

# Understanding and IMPROVING Large-Scale CODES

Achieving good performance on high-end computing systems is growing ever more challenging due to enormous scale, increasing architectural complexity, and increasing application complexity. To address these challenges in DOE's SciDAC-2 program, the Performance Engineering Research Institute has embarked on an ambitious research plan encompassing performance modeling and prediction, automatic performance optimization, and performance engineering of high-profile applications.

Within just five years, systems with one million processors are expected, which poses a challenge not only to application developers but also to those engaged in performance tuning.

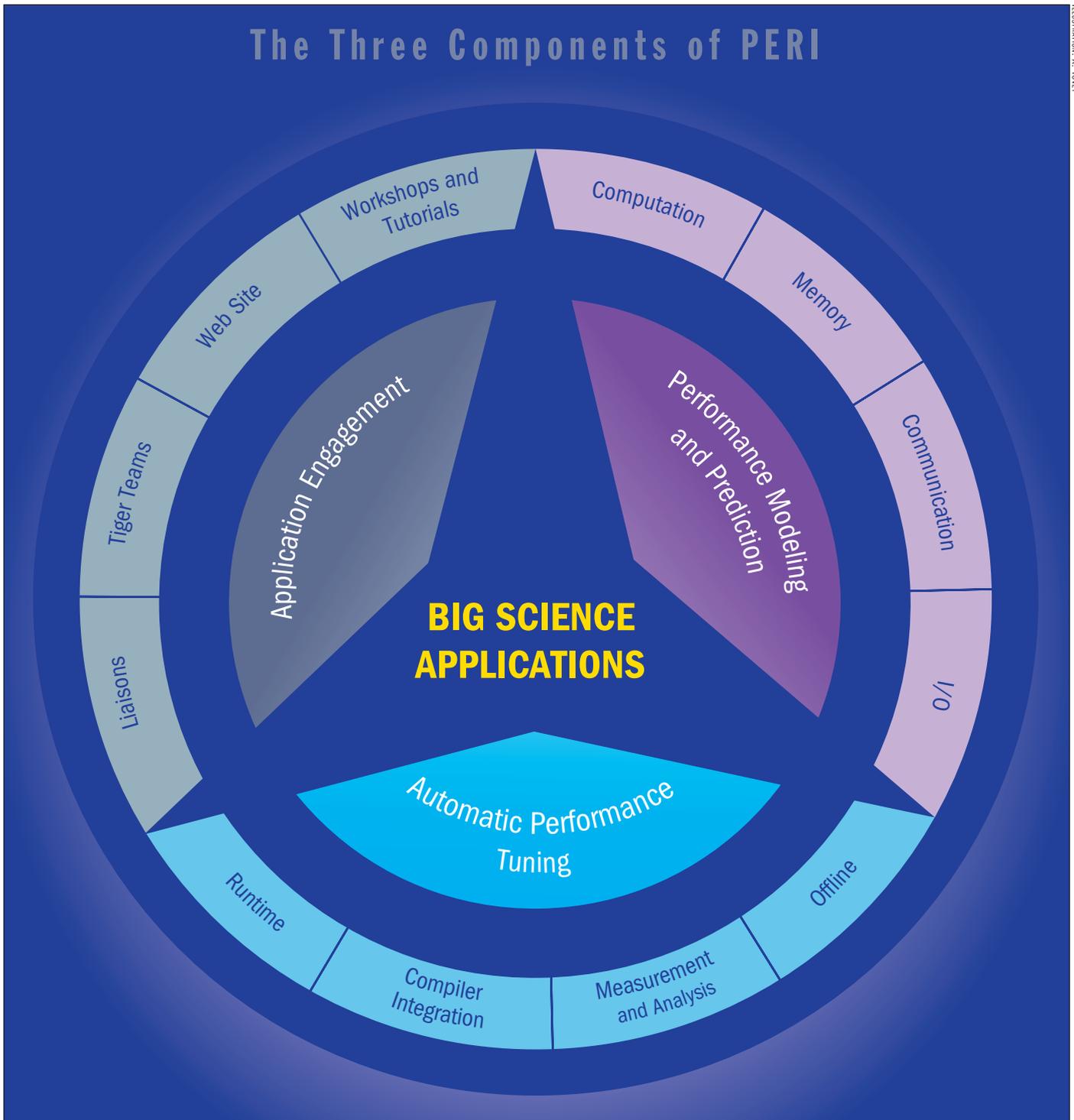
## A SciDAC Institute

Understanding and enhancing the performance of large-scale scientific programs is a crucial component of the high-performance computing world. This is due not only to the increasing processor count, architectural complexity, and application complexity, but also to the sheer cost of these systems. A quick calculation shows that a 30% increase in the performance of two of the major SciDAC applications codes—which together use approximately 10% of the National Energy Research Scientific Computing (NERSC) Center and Oak Ridge National Laboratory (ORNL) high-end systems over three years—represents a saving of around \$6 million.

Within just five years, systems with one million processors are expected, which poses a challenge not only to application developers but also to those engaged in performance tuning. Earlier research and development by us and others in the performance research area focused on the memory wall—the rising disparity between processor speed and memory latency. Now the emerging multi-core commodity microprocessor designs, with many processors on a single chip and large shared caches, create even greater penalties for off-

chip memory accesses and further increase optimization complexity. With the release of systems such as the Cray X1, custom vector processing systems have re-emerged in U.S. markets. Other emerging designs include single-instruction multiple-data (SIMD) extensions, field-programmable gate arrays (FPGAs), graphics processors, and the Sony–Toshiba–IBM Cell processor. Understanding the performance implications for such diverse architectures is a daunting task.

In concert with the growing scale and complexity of the systems is the growing scale and complexity of the scientific applications themselves. Applications are increasingly multilingual, with source code and libraries created using a blend of Fortran 77, Fortran 90, C, C++, Java, and even interpreted languages such as Python. Large applications typically have rather complex build processes, involving code preprocessors, macros, and make files. Effective performance analysis methodologies must deal seamlessly with such structures. Applications can be large, often exceeding one million lines of code. Optimizations may be required at many locations in the code, and seeming local changes can affect global data structures. Applications are often compo-



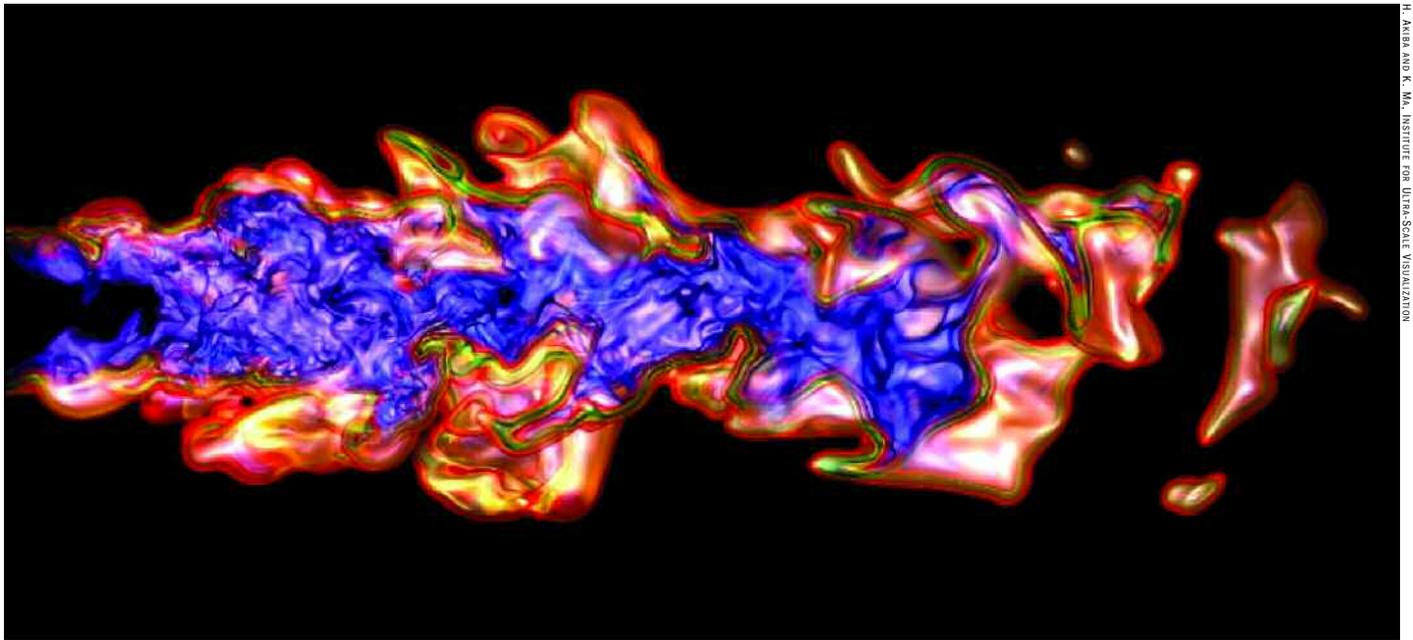
**Figure 1.** The three components of the SciDAC Performance Engineering Research Institute (PERI).

mentized and performance can depend significantly on the context in which the components are used. Finally, applications increasingly involve advanced features such as adaptive mesh refinement, data-intensive operations, and multi-scale, multi-physics, and multi-method computations.

The Performance Engineering Research Institute (PERI) emphasizes three aspects of performance tuning for high-end systems and the complex SciDAC applications that run on them:

- Performance modeling of applications and systems
- Automatic performance tuning
- Application engagement and tuning

Figure 1 illustrates the interplay of PERI's activities, which are the focus of this article. The next section discusses the modeling activities undertaken both to better understand the performance of applications and to determine reasonable



H. AKIBA AND K. MA, INSTITUTE FOR ULTRA-SCALE VISUALIZATION

**Figure 2.** Heat release (red) and progress variable (purple) of a turbulent lean methane–air Bunsen flame simulated using S3D. The volume rendering was performed by Dr. Hiroshi Akiba and Dr. Kwan-Liu Ma of the University of California–Davis and the SciDAC Institute for Ultra-Scale Visualization, and by Dr. Ramanan Sankaran, Dr. Evatt R. Hawkes, and Dr. Jacqueline H. Chen of SNL.

bounds on expected performance. Following that is a presentation of the PERI vision for creating an automatic performance tuning capability, which ideally will alleviate scientific programmers of this burden. Automating performance tuning is a long-term research project, and the SciDAC program has scientific objectives that will benefit greatly from its outcome. This is followed by a discussion of how PERI is engaging with DOE computational scientists to address today’s most pressing performance problems. The article concludes with a summary of the current state of the PERI SciDAC-2 project. This effort is just beginning, but substantial progress has been made, both in forming the team and striving towards meeting goals—the near-term goal of helping DOE successfully transition into the petascale era and the long-term research goal of automating the process of performance tuning (sidebar “PERI Goals,” p29).

### Performance Modeling and Prediction

The goal of performance modeling is to understand the performance of an application on a computer system via measurement and analysis. This information can be used for a variety of tasks: evaluating architectural tradeoffs early in the system design cycle, validating performance of a new system installation, guiding algorithm choice when developing a new application, improving optimization of applications on specific platforms, and guiding the application of techniques for automated tuning and optimization. Modeling is now an integral part of many high-end system procurements, thus making per-

formance research useful beyond the confines of performance tuning. For performance engineering, modeling analyses—when coupled with empirical data—can signal when tuning is needed and, just as importantly, when tuning is complete. Naturally, if models are to support automatic performance tuning, then the models themselves must be automatically generated.

Traditional performance modeling and prediction has been done via some combination of three methods—analytical modeling, statistical modeling derived from measurement, and simulation. In the earlier SciDAC-1 Performance Evaluation Research Center (PERC), researchers developed a semi-automatic yet accurate methodology based on application signatures, machine profiles and convolutions. These methodologies allow us to predict performance to within reasonable tolerances for an important set of applications on traditional clusters of SMPs for specific inputs and processor counts.

PERI is extending these techniques to account not only for the effects of emerging architectures, but also to model scaling of input and processor counts. It has been shown that modeling the response of a system’s memory hierarchy to an application’s workload is crucial for accurately predicting its performance on today’s systems with their deep memory hierarchies. The current state-of-the-art works well for weak scaling, that is, increasing the processor count proportionally with input. PERI is developing advanced schemes for modeling application performance, such as by using neural networks. Researchers are also exploring variations of exist-

Automating performance tuning is a long-term research project, and the SciDAC program has scientific objectives that will benefit greatly from its outcome.

# PERI Goals

The Performance Engineering Research Institute for Enabling Technology (PERI) is a SciDAC Institute focused on delivering petascale performance to complex scientific applications running on Leadership Class computing systems.

As SciDAC and the scientific computing community look to the future, achieving good 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 is pursuing a unified, tripartite research plan encompassing:

- Performance modeling and prediction
- Automatic performance optimization
- Performance engineering of high profile applications

The PERI performance modeling and prediction activity will develop and refine performance models.

This will significantly reduce the cost of collecting data upon which the models are based, and will increase model fidelity and speed.

The automatic performance optimization effort is spurred by the strong user preference for automatic tools. This work is building on previous successful activities such as ATLAS, which has automatically tuned components of the LAPACK linear algebra library, the highly successful FFTW library, and other recent work (sidebar “Software Overview,” p32).

In the third major component, application engagement, PERI directly interacts with SciDAC applications, including tiger teams that focus on particular codes (sidebar “The GTC Tiger Team,” p34).

The PERI website (Further Reading, p35) highlights the tools and publications produced by the PERI project, and most of the ongoing work and discussion can be found on the PERI wiki: [http://www.peri-scidac.org/wiki/index.php/Main\\_Page](http://www.peri-scidac.org/wiki/index.php/Main_Page)

One of the goals is to provide the ability to reliably forecast the performance of a code on a machine size that has not yet been built.

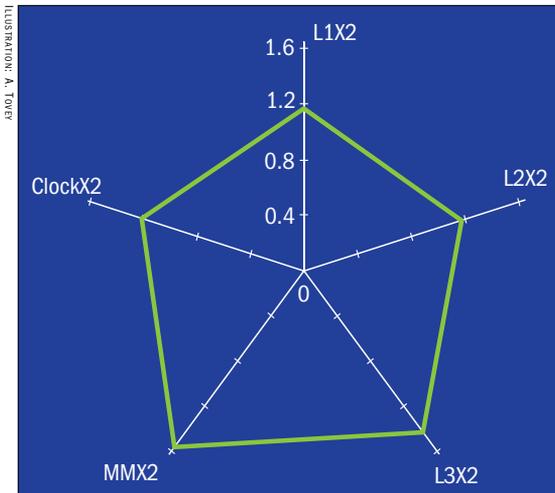


Figure 3. Kiviatt diagram of Power4 system options.

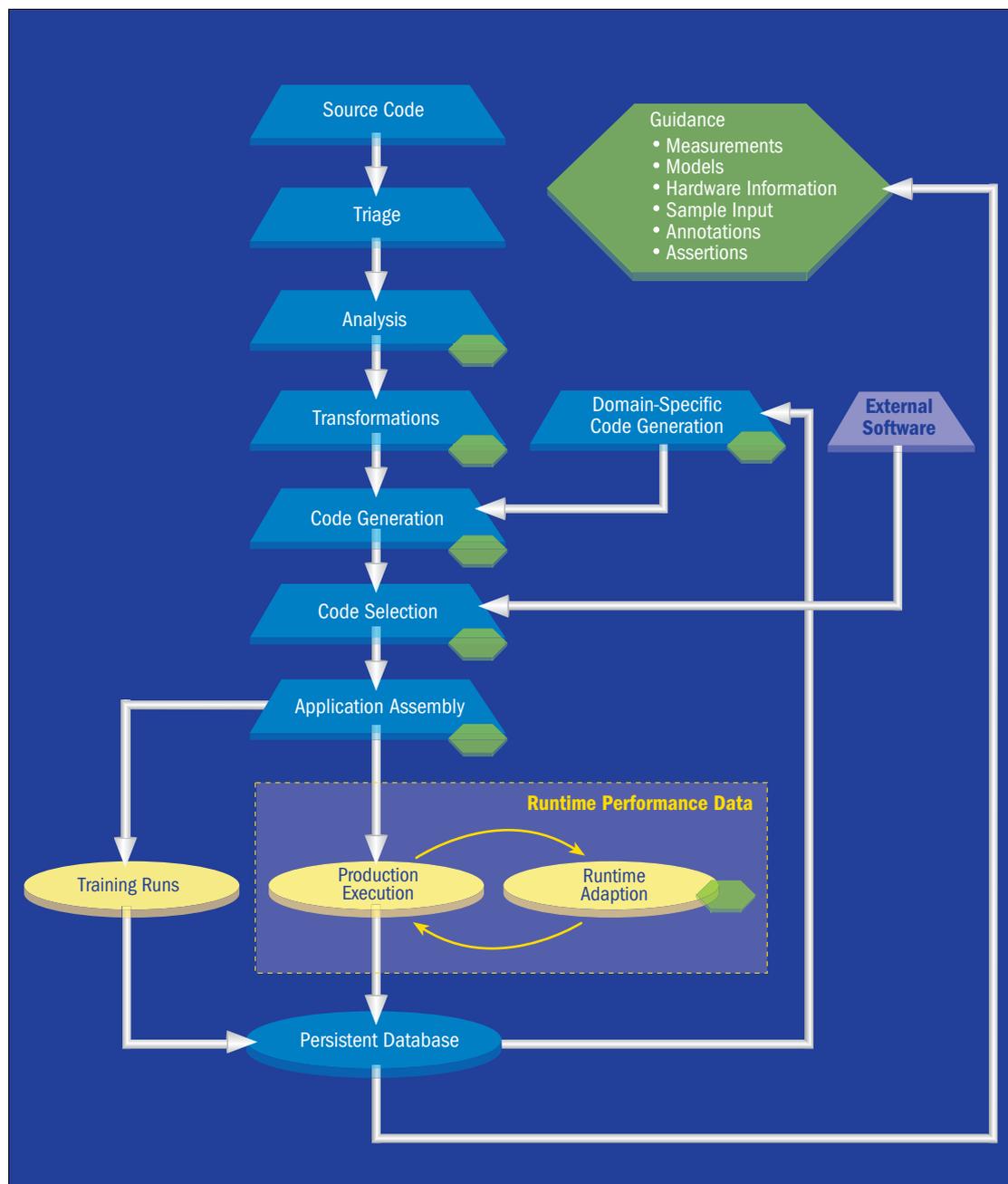
ing techniques and parameterized statistical models built from empirical observations to predict application scaling. This Institute is also pursuing methods for automated extrapolation of scaling models, as a function of increasing processor count, while holding the input constant. One of the goals is to provide the ability to reliably forecast the performance of a code on a machine size that has not yet been built.

PERI is also extending the framework to model communication performance as a function of the type, size, and frequency of application messages, and the characteristics of the interconnect. Several parallel communication models have been

developed that predict performance of message-passing operations based on system parameters. Assessing the parameters for these models within local area networks is relatively straightforward and the methods to approximate them have already been established, and so they are well understood. The models, which are similar to PlogP, capture the effects of network bandwidth and latency; however, a more robust model must also account for noise, contention, and concurrency limits. PERI is developing performance models directly from observed characteristics of applications on existing architectures. Predictions from such models can serve as the basis to optimize collective MPI operations, and permit us to predict network performance in a very general way. This work will require us to develop a new open-source network simulator to analyze communication performance.

Finally, researchers will reduce the time needed to develop models, since automated tuning requires on-the-fly model modification. For example, a compiler or application may propose a code change in response to a performance observation and need an immediate forecast of the performance impact of the change. Dynamic tracing—the foundation of current modeling methods—requires running existing codes and can be quite time consuming. Static analysis of binary executables can make trace acquisition much faster by limiting it to only those features that are not known before execution. User annotations can

Most of the energy consumed by mankind is derived from combustion, and S3D was developed to help scientists better model and understand its fundamental properties.



**Figure 4.** The PERI automatic tuning workflow. Blue polygons indicate specific tools or parts of tools to support automated empirical tuning. Yellow ovals indicate activities that are part of a code that is using automatic tuning at runtime. Green hexagons indicate information may be supplied to guide the optimization selection during empirical tuning. The large green hexagon lists the type of information that may be used.

broaden the reach of modeling by specifying at a high level the expected characteristics of code fragments. Application phase modeling can reduce the amount of data required to form models. PERI is exploring less expensive techniques to identify dynamic phases through statistical sampling and time-series cluster analysis. For on-the-fly observation, DynInst is being used to attach to a running application, slow it down momentarily to measure something, then detach. PERI will advance automated, rapid, machine-independent model formation to push the efficacy of perform-

ance modeling down into lower levels of the application and architecture lifecycle.

**Modeling Combustion**

PERI’s collaboration with the S3D code team at Sandia National Laboratories (SNL) serves as a good example of modeling work. Most of the energy consumed by mankind is derived from combustion, and S3D was developed to help scientists better model and understand its fundamental properties. Figure 2 (p28) depicts the heat release rate and progress variable of a turbulent

lean methane-air Bunsen flame. Figure 3 (p29) is a Kivi diagram showing predicted performance improvements relative to the IBM Power4 architecture (normalized to 1) for a two-fold improvement in dimensions of CPU clockspeed, L1 bandwidth, L2 bandwidth (inclusive of L1), L3 bandwidth (inclusive of L1 and L2), and main memory bandwidth. Note that S3D has some floating-point limited code, so there is some improvement to be had by doubling the clock, and hence the floating-point issue rate. But, as is usual for many of these models of scientific codes, there appears to be more to be gained by doubling bandwidth to lower levels of the memory hierarchy. From a performance tuning perspective it appears that optimizations affecting data layout and working set size to improve cache miss rates would yield the most significant improvements.

### Automatic Performance Tuning

In discussions with application scientists it is clear that users want to focus on their science, and not be burdened with optimizing their code's performance. Thus, the ideal performance tool analyzes and optimizes performance without human intervention, a long-term vision termed automatic performance tuning. This vision encompasses tools that analyze a scientific application, both as source code and during execution, generate a space of tuning options, and search for a near-optimal performance solution. There are numerous daunting challenges to realizing the vision, including enhancement of automatic code manipulation tools, automatic run-time parameter selection, automatic communication optimization, and intelligent heuristics to control the combinatorial explosion of tuning possibilities. On the other hand, PERI is encouraged by recent successful results such as ATLAS, which has automatically tuned components of the LAPACK linear algebra library (sidebar "Software Overview," p32). Researchers are also studying techniques used in the highly successful FFTW library and several other related projects.

Figure 4 illustrates the automated performance tuning process and integration pursued by PERI. Researchers are attempting to integrate performance measurement and modeling techniques with code transformations to create an automated tuning process for optimizing complex codes on large-scale architectures. The result will be an integrated compile-time and run-time optimization methodology that can reduce dependence on human experts and automate key aspects of code optimization. The color and shape code in figure 4 indicates the processes associated with the automation of empirical tuning on either libraries or whole applications.

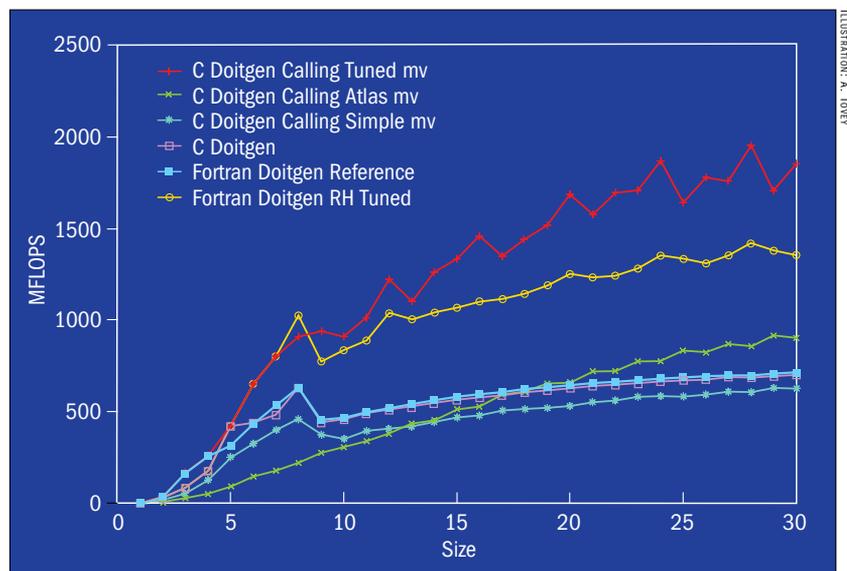


Figure 5. Early automatic tuning result for the doitgen kernel of MADNESS.

As shown in figure 4, the main input to the automatic tuning process is the application source code. In addition, there may also be external code such as libraries, ancillary information such as performance models or annotations, sample input data, and historical data from previous executions and analyses. With these inputs, it is anticipated that the automatic tuning process involves the following steps:

**Triage.** This step involves performance measurement, analysis, and modeling to determine whether an application has opportunities for optimization.

**Semantic analysis.** This step involves analysis of program semantics to support safe transformation of the source code, including traditional compiler analyses to determine data and control dependencies. Here, semantic information provided by the user can also be exploited.

**Transformation.** Transformations include traditional optimizations such as loop optimizations and in-lining, as well as more aggressive data structure reorganizations and domain-specific optimizations. Tiling transformations may be parameterized to allow for input size and machine characteristic tuning. Unlike traditional compiler transformations, this allows user input.

**Code generation.** The code generation phase produces a set of possible implementations to be considered. Code generation may either come from general transformations to source code in an application or from a domain-specific tool that produces a set of implementations for a given computation, as is the case with the ATLAS BLAS generator.

There likely will not be a single automatic tuning tool, but rather a suite of interacting tools which are themselves research projects.

## Software Overview

The following provides an overview of PERI performance analysis software. More detailed information about each software tool can be found on individual tool homepages, which are provided below, and as links at: <http://www.peri-scidac.org/perci/tools/>

**Active Harmony** is software architecture that supports distributed execution of computational objects and emphasizes adapting to heterogeneous and changing environments. <http://www.dyninst.org/harmony/>

**ATLAS** (Automatically Tuned Linear Algebra Software) provides C and Fortran 77 interfaces to a portably efficient BLAS implementation, as well as a few routines from LAPACK. <http://math-atlas.sourceforge.net/>

**HPCToolkit** is an open-source suite of multi-platform tools for profile-based performance analysis of applications. <http://www.hipersoft.rice.edu/hpctoolkit/>

**KOJAK** is a trace-based performance-analysis tool for parallel applications supporting the programming models MPI, OpenMP, SHMEM, and combinations thereof. It includes instrumentation, post-processing of performance data, and result presentation. <http://icl.cs.utk.edu/kojak/>

**mpiP** is a lightweight profiling library for MPI applications. <http://sourceforge.net/projects/mpiP>

**OSKI** (Optimized Sparse Kernel Interface) is a collection of low-level C primitives that provide automatically tuned computational kernels on sparse matrices. <http://bebop.cs.berkeley.edu/oski/>

**PAPI** (Performance Application Programming Interface) provides a cross-platform interface to the hardware performance counters found in most modern microprocessors. <http://icl.cs.utk.edu/papi/>

**PDT** (Program Database Toolkit) is a framework for analyzing source code written in several programming languages. It is useful for making rich program knowledge accessible to developers of static and dynamic analysis tools. <http://www.cs.uoregon.edu/research/pdt>

**Rose** is a project to define a new type of compiler technology that allows compilation techniques to address the optimization of user-defined abstractions. Tools support C, C++, and Fortran 90. <http://www.llnl.gov/CASC/rose/>

**SvPablo** is a graphical performance analysis environment for performance tuning and visualization. This tool supports both interactive and automatic source-code instrumentation. <http://www.renci.org/projects/pablo.php>

**TAU** is a portable profiling and tracing toolkit for performance analysis of parallel programs written in Fortran, C, C++, Java, and Python. <http://www.cs.uoregon.edu/research/tau>

**Offline search.** This phase evaluates the generated code to select the “best” version. Offline search entails running the generated code and searching for the best-performing implementation. The search process may be constrained by guidance from a performance model or user input. By viewing these constraints as guidance, PERI allows the extremes of pure search-based, model-based, or user-directed, as well as arbitrary combinations.

**Application assembly.** At this point, the components of optimized code are integrated to produce an executable code, including possible instrumentation and support for dynamic tuning.

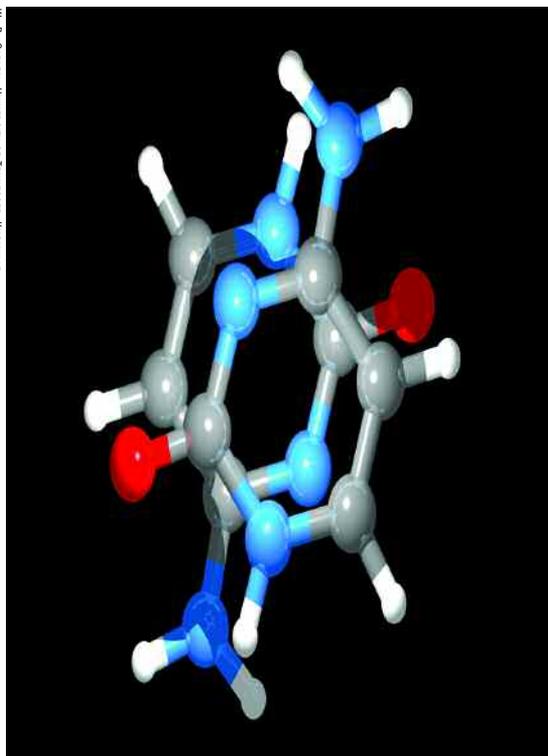
**Training runs.** Training runs involve a separate execution step designed mainly to produce performance data for feedback into the optimization

process. This step may be used prior to a large run to have the code well-tuned for a particular input set.

**Online adaptation.** Finally, optimizations may occur during production runs, especially for problems or machines whose optimal configuration changes during the execution.

Automatic tuning of a particular application need not involve all of these steps. Furthermore, there likely will not be a single automatic tuning tool, but rather a suite of interacting tools which are themselves research projects.

A key part of the automatic tuning process is the maintenance of a persistent store of performance information from both training and production runs. Of particular concern are changes in the behavior of production codes over time. Such changes can be symptomatic of changes in the



**Figure 6.** DNA base pair stacking energy calculations using the MADNESS code.

hardware, of the versions and configuration of system software, of changes to the application, or of changes to problems being solved. Regardless of the source, such changes require analysis and remediation. The problem of maintaining persistent performance data is recognized across the HPC community. PERI therefore formed a Performance Database Working Group, which involves PERI researchers as well as colleagues at the University of Oregon, Portland State University, and Texas A&M University. The group has developed technology for storing performance data collected by a number of performance measurement and analysis tools, including TAU, PerfTrack, Prophecy, and SvPablo. The PERI Database system provides web interfaces that link to the performance data and analysis tools in each tool's home database.

### Application to MADNESS

While automatic tuning is PERI's long-term research goal, several of PERI's research groups have already applied their autotuning frameworks to *doitgen*, a core computational kernel in MADNESS ("Speed and Precision in Quantum Chemistry," *SciDAC Review*, Spring 2007, p54), which computes the reduction sum of a three-dimensional matrix multiplied by a two-dimensional matrix. The group experimented with a range of transformations including array contraction, loop unrolling, and a customized code generator for matrix-vector multiplication. Figure 5 (p31)

shows the results on an Opteron™. The automatically tuned code fragment is 1.5 times faster than the hand-tuned Fortran version. PERI also experimented with compiler-generated code for use of Intel's SSE-3 SIMD compute engine, achieving up to a 1.23-fold speedup over the original, hand-tuned code.

This experiment also illustrated the importance of tuning the code in the context of the overall application and pointed out the limitations of tuning isolated kernels. When the autotuned code was put back into the overall MADNESS application, the entire application ran slightly slower than before. The reason was that only one case had been optimized, and the optimizations had slowed down others. After discussion with the code developer, PERI is ready to proceed to achieve an overall improvement in the entire application's performance.

### Application Engagement

The key long-term research objective of PERI is to automate as much of the performance tuning process as possible. Ideally in five years PERI will produce a prototype of the kind of system that will free scientific programmers from the burden of tuning their codes, especially when simply porting from one system to another. While this may offer today's scientific programmers hope for a brighter future, it does little to help with the immediate problems they face as they try to ready their codes for petascale machines. PERI has therefore created a third activity called application engagement wherein PERI researchers will bring their tools and skills to bear in order to help DOE meet its performance objectives, and to ground the Institute's own research in practical experience. PERI has a two-pronged application engagement strategy.

The first strategy is establishing long-term liaison relationships with many of the application teams. PERI liaisons who work with application teams without significant, immediate performance optimization needs provide these application teams with advice on how to collect performance data and track performance evolution, and ensure that PERI becomes aware of any changes in these needs. For application teams with immediate performance needs, the PERI liaison works actively with the team to help them meet their needs, utilizing other PERI personnel as needed. The status of a PERI liaison activity, passive or active, changes over time as the performance needs of the application teams change. As of June 2007, PERI is working actively with six application teams and passively with ten others. The nature of each interaction is specific to each application team.

The key long-term research objective of PERI is to automate as much of the performance tuning process as possible.

The tiger teams, consisting of several PERI researchers, strive to improve application performance by applying the full range of PERI capabilities.

## The GTC Tiger Team

A more detailed description of the Gyrokinetic Turbulence Code (GTC) Tiger Team further illustrates PERI application engagement activities. GTC is a particle-in-cell (PIC) code for gyrokinetic simulation of fusion plasmas for studying turbulent transport. Figure 8 is an illustration of a self-consistently generated electrostatic potential during the non-linear phase of the turbulence. GTC’s developers have the long-term performance goal of scaling the new shaped version of GTC (GTC\_s) to tens of thousands of cores to enable simulation of ITER-size plasmas.

A core group of PERI researchers at the University of Tennessee and Rice University, supplemented by outside participants at the University of Oregon and Texas A&M University, make up the PERI GTC Tiger Team. This team is studying the performance

characteristics of GTC and is collaborating with the code developers on performance optimization of GTC\_s. A timing profile for a 64-processor run on the ORNL Cray XT3/4 is shown in figure 7. Inter-process communication is shown to take a minimal amount of time. The PUSH1 and CHARGE1 routines implement a scatter-gather algorithm that is a known performance bottleneck. Initial hand optimization has improved the performance of the scatter algorithm by approximately 10%. Current effort is focused on optimizing the data layout—that is, the order in which the particles are processed—to improve the cache and TLB performance of the scatter-gather. In particular, researchers are investigating a space-filling curve approach to re-ordering the particles in an optimal manner.

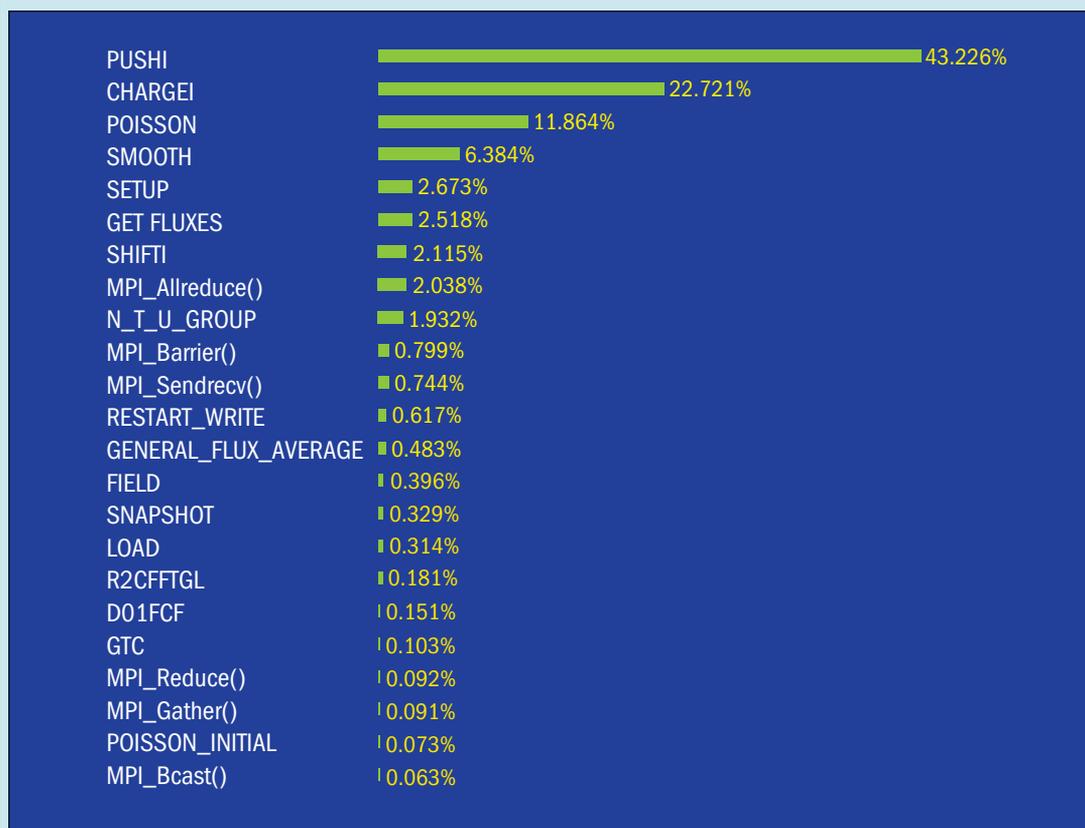
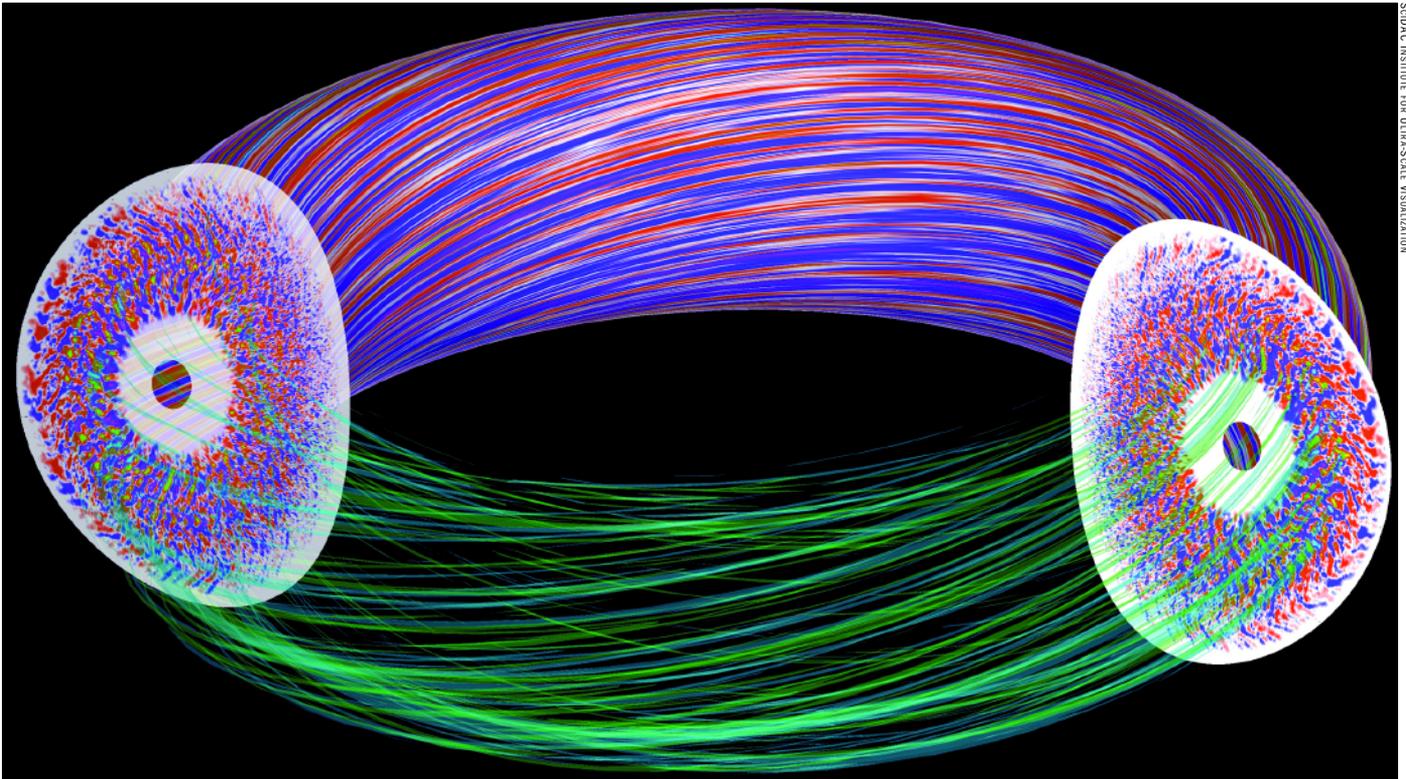


Figure 7. Mean time profile for GTC\_s on 64 processors of Jaguar, ORNL’s Cray XT3/4.

The other primary PERI application engagement strategy involves tiger teams. A tiger team works directly with application teams with immediate, high-profile performance requirements. The tiger teams, consisting of several PERI researchers, strive to improve application performance by applying the full range of PERI capabilities, including not only performance modeling and automated tuning research but

also in-depth familiarity with today’s state-of-the-art performance-analysis tools. Tiger team assignments are of a relatively short duration, lasting from six months to a year. As of June 2007, PERI tiger teams are working with two application codes that will be part of the 2007 JOULE report, S3D and GTC\_s (sidebar “The GTC Tiger Team”). PERI has already identified significant opportunities for performance



**Figure 8.** *GTC\_s image of the electrostatic potential in a plasma.*

improvements on both applications. Current work is focused on providing these improvements through automated tools that support the continuing code evolution required by the JOULE criteria.

### Summary

PERI was created to focus on the increasingly difficult problem of achieving high throughput on large-scale scientific computing systems. These performance challenges arise not only from the scale and complexity of leadership-class computers, but also from the increasing sophistication of today's scientific software. Experience has shown that scientists want to focus their programming efforts on discovery and do not want to be burdened by the need to constantly refine their codes to maximize performance. Performance tools that they can use themselves are not embraced, but rather viewed as a necessary evil.

To relieve scientists from the burden of performance tuning, PERI has embarked on a research program addressing three different aspects of performance tuning—performance modeling of applications and systems, automatic performance tuning, and application engagement and tuning. PERI's application engagement activities are intended to both help scientists address today's performance related problems, and automatic performance tuning research will lead to technology in the future that will significantly

reduce—and, possibly, someday eliminate—this burden. Performance modeling informs both of these activities.

Like all SciDAC-2 projects, PERI is a new project, but it builds on five years of SciDAC-1 research and decades of prior work. PERI is off to a good start, and its investigators have already made contributions to SciDAC-2 and to DOE's 2007 Joule codes. The PERI team looks forward confidently to an era of petascale computing in which scientific codes migrate amongst a variety of leadership-class computing systems without overburdening developers with the need to continually retune in order to achieve acceptable levels of throughput. ●

**Contributors:** Dr. David H. Bailey, LBNL; Dr. Robert Lucas (PI), University of Southern California; Dr. Paul Hovland and Dr. Boyana Norris, ANL; Dr. Kathy Yelcik and Dr. Dan Gunter, LBNL; Dr. Bronis de Supinski and Dr. Dan Quinlan, LLNL; Dr. Pat Worley, Dr. Jeff Vetter, and Dr. Phil Roth, ORNL; Dr. John Mellor-Crummy, Rice University; Dr. Allan-Snavely, University of California—San Diego; Dr. Jeff Hollingsworth, University of Maryland; Dr. Dan Reed, Dr. Rob Fowler, and Dr. Ying Zhang, University of North Carolina; Dr. Mary Hall and Dr. Jacques Chame, University of Southern California; Dr. Jack Dongarra and Dr. Shirley Moore, University of Tennessee—Knoxville

### Further Reading

<http://www.peri-scidac.org>