ROSE compiler was undergoing extensive testing in preparation for release at the end of the PERI project. CHiLL and Omega Plus are available from http://www.cs.utah.edu/~chunchen/.

IV. Application Engagement

Throughout the course of SciDAC-2, PERI researchers worked closely with application scientists in DOE-funded computational science projects. PERI had two primary mechanisms for this application engagement: Tiger Teams and Application Liaisons. We proactively communicated with application groups through these activities and otherwise. PERI participants attended application team meetings to make presentations, hosted individuals or groups at our sites to demonstrate tools, and communicated directly with application developers to identify optimizations important to their codes.

Pat Worley of ORNL initially had overall management responsibilities for engagement and liaison activities. When other responsibilities led to his having to defer this role, Dr. Lucas reassigned it to Bronis de Supinski of LLNL, who had already been coordinating PERI's Tiger Teams.

Most of the computing resources for the PERI engagement activities on the ORNL and ANL Leadership Class systems were provided by the Performance Evaluation and Analysis Consortium End Station (PEAC) INCITE project, for which Pat Worley was the principal investigator. During the course of PERI, Pat Worley led two successful proposals for 3-year PEAC INSIGHT projects.

PERI application liaisons were established as long-term relationships with DOE computational science projects having clearly identified performance needs and a desire to work with us. These interactions are defined and maintained by individual PERI personnel, who are assigned to be the liaisons between the science application project and PERI. The initial assignments were motivated by the information collected in the PERI Application Survey. The nature and number of liaison interactions evolved over the course of PERI, based on updates to the application surveys and requests from science application personnel, from DOE computing centers, and from DOE headquarters.

The nature of the interaction between PERI and each science application project was unique to each, and varied over time. Some of the PERI liaisons were engaged actively, helping enhance the performance of their colleague's codes, bringing in other PERI resources and expertise as needed, and educating PERI researchers as to the important performance issues. Other interactions are more passive, with the liaison simply staying in touch, tracking performance needs, advising on performance issues, and looking for opportunities for PERI to contribute, but not directly engaged for the moment in tuning.

PERI reached out to every single SciDAC-2 Scientific Application Partnership (SAP), in an effort to establish these performance engineering collaborations. Many enthusiastically agreed, some politely declined if they felt they didn't have the resources to hold up their end of the bargain, and some frankly ignored us. Once potential collaborators had been identified, PERI liaisons were assigned in such a manner as to exploit familiarity and proximity. We felt that this would maximize the probability of success. Therefore, USC was assigned to be the liaison with the Hierarchic-

al Petascale Simulation Framework for Stress Corrosion Cracking SAP, led by USC's Prof. Priya Vashishta.

As part of the active liaison relationship with the Hierarchical Petascale Simulation Framework for Stress Corrosion Cracking application, USC/ISI and Utah worked with Aiichiro Nakano's group at USC to optimize a quantum mechanical code. Four key computational kernels were identified that exhibited opportunities for improved performance: two linear algebra kernels, a sparse-matrix kernel and a stencil computation. The two linear algebra kernels were replaced by linear algebra libraries, and the sparse-matrix kernel was optimized by hand, resulting on a 25% performance improvement. Through this initial hand-tuning, the performance of the application improved by 19.8% on an Intel Core2Duo. The analysis also indicated that this code is an excellent candidate for further optimization from applying a number of automatic performance tuning technologies.

USC/ISI and Utah then worked with Aiichiro Nakano's group to optimize the stencil kernel using PERI autotuning tools. The kernel is part of a seismic wave propagation simulation where a three-dimensional, 6^{th} order stencil computes spatial derivatives on uniform grids. The original computation had been parallelized at multiple levels. The three-dimensional data space is distributed among cluster nodes, with computation following the "owner's compute" rule, and node-to-node communication implemented in MPI. Each node has thread-level parallelism and each thread has SIMD parallelism. SIMD parallelism had been implemented manually using SSE intrinsics.

This original stencil computation suffered from high TLB misses (approximately 25%) and high second-level cache misses on each node, on a 512-node cluster with Quad-Core Intel Xeon dual-processor nodes running at 2.33GHz with 4MB L2 per die. To optimize performance on a single node, the ISI /Utah team used loop tiling to improve locality at the L2 cache and TLB, and loop unrolling and scalar replacement to improve locality at the register level. Loop unrolling also exposed SIMD parallelism to the native compiler, which successfully generated SSE code. CHiLL was used to automatically generate code variants of the stencil kernel with tile and unroll sizes as unbound parameters, and an empirical search determined the tile and unroll sizes that achieved best performance. These optimizations resulted in a 2.56 speedup for the stencil computation of a single thread on a single node.

In 2010, PERI was the recipient of three SciDAC-e awards. These were augmentations to PERI funded by the American Recovery and Reinvestment Act (ARRA) of 2009. USC led one of the three activities, a project entitled the Performance Engineering Research Institute SciDAC-e Augmentation: Performance Enhancement of Simulating the Dynamics of Photoexcitation for Solar Energy Conversion. The others were led by LBNL and UNC. The SciDAC-e projects are required to submit final reports of their own, and thus will not be the subject of further discussion herein.

VI. Institute Management and Outside Interactions

Institute Management

Robert Lucas of USC/ISI was the overall manager of the PERI institute, assisted by David H. Bailey of LBNL. This included monitoring project activity, conducting biweekly teleconferences, overseeing twice-yearly project meetings, preparing reports, and representing PERI in various community activities.

PERI was a widely distributed project, so we went to great lengths to stay connected and coordinated. We had teleconferences, a meeting each year at the SC conference, and two annual PERI team meetings. Teleconferences were held approximately every two weeks. The precise schedule varied depending on the needs of the project as well as the availability of the PIs. PERI team meetings were scheduled twice every year for the five-year duration of SciDAC-2. Nine of PERI's ten initial institutions, as well as Utah, took a turn hosting the meeting. The meeting schedule is depicted in Table 3, below. Meetings were also held every Monday at the annual SC conference.

September 26 – 27, 2006	Argonne National Laboratory
March 12 – 13, 2007	Lawrence Berkeley National Laboratory
September 19 – 21, 2007	University of North Carolina
February 24 – 26, 2008	University of California, San Diego
September 25 – 26, 2008	Oak Ridge National Laboratory
March 25 – 26, 2009	University of Southern California
September 28 – 29, 2009	University of Maryland
March 23 – 24, 2010	University of Tennessee, Knoxville
September 1 – 2, 2010	University of Utah
March 1 – 2, 2011	Rice University

Table 3. Dates and locations of the PERI bi-annual project meetings.

PERI was augmented with three SciDAC-e wards in FY2010. These projects are led by Lucas, Bailey, and Fowler. They include nine of the PERI institutions, as well as new subcontractors including Duke, Iowa, Oregon, and Sandia. These awards came very late in FY2010. As a result, DOE granted PERI a no-cost extension until December 14, 2012.

In 2010, Bailey took the lead in organizing and producing a new book entitled *Performance Tuning of Scientific Applications*. It is co-edited by Lucas and by Samuel Williams of LBNL, and includes 16 chapters, most of which were co-authored by PERI-funded personnel. See the section on "books" in the publication section below for additional details.

Dan Gunter of LBNL maintained and enhanced the PERI website and wiki. He also performed server administration to make sure that the wiki software, MediaWiki, had been kept up-to-date and secure. The website was organized to follow DOE guidelines, and kept a current list of liaison activities, mailing lists, and PERI publications. The wiki was used by PERI PIs, and Gunter organized the content to clearly reflect the then current activities of the PERI tiger team efforts

and to provide a "home page" for the PERI search, database, and transformations working groups.

Other External Interactions

In addition to creating formal liaison relationships with SciDAC application code teams and forming Tiger Teams, PERI researchers worked with a variety of other DOE-sponsored computational projects. For example, work on the University of Oregon's TAU Performance System received PERI funding in 2008, and was part of SciDAC-e. Other external performance collaborations included Portland State University, the University of Texas at San Antonio, the University of Wisconsin, Barcelona, and Julich Supercomputing Center. In addition, both directly and through the PEAC End Station, PERI interacted frequently with computer center staff at the ALCF, the NLCF, and NERSC, and with IBM and Cray. For example, PERI ported and maintained performance tools, such as PAPI and HPCToolkit, required by PERI research and engagement activities, and made these available to the centers' users. Finally, PERI researchers were active participants in workshops that target planning and preparing application teams for future peta- and exascale computer systems.

Computer System Access

The Performance Engineering and Analysis Consortium End Station (PEAC) INCITE project provided the PERI project with access to the Leadership Class Systems at ANL and at ORNL. Access to these systems was critical to achieving PERI's goals in both research and engagement. PEAC also provides access to the larger performance engineering and research community, and provides a mechanism for this larger community to contribute to the success of the DOE Leadership Computing Facilities (LCF) and DOE's goals in computational science at scale. Pat Worley led the effort to submit the PEAC proposals, requesting in 2010 40M CPU hours on the Cray XT systems at ORNL and 20M CPU hours on the IBM BG/P system at ANL. A separate but more modest allocation was also obtained by Bailey on the NERSC facility for PERI project usage.

Outreach to SciDAC-2 and the Performance Community

Although PERI included many of the researchers working in performance evaluation, there were other projects in that area, some funded by DOE SC. PERI's intent was to be inclusive. We therefore established collaborative relationships with several other performance evaluation projects. These collaborations included access to our benchmark applications as well as substantive relationships. For example, we are collaborated closely with the TASCS CET and CSCAPES Institutes.

Training

During the course of PERI, Mary Hall directed students working on performance engineering research at both USC and Utah. In the latter case, USC subcontracted for her services from the University of Utah. Mary also had a post doctoral scholar.

Chun Chen, CS PhD at USC, then post doc at both USC and Utah

Protonu Basu, CS PhD at Utah

Malik Khan, CS PhD at USC

Shreyas Ramalingam, CS PhD at Utah

Bibliography

Books:

David H. Bailey, Robert F. Lucas and Samuel W. Williams, ed., *Performance Tuning of Scientific Applications*, Taylor and Francis, New York, manuscript now complete; to be released in late 2010.

Chapters 1, 2, 3, 4, 7,10, 11, 12, 14, 15, 16 were directly authored or co-authored by PERI-funded researchers; all chapters were at least edited by PERI-funded personnel:

1. "Introduction" by David H. Bailey
2. "Parallel Computer Architecture, by Samuel W. Williams and David H. Bailey
3. "Software Interfaces to Hardware Counters," by Shirley V. Moore, Daniel K. Terpstra, and Vincent M. Weaver.
4. "Measurement and Analysis of Parallel Program Performance using TAU and HPCToolkit," by Allen D. Malony, John Mellor-Crummey, and Sameer S. Shende.
5. "Trace-Based Tools," by Jesus Labarta (Labarta is at the University of Barcelona in Spain, but has collaborated with PERI in application analysis).
6. "Large-Scale Numerical Simulations on High-End Computational Platforms," Leonid Oliker, Jonathan Carter, Vincent Beckner, John Bell, Harvey Wasserman, Mark Adams, Stephane Ethier, and Erik Schnetter.
7. "Performance Modeling: The Convolution Approach," by David H Bailey, Allan Snavely, and Laura Carrington.
8. "Analytic Modeling for Memory Access Patterns Based on Apex-MAP," by Erich Strohmaier, Hongzhang Shan, and Khaled Ibrahim.
9. "The Roofline Model," by Samuel W. Williams.
10. "End-to-end Autotuning with Active Harmony," by Jeffrey K. Hollingsworth and Ananta Tiwari.
11. "Languages and Compilers for Autotuning," by Mary Hall and Jacqueline Chame.
12. "Empirical Performance Tuning of Dense Linear Algebra Software," by Jack Dongarra and Shirley Moore.
13. "Autotuning Memory-Intensive Kernels for Multicore." By Samuel W. Williams, Kaushik Datta, Leonid Oliker, Jonathan Carter, John Shalf, and Katherine Yelick.
14. "Flexible Tools Supporting a Scalable First-Principles MD Code," Bronis R. de Supinski, Martin Schulz, and Erik W. Draeger.

15. "The Community Climate System Model," by Patrick H. Worley.

16. "Tuning an Electronic Structure Code," by David H Bailey, Lin-Wang Wang, Hongzhang Shan, Zhengji Zhao, Juan Meza, Erich Strohmaier, and Byounghak Lee.

Technical Articles:

[Bailey2007] David H. Bailey, Robert Lucas, Paul Hovland, Boyana Norris, Kathy Yelick, Dan Gunter, Bronis de Supinski, Dan Quinlan, Pat Worley, Jeff Vetter, Phil Roth, John Mellor-Crummey, Allan Snavely, Jeff Hollingsworth, Dan Reed, Rob Fowler, Ying Zhang, Mary Hall, Jacqueline Chame, Jack Dongarra, Shirley Moore, "Performance Engineering: Understanding and Improving the Performance of Large-Scale Codes," *CT Watch Quarterly*, vol. 3, no. 4, pg. 18–23, November 2007.

[Chen2007] C. Chen, J. Shin, S. Kintali, J. Chame and M. Hall, "Model-Guided Empirical Optimization for Multimedia Extension Architectures: A Case Study," *Proceedings of the Workshop on Performance Optimization of High-Level Languages*, held in conjunction with IPDPS'07, March 2007.

[Nelson2008] Y. Nelson, B. Bansal, M. Hall, A. Nakano, K. Lerman, "Model-Guided Performance Tuning of Parameter Values: A Case Study with Molecular Dynamics Visualization," *Proceedings of the Workshop on High-Level Parallel Programming Models and Supportive Environments*, held in conjunction with IPDPS '08, April 2008.

[Hall2008] M. Hall, Y. Gil, R. Lucas, "Self-Configuring Applications for Heterogeneous Systems: Program Composition and Optimization Using Cognitive Techniques," *Proceedings of the IEEE, Special Issue on Cutting-Edge Computing*, vol. 96(5), May 2008.

[Bailey2008] David Bailey, Jacqueline Chame, Chun Chen, Jack Dongarra, Mary Hall, Jeffrey K. Hollingsworth, Paul Hovland, Shirley Moore, Keith Seymour, Jaewook Shin, Ananta Tiwari, Sam Williams, Haihang You, "PERI Autotuning," *Journal of Physics: Conference Series 125* (2008), November 2008.

[Hall2009] M. Hall, J. Chame, C. Chen, J. Shin, G. Rudy, M. Khan, "Loop Transformation Recipes for Code Generation and Autotuning", *The 22nd International Workshop on Languages and Compilers for Parallel Computing*, October 2009.

[Shin2010] Jaewook Shin, Mary Hall, Jacqueline Chame, Chun Chen, Paul Fisher and Paul Hovland, "Speeding up Nek5000 with Autotuning and Specialization", *The 24th International Conference in Supercomputing (ICS 2010)*, June 2010.

[Rudy2010] G. Rudy, M. Hall, C. Chen, J. Chame, M. Khan, "A Programming Language Interface to Describe Transformations and Code Generation," *The 23rd International Workshop on Languages and Compilers for Parallel Computing*, October 2010.

[Chame 2011] Jacqueline Chame, Chun Chen, Mary Hall, Jeffrey K. Hollingsworth, Kumar Mahinthakumar, Gabriel Marin, Shreyas Ramalingam, Sarat Sreepathi, Vamsi Sripathi, Ananta Tiwari, "PERI Autotuning of PFLOTRAN," *Journal of Physics (to appear), Proceedings of SciDAC 2011*, July 2011.