# Modeling the Office of Science Ten Year Facilities Plan: The PERI Architecture Tiger Team

**Article** *in* Journal of Physics Conference Series · July 2009

DOI: 10.1088/1742-6596/180/1/012039

David H. Bailey
Lawrence Berkeley National Laboratory
**381** PUBLICATIONS **12,686** CITATIONS
SEE PROFILE

Laura Carrington
San Diego Supercomputer Center
**76** PUBLICATIONS **1,738** CITATIONS
SEE PROFILE

Anshu Dubey
Argonne National Laboratory
**84** PUBLICATIONS **1,339** CITATIONS
SEE PROFILE

Dan Gunter
Lawrence Berkeley National Laboratory
**96** PUBLICATIONS **7,604** CITATIONS
SEE PROFILE

**Title:**
Modeling the Office of Science Ten Year Facilities Plan: The PERI Architecture Tiger Team

**Author:**
de Supinski, Bronis R.

**Publication Date:**
10-09-2009

**Publication Info:**
Lawrence Berkeley National Laboratory

**Permalink:**
http://escholarship.org/uc/item/88k1x70r

# Modeling the Office of Science Ten Year Facilities Plan: The PERI Architecture Tiger Team

**Bronis R. de Supinski**[1], **Sadaf Alam**[2], **David H. Bailey**[3], **Laura Carrington**[4], **Chris Daley**[5], **Anshu Dubey**[5], **Todd Gamblin**[1], **Dan Gunter**[3], **Paul D. Hovland**[6], **Heike Jagode**[7], **Karen Karavanic**[8], **Gabriel Marin**[2], **John Mellor-Crummey**[9], **Shirley Moore**[7], **Boyana Norris**[6], **Leonid Oliker**[3], **Catherine Olschanowsky**[4], **Philip C. Roth**[2], **Martin Schulz**[1], **Sameer Shende**[10], **Allan Snavely**[4], **Wyatt Spear**[10], **Mustafa Tikir**[4], **Jeff Vetter**[2], **Pat Worley**[2], and **Nicholas Wright**[4]

[1] Lawrence Livermore National Laboratory, Livermore, California
[2] Oak Ridge National Laboratory, Oak Ridge, Tennessee
[3] Lawrence Berkeley National Laboratory, Berkeley, California
[4] San Diego Supercomputing Center, San Diego, California
[5] University of Chicago, Chicago, Illinois
[6] Argonne National Laboratory, Argonne, Illinois
[7] University of Tennessee - Knoxville, Knoxville, Tennessee
[8] Portland State University, Portland, Oregon
[9] Rice University, Houston, Texas
[10] University of Oregon, Eugene, Oregon

E-mail: bronis@llnl.gov, alamsr@ornl.gov, dhbailey@lbl.gov, lcarring@sdsc.edu, cdaley@flash.uchicago.edu, a-dubey1@uchicago.edu, tgamblin@llnl.gov, dkgunter@lbl.gov, hovland@mcs.anl.gov, jagode@eecs.utk.edu, karavan@cs.pdx.edu, maring@ornl.gov, johnmc@cs.rice.edu, shirley@cs.utk.edu, norris@mcs.anl.gov, loliker@lbl.gov, cmills@sdsc.edu, rothpc@ornl.gov, schulzm@llnl.gov, sameer@cs.uoregon.edu, allans@sdsc.edu, wspear@cs.uoregon.edu, mtikir@sdsc.edu, vetter@ornl.gov, worleyph@ornl.gov, nwright@sdsc.edu

**Abstract.** The Performance Engineering Institute (PERI) originally proposed a tiger team activity as a mechanism to target significant effort optimizing key Office of Science applications, a model that was successfully realized with the assistance of two JOULE metric teams. However, the Office of Science requested a new focus beginning in 2008: assistance in forming its ten year facilities plan. To meet this request, PERI formed the Architecture Tiger Team, which is modeling the performance of key science applications on future architectures, with S3D, FLASH and GTC chosen as the first application targets. In this activity, we have measured the performance of these applications on current systems in order to understand their baseline performance and to ensure that our modeling activity focuses on the right versions and inputs of the applications. We have applied a variety of modeling techniques to anticipate the performance of these applications on a range of anticipated systems. While our initial findings predict that Office of Science applications will continue to perform well on future machines from major hardware vendors, we have also encountered several areas in which we must extend our modeling techniques in order to fulfill our mission accurately and completely. In addition, we anticipate that models of a wider range of applications will reveal critical differences between expected future systems, thus providing guidance for future Office of Science procurement decisions, and will enable DOE applications to exploit machines in future facilities fully.

## 1. Introduction

Sustained performance improvements are integral to the DOE Office of Science SciDAC program's mission to advance large-scale scientific modeling and simulation. Simulation is a key investigative technique for disciplines where experimentation is expensive, dangerous, or impossible. Increased performance can enable faster simulations and more timely predictions, or it can be used to increase the accuracy of existing physical models, enabling more predictive simulations. Research enabled by the SciDAC program will have far-reaching effects in fields such as basic energy, biology, environmental science, fusion energy, and high-energy physics.

The Performance Engineering Research Institute (PERI) tiger team activity targets critical SciDAC performance needs. The original intent was for each tiger team to focus the efforts of several PERI researcher on improving performance of an Office of Science application, with the application selected based on Office of Science mission objectives and application readiness for the focused effort. Thus, each tiger team was envisioned as a relatively short-term activity (six months to at most one year). In 2007, our tiger teams had a positive impact on two key DOE applications participating in the JOULE metric. We improved the performance of a turbulent combustion code (S3D [1]) on Oak Ridge's Cray XT5 Jaguar system by 13%. Similarly, we improved the performance of the Gyrokinetic Toroidal Code (GTC) [2, 3] by 10% on Jaguar and by 15% on Argonne National Laboratory's (ANL's) Intrepid Blue Gene/P system.

In 2008, the Office of Science requested that PERI provide assistance in their ten year facilities plan. In particular, they wanted PERI to provide guidance in how key applications would perform across the range of future systems expected to be offered by major vendors in that period. Thus, we redefined the scope of our tiger team activity to handle this request and started the PERI Architecture Tiger Team. This team's goal is to model the behavior of selected applications and to predict their performance on anticipated future systems, instead of to improve their performance on current systems. To fulfill the goal, we must consider a wider range of Office of Science applications and evaluate the suitability of current and future high performance computing (HPC) architectures for the applications. This broader scope has led us to include nearly all PERI researchers on the Architecture Tiger Team.

Several factors complicate the Architecture Tiger Team's charge. Large scale simulations are complex software artifacts for which the performance depends on input and frequently evolves during the course of a simulation. Further, small source code changes can lead to significant performance changes. Thus, modeling their performance across a variety of existing architectures remains a topic of research. For example, modeling at larger scales than are currently run requires changes to most existing modeling methodologies. In order to model the performance for systems that will emerge over the ten year period, we must not only overcome these challenges but also anticipate how the software, as well as the hardware, will evolve.

For these reasons, we have developed a three-part, iterative plan, with each iteration focusing our modeling effort on a different (or growing) set of applications. First, we extensively measure the performance of the applications at scale with a variety of state-of-the-art performance analysis tools. These measurements ensure that we have appropriate versions of the applications: although we are no longer focused on optimization, we still apply our expertise in this direction. Thus, this activity can also provide some benefit to the application teams.

Second, we use these measurements and other data to create predictive performance models that estimate the scaling properties of current applications on future hardware. In the first iteration of the Architecture Tiger Team, we have applied this strategy to three Office of Science early science applications: S3D, GTC, and FLASH [4, 5, 6], an astrophysical thermonuclear flash simulation. In this paper, we detail the preliminary results of this study, which indicate that these applications will perform well across the breadth of anticipated architectures.

In the third part of our process, we report findings to the Office of Science and work with them to select the applications for the next iteration. We are currently engaged in that selection process for the Architecture Tiger Team's second iteration. We are employing criteria that both reflect the importance of the applications to the Office of Science's mission and attempt to capture the breadth of characteristics of its applications. Simply put, we must ensure that the ten year facilities plan reflects the range of needs of the Office of Science's broad mission.

The rest of this report is organized as follows. We summarize key tools for our large-scale performance measurement activity in Section 2. Section 3 describes our performance modeling techniques. In Section 4, Section 5 and Section 6, we detail our initial findings with the S3D, FLASH, and GTC codes. We then state our initial conclusions and lessons learned for this on-going activity, including guidance in selecting the next set of applications, in Section 7.

## 2. Measurement

We have used a wide variety of performance analysis tools to characterize the behavior of S3D, GTC and FLASH on current Office of Science platforms at scale. The large volumes of performance data complicate performance measurement on systems such as Jaguar and Intrepid. Because these modern parallel applications can have dynamic behavior, understanding their performance can potentially require measuring all application processes. However, the overhead of data collection and aggregation on large machines can perturb running applications, making the measurements, and thus the models that we derive from them, inaccurate. Further, too much performance data can make analysis prohibitively expensive.

### *2.1. Performance Analysis Tools*

To address these challenges, we have employed a wide variety of tools for measuing performance data of the three applications that we studied in the first iteration of the Architecture Tiger Team. We briefly describe some of our key performance tools in this section; we present results of applying them in Sections 4, 5, and 6.

*2.1.1. Vampir* The Vampir suite, developed at the University of Dresden - Germany, consists of VampirTrace for instrumentation, monitoring and recording and VampirServer for visualization and analysis [7, 8]. It stores event traces in the Open Trace Format (OTF) [9]. VampirTrace can examine many performance metrics, *e.g.,* MPI communication events, subroutine calls from user code, hardware performance counters, I/O events and memory allocation. VampirServer implements a client/server model with a distributed server, allowing interactive visualization of traces with over 1,000 processes and uncompressed size of up to 100 GBytes [8].

*2.1.2. mpiP* mpiP [10], an MPI profiling tool, measures cumulative time spent in all MPI call sites across all processes in an application. Like other profiling tools, mpiP only collects statistical information, as opposed to full trace data like Vampir. mpiP generates a single file, which is much smaller than a full trace file, but which loses timing information.

*2.1.3. TAU* The TAU Performance System is a portable profiling and tracing toolkit for performance analysis of parallel programs [11]. It comes with a wide selection of features to measure specific functions, code regions and user-defined events in parallel applications. The user must recompile his or her application with the TAU compilers and then run a parallel job. TAU then outputs a trace or profile as desired. TAU also provides extensive data mining and analysis tools for processing information after it has been measured and stored.

*2.1.4. Libra* Libra [12] is a tool for scalable load-balance analysis developed at Lawrence Livermore National Laboratory and the University of North Carolina at Chapel Hill. Unlike full trace tools, Libra uses aggressive, lossy wavelet compression to reduce the volume of load-balance data significantly before recording it. Libra can achieve 100:1 to 1000:1 compression on load-balance data and it provides a scalable client-side visualization tool for viewing recorded traces. Libra records measured code regions by call site.

### *2.2. Platforms*

In the first iteration of the Architecture Tiger Team, we conducted extensive measurements of our target applications' performance on two leadership-class systems. The first system is Argonne National Laboratory's Blue Gene/P system, Intrepid. Intrepid contains 163,840 PowerPC 450 cores running at 800 MHz, and sustained LINPACK performance of 450 teraflops. The second system is the Cray XT4 Jaguar system at Oak Ridge National Laboratory. Jaguar contains 31,328 Opteron cores running at 2.1 GHz, and has sustained performance of over 205 teraflops. Both systems use quad-core nodes.

Jaguar and Intrepid have slightly different network and I/O configurations. Jaguar uses a 3D mesh network for communication between nodes, whereas Intrepid uses a full 3D torus and also uses a tree network and a barrier network for collective communication. Both systems have dedicated I/O nodes that relay I/O operations between applications and the parallel filesystem. On Intrepid, I/O nodes are internal nodes in the tree network; compute nodes communicate with I/O nodes through the tree network, and the I/O nodes communicate with the parallel filesystem over Myrinet links. On Jaguar, the I/O nodes are situated along one side of the 3D mesh, and compute nodes communicate with them over the mesh network.

## 3. Modeling

We have developed several techniques to predict performance of DOE applications on leadership class facilities. In this section, we give a brief overview of these techniques, while we discuss the results of applying them to S3D, FLASH and GTC in Sections 4, 5 and 6.

### *3.1. Convolving Machine Profiles with Application Signatures*

To predict the performance of applications on future architectures, we have developed an approach [13] that separates application-specific measurements from machine-specific measurements. Our approach involves two key components:

**Machines Profiles** that characterize the rates at which a machine can (or is expected to) carry out fundamental operations abstract from any particular application;

**Application Signatures** that characterize the fundamental operations that an application must execute independent of any particular machine.

Our approach enables performance predictions of applications on current systems by convolving application signatures with profiles of the existing systems, and on future systems by convolving the application profiles with profiles generated from the *expected* performance parameters of the future systems. Conceptually, a *convolution* defines an algebraic mapping of application signatures onto runtimes to arrive at a performance prediction.

Given an application profile $A$, and a machine profile $M$, we define $P$, a matrix of runtimes, such that $p_{ij} = \sum_{k=1}^{p} a_{ik} m_{kj}$, or:

$$\begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \end{bmatrix} \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \\ m_{41} & m_{42} & m_{43} \\ m_{51} & m_{52} & m_{53} \end{bmatrix}$$

$$e.g., p_{32} = a_{31} m_{12} + a_{33} m_{22} + a_{33} m_{32} + a_{34} m_{42} + a_{35} m_{53}$$

The rows of $P$ correspond to applications while the columns correspond to systems and each $p_{ij}$ is the expected runtime of application $i$ on system $j$. The rows of $A$ are applications, and the columns are operation counts. Any row of $A$ is the *signature* for application $i$. Likewise, rows of $M$ are bandwidths measured by some benchmark for a particular system while each column is the *profile* of a particular system.

This approach is generic, and we could apply it to measurements from any of the tools mentioned in Section 2 to produce application signatures targeted at particular types of analysis. However, in order to reflect the impact of timing considerations, we currently use traces of memory operations to characterize the fundamental operations of computational code regions and message traces similar to those produced by Vampir [14, 15]. We use cache simulation to convolve the signatures of the computational regions with characterizations of the memory system obtained with the MultiMAPS benchmark from SDSC. We then use a high-level network simulation such as Dimemas [16, 17] or SDSC's PSiNS [18] to convolve message trace signature with simple network signatures that capture latency and bandwidth and the predicted performance of the computational regions.

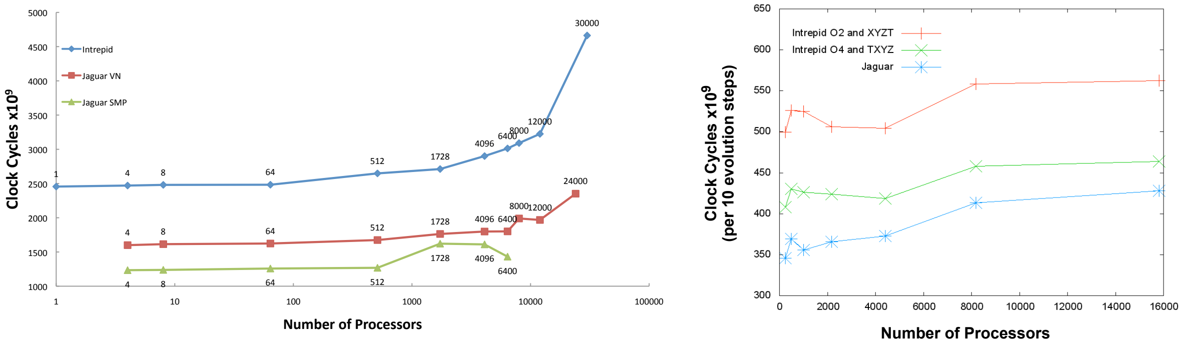
### *3.2. Modeling Assertions*

An alternative modeling strategy, called *Modeling Assertions* (MA) [19], constructs symbolic models of application performance. This technique allows users to annotate their code with expressions revealing the relationships among important input parameters, computation, and communication. These annotations, in the form of pragmas or directives, capture the anticipated performance, in terms of time or other metrics such as cache misses or floating point operations. As the application runs, the MA library checks the model against application structure and key model input parameters. Symbolic performance models complement empirically derived models because the symbolic models expose sensitivities across important parameters and can be scaled to any parameter range.

### *3.3. Load Balance Modeling*

In addition to models for application runtimes, we have developed models of the load balance properties of large systems. Libra's compressed representation of system-wide load balance traces uses a wavelet approximation to allow for multiscale representations of load balance properties. The structure of this approximation supports extraction of a low-resolution model of an application's load balance without recording exhaustive measurements.

This type of compact model will enable us to verify, possibly with distributed extensions of MA, that specific processes have workloads within their expected bounds. Further, our low-resolution models will eventually enable us to *predict*, within some confidence, the evolution of load on large systems and to incorporate dynamically derived load balance guidelines to application-specific load balance components. This solution will dramatically reduce the burden of large-scale measurement on the application developers, enabling them to concentrate on how best to redistribute work within their applications.



(a) S3D Turbulent Combustion

(b) FLASH White Dwarf Deflagration

**Figure 1.** Weak Scaling for Office of Science Codes on Intrepid and Jaguar

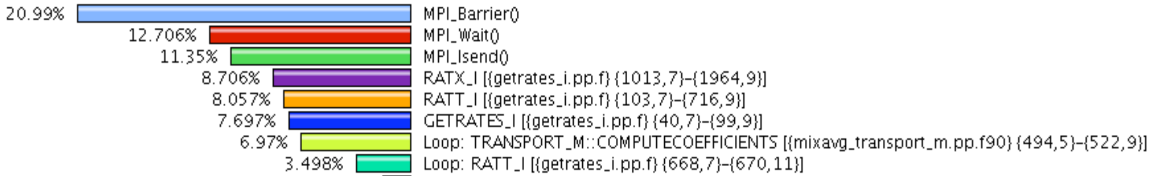


**Figure 2.** Most Time Consuming S3D Routines on Jaguar at Scale

## 4. S3D

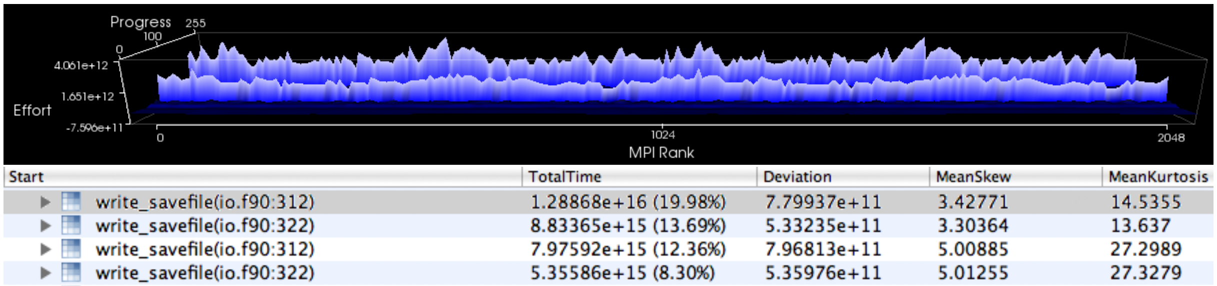
S3D [1] is a state of the art turbulent combustion simulation. The code, which was developed at the Combustion Research Facility at Sandia National Laboratories in Livermore, California, won a 2007 INCITE award for six million hours on the XT3/4 Jaguar system at ORNL's National Center for Computational Sciences. S3D solves the compressible reacting Navier-Stokes equations by using high-fidelity numerical methods. Principal components include an eighth-order finite-difference solver, a fourth-order Runge-Kutta integrator, a hierarchy of molecular transport models and detailed chemistry. The use of direct numerical simulation (DNS) enables scientists to study the microphysics of turbulent reacting flows, as this gives full access to time-resolved fields and provides physical insight into chemistry turbulence interactions. Perhaps more importantly, S3D is critical for accurate simulations of larger systems. The detail afforded by the DNS model enables the development of reduced model descriptions that can be used in macroscale simulations of engineering-level systems.

S3D is architected for scalability. It uses a 3D domain decomposition, where each MPI process manages an equal number of grid points and has the same computational load. Interprocessor communication in this decomposition is only between nearest neighbors, and S3D uses large messages and can overlap communication and computation. All-to-all communication is required only for monitoring and synchronization ahead of I/O.

### *4.1. Measurement of S3D*

We conducted weak scaling measurements of S3D simulating an ethylene burn, with 27,000 grid points per process. Figure 1(a) shows results for Intrepid and Jaguar, in clock cycles required for an entire run. On both systems, S3D scales almost perfectly up to 4,096 cores. After this point, runtimes begin to increase, until at 30,000 processes the runtime is twice that of the baseline, 4-core run on Intrepid. On Jaguar, our 24,000-core run took approximately 30% longer than the baseline run.

We used optimized TAU instrumentation in order to determine the cause of S3D runtime



| Start | TotalTime | Deviation | MeanSkew | MeanKurtosis |
|---|---|---|---|---|
| ▶ write_savefile(io.f90:312) | 1.28868e+16 (19.98%) | 7.79937e+11 | 3.42771 | 14.5355 |
| ▶ write_savefile(io.f90:322) | 8.83365e+15 (13.69%) | 5.33235e+11 | 3.30364 | 13.637 |
| ▶ write_savefile(io.f90:312) | 7.97592e+15 (12.36%) | 7.96813e+11 | 5.00885 | 27.2989 |
| ▶ write_savefile(io.f90:322) | 5.35586e+15 (8.30%) | 5.35976e+11 | 5.01255 | 27.3279 |

**Figure 3.** Percent Total Runtime for Most Costly MPI Operationsin S3D Checkpoint on Intrepid at 16,384 Cores. Load Balance Profile of Topmost `MPI_Barrier()` Callpath is Shown.

increases at scale. Figure 2 shows the top eight entries in the profile; the first three are (`MPI_Barrier`, `MPI_Wait`, and `MPI_Isend`). S3D spends the bulk of its time in these routines at large core counts with its default I/O scheme that writes a file per MPI task. The next most time consuming routines (three subroutines — `RATX_I`, `RATT_I` and `GETRATES_I` — and two loops) are in the parallel solver.

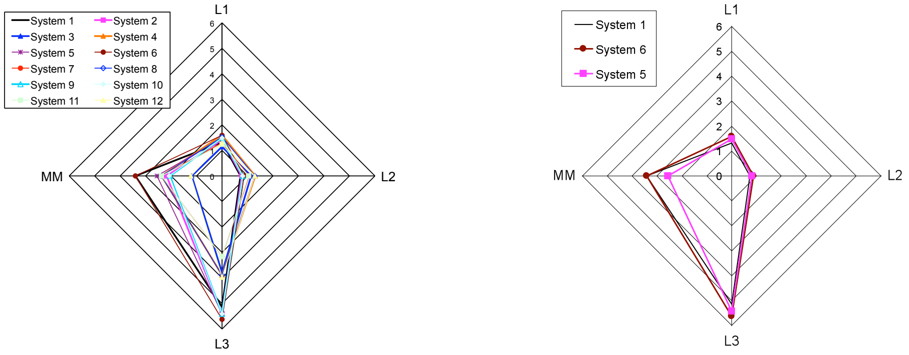
Applying Libra to S3D reveals the underlying issue: the default IO configuration taxes the I/O systems excessively. Figure 3 shows a Libra plot of time spent in the most time-consuming `MPI_Barrier` call on 2,048 processes of a 200-timestep, 16,384 process run of S3D. MPI rank is on the x-axis, and time spent in `MPI_Barrier` per timestep is on the z-axis. Clearly, the variance in the MPI times shown in Figure 2 reflects a load imbalance caused by highly variable times to complete the I/O phase across the MPI tasks. Other I/O configurations, including one that performs writes from a subset of the MPI tasks, offer better scaling performance. Since I/O behavior is a critical component in S3D's overall performance, we are currently working with the Petascale Data Storage Institute (PDSI) to understand and to model it.

### *4.2. Modeling of S3D*

We now detail performance models of S3D's computational regions, leaving models that include I/O for future work. The Kiviat diagram in Figure 4(a) shows anticipated memory system parameters for several hardware vendors, which we anonymize here due to NDA considerations. The four axes are memory bandwidths for L1, L2, and L3 caches and for main memory (MM). System 1 represents these parameters for Jaguar. In the diagram, the noticeable differences between current and future systems are the significant changes in L3 and main memory bandwidth. Our modeling analysis explores the impact that this difference will have on S3D.

Table 1 shows the results of convolving the machine profiles shown in Figure 4(a) with S3D memory profiles. These results indicate that the differences in memory system performance *will* impact S3D runtimes significantly. We predict that S3D's $C_2H_4$ problem will perform well on all expected future systems but will perform best on those systems with the most main memory bandwidth. Although not shown here, this effect is not true for all applications. For example, our predictions indicate the memory system differences will provide little benefit to WRF, a weather forecasting simulation.

An important consideration for models of S3D is that its memory behavior is scale invariant. We have compared memory traces across a range of job sizes and found that they are very consistent under weak scaling. We have previously shown that we can predict S3D performance on Jaguar as we scale the number of MPI tasks directly from traces of smaller runs. Thus, we expect the results in Table 1 to hold for much larger systems.



(a) Twelve Machine Profiles Used with S3D (b) Three Machine Profiles Used with FLASH

**Figure 4.** Memory Profiles of Systems Expected by 2012

| CPUs | Sys. 1 | Sys. 2 | Sys. 3 | Sys. 4 | Sys. 5 | Sys. 6 | Sys. 7 | Sys. 8 |
|---|---|---|---|---|---|---|---|---|
| 8 | 1 | .72 | .78 | .66 | .77 | .81 | .74 | .61 |
| 64 | 1 | .73 | .78 | .67 | .77 | .81 | .73 | .62 |
| 512 | 1 | .72 | .78 | .66 | .77 | .81 | .74 | .62 |

**Table 1.** Prediction of S3D $C_2H_4$ Benchmark Performance on Systems Anticipated by 2012

## 5. FLASH

FLASH is a parallel, block-structured AMR code designed for compressible reactive flows [4, 5, 6]. Its capabilities span a broad range of applications, from laser-driven shock instabilities to fusion burn in type Ia supernovae. FLASH has run successfully on many leadership class systems. It is fully modular, and its components are used to create many different astrophysical applications.

### *5.1. Measuring FLASH*

Figure 1(b) shows CPU cycles per 10 evolution steps of a FLASH white dwarf deflagration simulation, varying system size. The code scales well for on both Intrepid and Jaguar. We observe that the curves are similar, increasing slightly as the MPI task count increases. We note that using a slightly different assignment of MPI tasks to processors makes a significant performance difference on Intrepid, with the normalized performance similar to that on Jaguar, with indications that they might cross at even higher core counts.

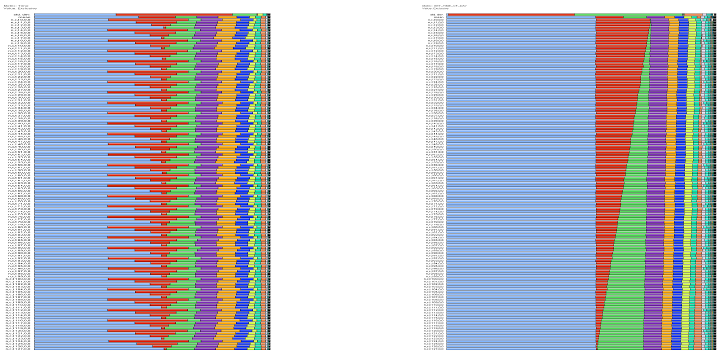
### *5.2. Modeling FLASH*

As with S3D, we modeled the memory behavior of the computational phases of FLASH for three anonymous future architectures for which Figure 4(b) shows memory system profiles. Our predictions in Table 2 show preliminary results for 128, 256 and 384 cores on those systems as well as Jaguar and Lonestar, the Sun Infiniband cluster at the Texas Advanced Computing Center. The reasonable accuracy on the existing systems lends confidence to the predictions on future systems. Overall, the results demonstrate that FLASH also will perform well on anticipated future systems and will benefit from improvements to main memory bandwidth.

We found that FLASH memory traces are not scale-invariant. Thus, although we expect FLASH to scale well based on our empirical measurements, we need to extend our modeling techniques to extrapolate memory traces from a set of traces gathered from smaller runs. We are currently pursuing this research direction. Initial predictions confirm that FLASH will scale reasonably well on future systems, although additional work remains to confirm this hypothesis on the very large processor counts anticipated during the Office of Science's ten year plan.

| CPUs | Lonestar | | | Jaguar | | | Prediction of Systems | | |
|---|---|---|---|---|---|---|---|---|---|
| | Predicted | Real | % err | Predicted | Real | % err | Sys. 1 | Sys. 5 | Sys. 6 |
| 128 | 246 | 227 | 8.2 | 285 | 258 | 10.3 | 195 | 188 | 148 |
| 256 | 127 | 131 | -2.8 | 145 | 138 | 5.5 | 99 | 95 | 75 |
| 384 | 86 | 103 | -16.4 | 99 | 97 | 2.0 | 67 | 64 | 51 |

**Table 2.** Performance Predictions for FLASH on Current and Future Systems



(a) Unoptimized Particle Initialization (b) Optimized Particle Initialization

**Figure 5.** GTC Load Balance Without and With Optimized Initialization

## 6. GTC

The Gyrokinetic Toroidal Code (GTC) is a particle-in-cell code to study microturbulence in magnetically confined fusion plasmas. GTC solves the gyro-averaged Vlasov equation and the gyrokinetic Poisson equation. This global code simulates the entire torus rather than just a flux tube. Written in Fortran 90/95, GTC was originally optimized for superscalar processors but is now a massively parallel code and frequently uses 1024 or more cores.

Our preliminary GTC results demonstrate how our measurement activity ensures that we use a valid version for modeling. Our initial GTC version had portions of the particle initialization commented out. Although the code executed correctly, this change led to significant load imbalance even at small scales. Figure 5(a) shows the TAU profile for GTC on 128 Jaguar cores with the unoptimized initialization. A different color in the figure represents each routine and each row, an MPI task. The figure clearly shows the load imbalance in the staggering of some routines based on the per task workload. The profile in Figure 5(b) shows that the optimized initialization corrects this imbalance. The optimizations result in the profile bars for the routines lining up much more evenly across tasks, thus improving GTC's runtime. This test demonstrates the importance of proper test code configuration for performance modeling.

## 7. Conclusion

We formed the PERI Architecture Tiger Team to assist the Office of Science in formulating its ten year facilities plan. Our role is to provide confidence that future leadership class systems will serve the broad range of simulations needed for the Office of Science to fulfill its mission. The first iteration of our iterative, three phase plan is nearing completion. Its results clearly demonstrate that S3D will perform well on anticipated future platforms with a preference for those that provide the highest main memory bandwidth. Our initial models for FLASH provide similar expectations although we must complete additional research that will enable scaling models for applications that do not exhibit scale-invariant memory reference behavior.

We are currently completing work on measuring and modeling GTC. Our measurement activity of both GTC and S3D demonstrated its value by ensuring that we model an appropriate version of the software. As we complete this first iteration, we are preparing for the next. We recommend to the Office of Science that we select the next set of applications with a careful eye toward those that exhibit performance differences on current platforms and that stress different aspects of memory and network performance. Thus, we anticipate focusing on at least one latency-sensitive application and one bandwidth-sensitive application.

## Acknowledgments

The work of de Supinski, Gamblin and Schulz was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344 (LLNL-CONF-413427). The work of Alam, Jagode, Roth, Vetter and Worley was sponsored by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under contract DE-AC05-00OR22725 with UT-Batelle, LLC. The work of Bailey, Gunter and Oliker was supported by the Director, Office of Computational and Technology Research, Division of Mathematical, Information, and Computational Sciences of the U.S. Department of Energy, under contract DE-AC02-05CH11231. The work of Hovland and Norris was supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under contract DE-AC02-06CH11357. The work of Moore was supported by the U.S. Department of Energy Office of Science under contract DE-FC02-06ER25761. The work of Shende and Spear was performed under the auspices of the U.S. Department of Energy by the University of Oregon under contracts DEFG02-07ER25826 and DE-FG02-05ER25680. Accordingly, the U.S. Government retains a nonexclusive, royalty free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

This research used resources of the National Center for Computational Sciences at Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC05-00OR22725, and resources of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-06CH11357.

## References

[1] Hawkes E R and Chen J H 2004 *Combustion and Flame* **138** 242–258

[2] Lee W W 1983 *Physics of Fluids* **26** 556

[3] Lee W W 1987 *Journal of Computional Physics* **72** 243

[4] Fryxell B, Olson K, Ricker P, Timmes F, Zingale M, Lamb D, MacNeice P, Rosner R, Truran J and Tufo H 2000 *Astrophysical Journal, Supplement* **131** 273–334

[5] Dubey A, Reid L and Fisher R 2008 *Physica Scripta* Special edition from Proceedings of the International Conference "Turbulent Mixing and Beyond," Trieste, Italy, August 2007.

[6] ASC Flash Center 2008 FLASH user's guide `http://flash.uchicago.edu/website/codesupport/secure/flash3_ug/`

[7] Brunst H 2008 *Integrative Concepts for Scalable Distributed Performance Analysis and Visualization of Parallel Programs* Ph.D. thesis Shaker Verlag

[8] Vampirserver user guide URL `http://www.vampir.eu`

[9] Jurenz M *VampirTrace Software and Documentation* ZIH, Technische Universität Dresden URL `http://www.tu-dresden.de/zih/vampirtrace`

[10] Vetter J and Chambreau C 2005 mpiP: Lightweight,scalable MPI profiling URL `http://www.llnl.gov/CASC/mpip`

[11] Shende S and Maloney A 2006 *International Journal of HPC Applications* **20** 287–331

[12] Gamblin T, de Supinski B R, Schulz M, Fowler R J and Reed D A 2008 Scalable load-balance measurement for SPMD codes *Supercomputing 2008 (SC'08)* (Austin, Texas) pp 46–57

[13] Snavely A, Wolter N and Carrington L 2001 Modeling application performance by convolving machine signatures with application profiles *IEEE Workshop on Workload Characterization, 2001.*

[14] Brunst H, Hoppe H C, Nagel W E and Winkler M 2001 Performance optimization for large scale computing: The scalable VAMPIR approach *Proceedings of the 2001 International Conference on Computational Science (ICCS 2001)* (San Francisco, CA) pp 751–760

[15] Brunst H, Kranzlmüller D and Nagel W 2005 *The International Series in Engineering and Computer Science, Distributed and Parallel Systems* **777** 92–102

[16] Labarta J, Girona S and Cortes T 1997 *Parallel Computing* **23** 23–34

[17] Girona S, Labarta J and Badia R M 2000 Validation of Dimemas communication model for MPI collective operations *European PVM/MPI Users' Group Meeting* pp 39–46

[18] Tikir M M, Laurenzano M, Carrington L and Snavely A 2009 PSINS: An open source event tracer and execution simulator for MPI applications *Euro-Par* (Delft, the Netherlands)

[19] Alam S R and Vetter J S 2006 A framework to develop symbolic performance models of parallel applications *IPDPS* (IEEE)