

The Institute for Sustained Performance, Energy, and Resilience

Fall 2013 Director's Report

Institute Director: Dr. Robert F. Lucas

University of Southern California

4676 Admiralty Way, Suite 1001

Marina del Rey, California, 90292

rflucas@isi.edu, 310-448-9449

DOE/Office of Science Program: Advanced Scientific Computing Research

DOE/Office of Science Program Manager: Dr. Ceren Susut

Collaborating Institutions and PIs:

Institution	Principal Investigator(s)	E-mail Contact	Phone Contact
Argonne Nat. Lab.	Paul Hovland* Sri Hari Krishna Narayanan Stefan Wild	hovland@mcs.anl.gov snarayan@mcs.anl.gov wild@mcs.anl.gov	630-252-6384 630-252-3365 630-252-9948
Lawr. Berkeley Nat. Lab.	Lenny Olike* Samuel Williams	LOlike@lbl.gov SWWilliams@lbl.gov	510-486-6625 510-486-5936
Lawr. Livermore Nat. Lab.	Bronis R. de Supinski* Todd Gamblin Chunhua Leo Liao Kathryn Mohror Daniel J. Quinlan	bronis@llnl.gov tgamblin@llnl.gov liao6@llnl.gov kathryn@llnl.gov dquinlan@llnl.gov	925-422-1062 925-422-9319 925-422-8190 925-423-2997 925-423-2668
Oak Ridge Nat. Lab.	Ed D'Azevedo Philip Roth Patrick H. Worley*	dazevedoef@ornl.gov rothpc@ornl.gov worleyph@ornl.gov	865-576-7925 865-241-1543 865-574-3128
U.C. San Diego	Laura Carrington* Ananta Tiwari	lcarring@sdsc.edu tiwari@sdsc.edu	858-534-5063 858-822-0886
Univ. of Maryland	Jeff Hollingsworth*	hollings@cs.umd.edu	301-405-2708
Univ. of North Carolina	Rob Fowler*	rjf@renci.org	919-445-9670
Univ. of Oregon	Allen D. Malony* Boyana Norris Sameer Shende	malony@cs.uoregon.edu norris@cs.uoregon.edu sameer@cs.uoregon.edu	541-346-4407 541-346-4413 541-346-0850
Univ. Southern Calif.	Jacqueline Chame Pedro Diniz Robert Lucas*	jchame@isi.edu diniz@isi.edu rflucas@isi.edu	310-448-9301 310-448-8246 310-448-9449
Univ. of Tennessee	George Bosilca Jack Dongarra Dan Terpstra*	bosilca@icl.utk.edu dongarra@icl.utk.edu terpstra@icl.utk.edu	865-974-6321 865-974-8295 865-974-0293
Univ. of Texas El Paso	Shirley Moore*	svmoore@utep.edu	915-747-5883
Univ. of Utah	Mary Hall* Manu Shantharam	mhall@cs.utah.edu manu.shantharam@cs.utah.edu	801-585-1039 801-585-1039

* Institutional point of contact.

Contents

Table of Contents	i
Director's Report	1
1 Executive Summary	1
2 Performance Engineering	3
2.1 End-to-end Performance	3
2.2 Performance Tools	5
2.3 Software Integration	6
2.4 Autotuning	7
2.5 Future Plans for Performance Engineering	11
3 Energy	12
3.1 Power Measurements	12
3.2 Energy Efficient HPC Research and Tools	12
3.3 Future Plans for Energy Efficiency	13
4 Resilience	13
4.1 Formal Approaches to System Resilience	14
4.2 Composing resilience techniques	14
4.3 Resilient Application Programming Interfaces	16
4.3.1 Fault mitigation	16
4.3.2 Programmer-guided Source Code Transformations	16
4.3.3 Algorithm-Based Resilience	16
4.3.4 Resilience with Checkpoint/Restart	17
4.4 Automating Resilience	17
4.5 Coarse Grain Resilience	18
4.6 Future Plans for Resilience	18
5 Overall Optimization of Performance, Energy, and Resilience	18
5.1 Optimization Research	18
5.2 Future Plans for Optimization	20
6 Engagement	20
6.1 Application Partnerships	20
6.2 Institute Awareness	22
6.3 Outreach	23
7 Augmentation	23
7.1 Transforming Geant4	23
7.2 Roofline Toolkit	23

8 Management	23
8.1 Organization	24
8.2 SciDAC Application Partnerships	25
8.3 Institute Directors Plan	25
9 Summary	25
10 Institutional Progress Reports	27
10.1 Argonne National Laboratory	27
10.1.1 Optimization	27
10.1.2 Performance modeling	28
10.1.3 Autotuning	28
10.1.4 Application Partnerships	29
10.1.5 Geant4 Supplement	29
10.1.6 ComPASS Supplement	29
10.2 Lawrence Berkeley National Laboratory	31
10.3 Lawrence Livermore National Laboratory	34
10.4 Oak Ridge National Laboratory	36
10.4.1 Performance Engineering	36
10.4.2 Engagement	37
EPSi	37
PSI	37
ACES4BGC	37
PISCEES	37
Multiphysics	37
10.5 University of California San Diego	38
10.5.1 Green Queue – Energy Efficiency in HPC	38
10.5.2 Multi-objective optimization of power, performance and energy	39
10.5.3 Outreach	40
10.6 University of Maryland	41
10.6.1 Integration	41
10.6.2 Multivariate Optimization	41
10.7 University of North Carolina	43
10.7.1 PowerMon2	44
10.7.2 RCRToolkit	44
10.7.3 Memory System Characterization	45
10.7.4 End-to-end performance	46
10.7.5 Application Outreach	46
10.7.6 Emerging SciDAC Engagement	46
10.7.7 SUPER Education and Outreach at UNC	47
10.8 University of Oregon	48
10.8.1 TAU enhancements	48
10.8.2 Autotuning Integration	49
10.8.3 Application Engagements	49
10.9 University of Southern California	51
10.10 University of Tennessee, Knoxville	53
10.10.1 Performance Measurement	53
10.10.2 Energy Measurement	53

10.10.3 Resilience	54
10.11 University of Texas El Paso	56
10.11.1 Performance	56
10.11.2 Energy	57
10.12 University of Utah	58
10.12.1 Enhancements to CodeGen+.	58
10.12.2 Initial analysis of MPAS code.	58
10.12.3 Detailed analysis of QCD code.	58
10.12.4 Analysis of XGC code.	58
10.12.5 Resilience: Current and Future Work.	58
A. References	59

1 Executive Summary

Over the course of the third phase of the Scientific Discovery through Advanced Computation (SciDAC-3) program, 2012–2016, computational scientists working on behalf of the Department of Energy’s Office of Science (DOE SC) will exploit a new generation of petascale computing resources to make previously inaccessible discoveries in a broad range of disciplines including physics, chemistry, and materials science. The computational systems underpinning this work will increase in performance potential from tens to hundreds of PFLOP/s, but in the process will evolve significantly from architectures in 2011, when SciDAC-3 began. Although Moore’s law continues unabated, the end of Dennard scaling has necessitated a fundamental shift in computer architecture focused on power efficiency [9]. To that end, processors are increasingly varied as they strive to satisfy performance, productivity, reliability, and energy efficiency in the face of divergent computational requirements. The diversity among these machines presents a number of challenges to merely porting today’s scientific applications, much less achieving good performance. Thus, by the year 2016, we anticipate that vastly increased scale (e.g., more chips, more cores per chip, wider SIMD) and heterogeneity will exacerbate performance optimization challenges while simultaneously promoting the issues of energy consumption and resilience to the forefront.

To ensure that DOE’s computational scientists can successfully exploit emerging high performance computing (HPC) systems, Advanced Scientific Computing Research (ASCR) funded the SciDAC-3 Institute for Sustained Performance, Energy, and Resilience (SUPER). This is a broadly-based project with expertise in compilers and other system tools, performance engineering, energy management, and resilience. We are following the successful model that we developed in the previous SciDAC-2 Performance Engineering Research Institute (PERI) of leveraging the research investments DOE and others have made and integrating the results to create new capabilities beyond the reach of any one group. To accomplish this, SUPER is organized into multi-institution teams conducting the following research activities, illustrated in Figure 1.1.

1. Performance portability: We are further extending PERI’s performance measurement and autotuning technology to petascale and heterogeneous systems, thus permitting scientists to exploit a wide range of high-end systems from a common code base. Team leader: Mary Hall, Utah.
2. Energy efficiency: Relatively minor code changes can enable significant energy savings in some applications, with little or no impact on performance. We are investigating application-level energy efficiency techniques to help reduce DOE’s energy costs. Team leader: Laura Carrington, UCSD.
3. Resilience: Petascale calculations are pressing the limits of reliability in both hardware and system software. We are exploring strategies to enable petascale applications to be resilient in the face of faults. Team leader: Bronis R. de Supinski, LLNL.
4. Optimization: We are building on tools from the mathematical optimization community to develop strategies that collectively optimize performance, energy efficiency and resilience. This work involves optimization models for different components and the use of multi-objective optimization techniques. Team leader: Paul Hovland, ANL.

SUPER researchers are collaborating with other SciDAC-3 institutes, Scientific Computation Application Partnerships (SAP), as well as the broader DOE computational sciences community. Towards that end, we are explicitly participating in twelve of the eighteen SAPs. These collaborations both focus our research on the real challenges facing petascale scientific computing as well as ensure the broad and immediate impact of the results of our research. We are working with DOE centers and HPC vendors to integrate, deploy, test, and document tools that achieve our objectives, and then actively collaborate with DOE center staff and scientific application teams to address high-priority codes and codes with special needs. Therefore, SUPER

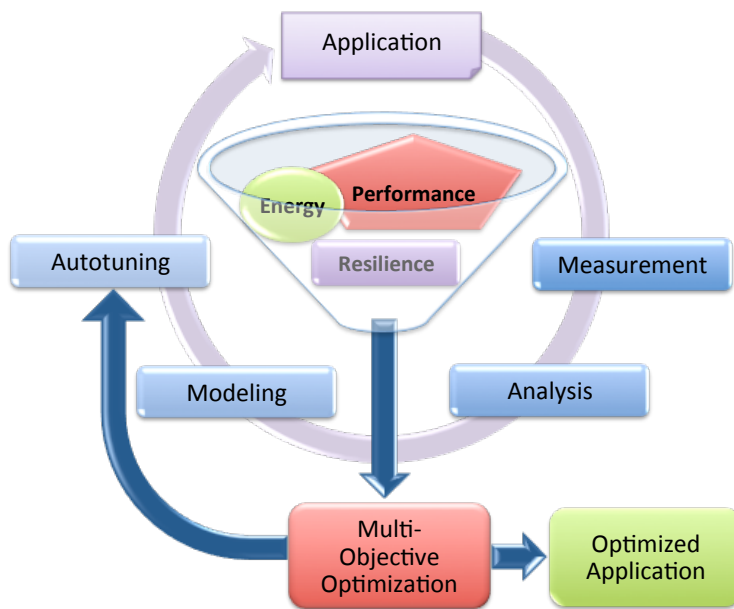


Figure 1.1: SUPER: Optimizing applications for emerging petascale architectures.

teams are also conducting the following engagement and outreach activities.

1. Application engagement: We are providing measurement and analysis tools to help application developers model and optimize their codes and decide whether closer collaboration with SUPER would be useful. We are working with the computing centers to enhance user-level measurement and analysis capabilities and to identify applications for which further performance, energy or resilience enhancement is needed. We are also beginning to work with other SciDAC-3 institutes to analyze and optimize their software and its use in application codes. Team leader: Patrick H. Worley, ORNL.
2. Tool integration: We are creating an integrated tool suite to enable end-to-end application optimization, including automated experiments for performance analysis and autotuning, power-aware instrumentation for energy evaluation, fault detection and vulnerability analysis, and scalable data management and presentation. Team leader: Allen D. Malony, Oregon.

The remainder of this Director's report is organized as follows. First we present the progress of our performance engineering research in section 2, including measurement of performance data, software integration, and automatic tuning. Section 3 presents our progress creating the tools that will allow scientific programmers to adapt to the emerging challenges of energy efficiency. Section 4 presents our progress in creating the tools that will allow scientific programmers to adapt to the emerging challenges of resilience. Section 5 presents the progress of our research towards optimally balancing performance, energy, and resilience. Engagement with others, in particular the new SciDAC-3 SAPs is discussed in section 6. Augmentations of SUPER to collaborate with High Energy Physics scientists on their Geant4 software and with the SciDAC-3 FASTMath institute on a roofline toolkit are discussed in section 7. We discuss how we manage the SUPER institute in section 8. Finally, we summarize and include brief statements of the work each individual institution has contributed to SUPER.

2 Performance Engineering

The SciDAC-2 PERI Institute focused on developing and applying tools to improve the performance of key DOE applications. As the complexity and diversity of modern architectures grows, the need for such tools is even more critical. In the first two years of SUPER, we have been capitalizing on the increased maturity of our tools and the existing team to push our performance analysis and optimization tools to the next level. We have focused our efforts on tool integration, optimizing SciDAC-e applications and enhancing support for heterogeneous systems that include GPUs. This work goes beyond the mostly socket-level optimization in PERI to also consider end-to-end performance.

2.1 End-to-end Performance

Each of the DOE SC computing centers, Argonne Leadership Computing Facility (ALCF), National Energy Research Scientific Computing Center (NERSC), and the Oak Ridge Leadership Computing Facility (OLCF), have tools and infrastructure for instrumenting, collecting, and analyzing performance. Unfortunately, many application groups at the centers do not use these capabilities in a consistent manner, and thus lose many of the advantages of doing so. In response, SUPER has established the feasibility of low overhead, easy adoption, scripting approaches to capturing and archiving end-to-end and holistic performance data for production application runs. Initial experiments targeted the Community Earth System Model (CESM) and the XGC1 plasma micro-turbulence core-edge simulation code running at NERSC and OLCF. We examined the use case of identifying and characterizing performance variability and the role of the node allocation and interference from other jobs in this variability.

Development of this data collection capability continues, including support for the BG/Q system at ALCF and the XC30 system at NERSC, and it is being introduced into additional climate and fusion experiments and codes, for example the XGC0 plasma simulation code. Data also continues to be collected for the existing experiments, providing sufficient volume to apply data mining techniques. The data collection capability is also being used to estimate how much time remains during for a given batch job for XGC0 and XGC1 so as to avoid abnormal termination when time is exhausted unexpectedly.

The multiple streams of performance data and of related meta-data being collected in the above effort are also being merged and archived within the TAUdb performance database, driving further development of TAUdb and associated analysis tools.

The next steps in this effort include implementing similar techniques in additional DOE codes, and demonstrating and promoting the utility of capturing and analyzing end-to-end data on a project-wide basis to center staff and to other center users. For example, we expect this data collection infrastructure will be used by the proposed DOE Earth System Model project for both development and production jobs. For SUPER, these data archives will be important drivers for further research and development, and the code instrumentation will provide hooks for the utilization of more sophisticated tools suitable for performance optimization. The data collection capability will also be extended to capture additional application and system data on both I/O and MPI communication characteristics and performance for production jobs, in support of new performance optimization research.

We note that Argonne is working with ParaTools, Inc. to enable automated performance measurements for applications run on ALCF as a way to capture production performance statistics that can help in end-to-end activities. These efforts are complementary to SUPER and we will look for opportunities for cross-collaboration.

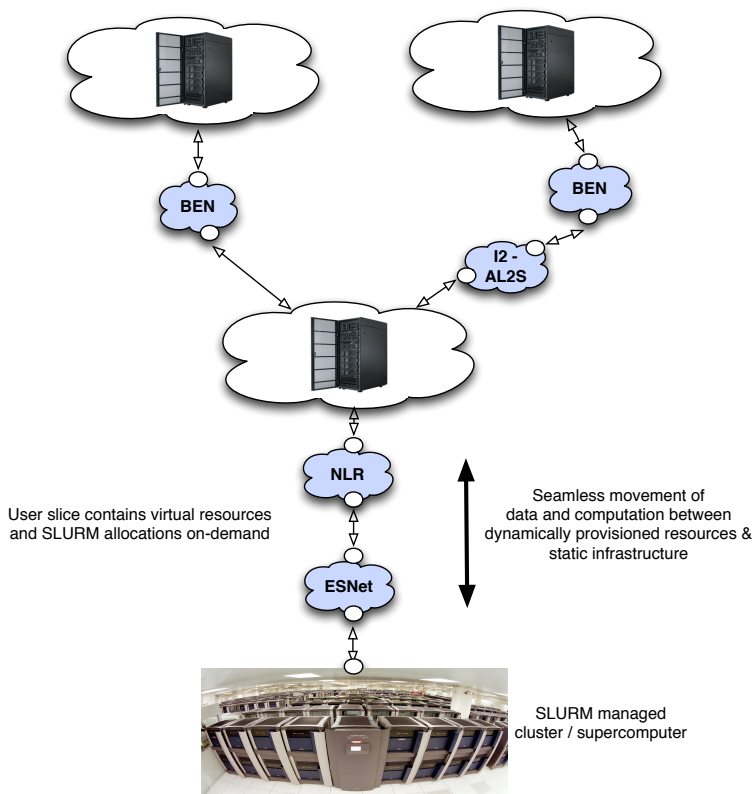


Figure 2.1: Resources allocated to reliably execute large ensembles include supercomputer partitions allocated and managed under SLURM.

Driven by the need for uncertainty quantification, validation requirements, optimization, and other inverse problems, an increasing fraction of scientific workloads takes the form of dynamically re-configurable ensembles and workflows. To run such workloads efficiently requires both that individual job steps run fast and that the computational campaigns run quickly and that they can also automatically and robustly adapt themselves in response to failures of system hardware, system software, file systems, and application (model) failures. SUPER is developing and evaluating an experimental workflow system to run robust distributed campaigns on collections of systems that include DOE leadership systems, Open Science Grid, and local institutional resources, depending on the characteristics of the application. This is based on extensions to Pegasus, Slurm, and other open source packages.

The initial target applications for the above work were chemistry and materials problems used in the search for new solar energy materials and catalysts. Affected projects include the Materials Genome effort and the Solar Fuels EFRC and SciDAC-e projects at the University of North Carolina. In the materials domain, any particular material may turn out to be metallic, an insulator, or a semi-conductor. Without *a priori* knowledge of the results, a simulation set up, *e. g.*, for a metal may fail if the material is non-metallic. This results in a high rate of model failure, which has traditionally been managed by hand or by extensions to the underlying chemistry code. This framework automates the process of reacting to failures at all levels, queuing alternate simulation runs, and (re-)allocating resources to continue. The set of resources allocated to the problem are called a “slice” and include compute nodes, storage, and reserved network bandwidth. During the current reporting period, the framework was extended to manage supercomputer partitions allocated using SLURM as part of a distributed slice (Figure 2.1. In addition to providing a glide-in scheduling capability, the ensemble framework interacts with SLURM system health monitoring facilities

to report on errors and to re-allocate resources. This work was initially demonstrated in the RENCIBOOTH at SC12 and will be shown at SC13.

2.2 Performance Tools

A core effort in SUPER's performance engineering is continued improvement of the tools used for instrumentation, measurement, analysis, and modeling. SUPER is blessed with project partners who have significant research and development strengths across all of these aspects and represent important leading tools in the HPC community.

One important component in SUPER's performance engineering effort is the support for retaining performance data produced from experiments and production runs. While different performance tools may generate performance measurement results, having a means to hold on to the information in a repository that provides common access helps to serve SUPER's joint analysis and integration objectives. Our TAUdb effort has continued to evolve to address the needs of the SUPER project. The SUPER TAUdb database currently hosts several hundred performance profiles from application engagements. These profiles include initial benchmarking as well as parametric studies to evaluate code modifications. The TAUdb database has been integrated into the Orio autotuning framework for use as a repository for both exhaustive and guided tuning searches.

In the areas of performance instrumentation and measurement, SUPER work has continued to progress. OpenMP measurement for selected runtimes has been further improved in TAU with improved support for the OpenMP Runtime Collector API [36]. To broaden the impact of this work, we have used the TAU library generator to construct a wrapper for the GNU OpenMP library (GOMP), and added event callback and state support for the GNU OpenMP runtime. With this support, even uninstrumented, unmodified executables compiled with gcc can generate detailed OpenMP performance information through the use of the callback mechanism or states queried during samples. TAU has also added support for the proposed "OpenMP Tools Application Programming Interfaces for Performance Analysis and Debugging" (OMPT) [17], which is an extension of the previous API. We have contributed to a community effort to add OMPT support within the recently open-sourced Intel runtime.

The most recent TAU release (2.22.2) included updated support for DyninstAPI v8.1i [49], which provides improved support for static rewriting, and full integration for static binaries in progress. The latest DynInstAPI includes support for loop level instrumentation, and allows for selective instrumentation at the routine and loop level. The latest release of the Program Database Toolkit (PDT) [50] also included support for PEBIL for OpenMP instrumentation, MAQAO, and PAPI 5.2 with support for power measurements and Intel Xeon Phi architecture, as described in Section 2.3. TAU version includes support for sampling in parallel programs using OpenMP or Pthread APIs, an updated TAUdb database, and ParaProf with support for editing TAUdb metadata, better support for displaying context atforms and compilers including the Edison Cray XC30 platform at NERSC. It also or a new API call for tracking power using the PAPI RAPL interface. Power measure periodically (typically every 10 seconds) when an interrupt timer is triggerd.specify the inter-interrupt interval and enable or disable the power monitoring. We augmented the mpiP lightweight MPI profiling tool to collect more detailed data about point-to-point and collective communication operations to help users better understand an application's logical communication topology [53]. We also began developing a tool that tracks how (and how much) a value influences computation that occurs later during a program run [43]. Finally, GPU performance measurement support in TAU for heterogeneous environments continues to track the CUDA/CUPTI technology. The goal in each of these cases is to provide more robust means of performance observation.

While memory, interconnect, and I/O performance have been improving, they have not kept pace with the rapid increase in the number and speed of the cores on each chip. Furthermore, with the end of Dennard scaling, the speed at which cores can be run in practice is constrained by the energy budget of the package, not the design and silicon implementation of the cores themselves. For example, if only one or two cores per package are active they can be run at a speed up to 40 percent faster than if all are active. “TurboBoost” or “computational sprinting” capability under hardware control can thus have a tremendous performance advantage, but it makes performance issues much harder to understand. To meet these challenges, particularly in the area simultaneous optimization of both performance and energy, SUPER is working on performance tool extensions for tracking these dynamic behaviors. These efforts necessitate real time monitoring of hardware in the “uncore” part of the chips. In contrast to past work that has focused on attributing costs to particular program constructs in individual threads, this effort will require new models and analyses suitable for system-wide performance characterization. Prototype software has been developed for Intel Xeon X86_64, Intel MIC, and AMD chips. The node-wide monitoring and modeling of uncore components is currently being used by SUPER collaborators at UNC, UO, and UTK, and is being deployed to the XStack XPRESS project. It is also being used by DoD-sponsored projects.

In addition to the tools enhancements, the SUPER project is committed to deploying performance capabilities to DOE HPC platforms. In addition, SUPER is open to the broader performance tool community and looks for opportunities for interactions and technology interoperation.

2.3 Software Integration

Performance engineering involves both the creation of methodologies and tools, as well as the integration of these to enable their more productive and powerful application. SUPER is pursuing synergistic opportunities to integrate across its performance engineering software activities. This is occurring in the end-to-end work in a major way through the use of different performance measurement and analysis tools, and the collection of performance information across applications and platforms for integrative study.

The TAU and the ROSE teams worked together to integrate the latest version of the ROSE compiler [28] into the Program Database Toolkit (PDT) [50]. The EDG parser for C and C++ as well as the ROSE software were ported to IBM BlueGene/Q (BGQ), Linux x86_64, and Mac OS X platforms. The integrated parser and IL analyzer component emits Program Database (PDB) records for program constructs such as routines, basic blocks, types of classes, structs, arguments passed to each routine, return data types, template parameters, types of statements, header file inclusion trees, and macro definitions. Previously, our teams had integrated an earlier version of the EDG parser (v3.3) in PDT. EDG v4.4 adds better support for C++ 2011, C99, UPC, enhanced support for template parsing, and assembly statements. Correspondingly, the PDB record creation component was updated to support the new features of the languages that are supported. Porting the parsers to the three platforms required creating static executables on BGQ and Linux x86_64 systems. On Mac OS X, we isolated the libstdc++ dependency of the parsers and created dynamic executables that can execute on the older Lion as well as the newer Mountain Lion systems. Our team created a test suite for validating the port and released this component as part of PDT v3.19p1. An updated version of TAU v2.22.3b4 [51] was released to use PDT 3.19p1 and it adds support for cascading failures that allows the instrumentation process to use a different parser if an older version does not parse the source code. It also includes better support for parsing standard C++ header files in PDT. The PDT release also includes support for PEBIL binary rewriting [26]. PEBIL provides loop-level instrumentation, and supports GNU and Intel compilers. TAU includes support for PAPI [52] hardware counters on the Intel Xeon Phi architecture.

In collaboration with researchers at Texas Advanced Computing Center (TACC), we have been working

on integrating the PerfExpert [11] and MACPO [41] tools developed at TACC with the SUPER auto-tuning framework. One accomplishment has been to automate the code transformations suggested by PerfExpert through use of the ROSE source-to-source compiler.

There has also been work on software integration in support of autotuning. SUPER investigators have implemented TAU within the CHiLL framework for performance measurement of variant experiments generated from CHiLL code transformations. This effort involved the integration of the ROSE compiler system for code outlining and Active Harmony for conducting autotuning searches. The integrated system automated the performance experiments, the storing of performance data in a TAUdb database, and the query of results from the database during search optimization. Furthermore, this process could derive optimized code variants for different parameter combinations and re-engage ROSE for automatic source specialization. Similarly, we have tested the integration of TAU measurement, database, and data mining support with the Orio framework for GPU code optimization. Figure 2.2 shows the data flow between the framework components. The integration results were presented at the DOE SciDAC Fall PI meeting [32]. Additional integration of CHiLL with Active Harmony, the ROSE compiler and the PBound performance modeling software is facilitating more automated autotuning, as discussed below.

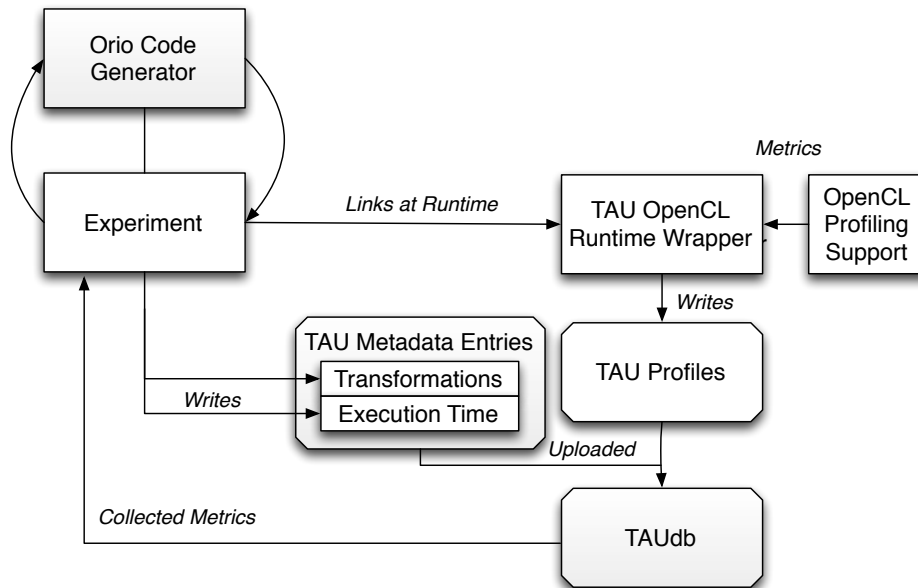


Figure 2.2: Data flow between Orio, TAU and TAUdb.

2.4 Autotuning

As a group, the SUPER researchers working on autotuning have been examining key computations from two SciDAC applications: (1) the PUSHE computation from XGC1; and, (2) a DSLash operator example extracted from the USQCD MILC code containing a conjugant gradient solver and hand-optimized SU(3) primitives. (These are detailed elsewhere in this report.) In XGC, we have gathered significant performance data using TAU, and collected both a CPU and GPU implementation of the code. Further work using the SUPER optimization tools will require rewriting this computation significantly.

For USQCD, we have also collected performance data, and in addition, SUPER researchers have rewritten the computation at a higher level that is more amenable to optimization.

```

for (it1=0; it1<4; it1++) {
  for (it2=0; it2<3; it2++) {
    for (it3=0; it3<3; it3++) {
      t_c_real[it3] += t_a_real[it1][it3][it2] * t_b_real[it1][it2];
      t_c_imag[it3] += t_a_real[it1][it3][it2] * t_b_imag[it1][it2];
      t_c_real[it3] -= t_a_imag[it1][it3][it2] * t_b_imag[it1][it2];
      t_c_imag[it3] += t_a_imag[it1][it3][it2] * t_b_real[it1][it2];
    }
  }
}

```

Figure 2.3: QCD computation reorganized as loop nest.

```

for (i=0; i<sites_per_node; i++) {
  ...
  for (it3=0; it3<3; it3++) {
    for (it1=0; it1<4; it1++) {
      for (it2=0; it2<3; it2++) {
        t_c_real[it3] += t_A_real[4*i+it1][it3][it2] * t_b_real[it1][it2];
        t_c_imag[it3] += t_A_real[4*i+it1][it3][it2] * t_b_imag[it1][it2];
        t_c_real[it3] -= t_A_imag[4*i+it1][it3][it2] * t_b_imag[it1][it2];
        t_c_imag[it3] += t_A_imag[4*i+it1][it3][it2] * t_b_real[it1][it2];
      }
    }
  }
  ...
}

```

Figure 2.4: Inlining mul_su3_mat_vec_sum_4dir.

```

* METHODOLOGY
--- Hand tuning [Jacque]
--- CHiLL [Amit]
* Remaining issues with inlining and SSE [Jacque]

* NEW IDEAS THAT HOLD PROMISE
--- Fusion of functions
--- Data layout exploration

```

To make the code more amenable to optimization tools, we have manually rewritten the computation in function `mul_su3_mat_vec_sum_4dir()`.

We replaced the array of pointers to matrices of type `su3_matrix` with two 3-D arrays. One 3-D array keeps the real components of elements of four 3×3 matrices, and the other 3-D array keeps the imaginary components. The four input vectors `b0`, `b1`, `b2` and `b3`, are copied to two 2-D arrays of size 4×3 . One 2-D array keeps the real components of the 4 vectors of size 3, while the other keeps the imaginary components.

With the new data structures in place, the computation is reorganized as a three-deep loop nest as shown in Figure 2.3. This reorganization makes the computation more amenable to analysis and optimization tools. In the loop nest shown in Figure 2.3, the array indices are functions of the loop indexing variables, which allows compilers and optimization tools to analyse array accesses and identify optimization opportunities. For example, a tool can determine that it is safe to permute all loops and automatically change the loop order to `it3`, `it1`, `it2` to exploit the temporal reuse of array references `t_c_real` and `t_c_imag` in loop `it3` and the spatial reuse of `t_a_real`, `t_a_imag`, `t_b_real` and `t_b_imag` in loop `it2`.

To expose more optimization opportunities we manually inlined the loop in Figure 2.3. As a result the caller function has 4 loop nests of depth 4, such as the one shown in Figure 2.4. We also replaced the array of pointers to `su3_matrix` matrices with two 3-D arrays of size $(\text{sites_per_node} \times 4) \times 3 \times 3$ each, `t_A_real` and `t_A_imag`.

Additionally, recent engagement with the BES funded research to optimize NWChem performance, with

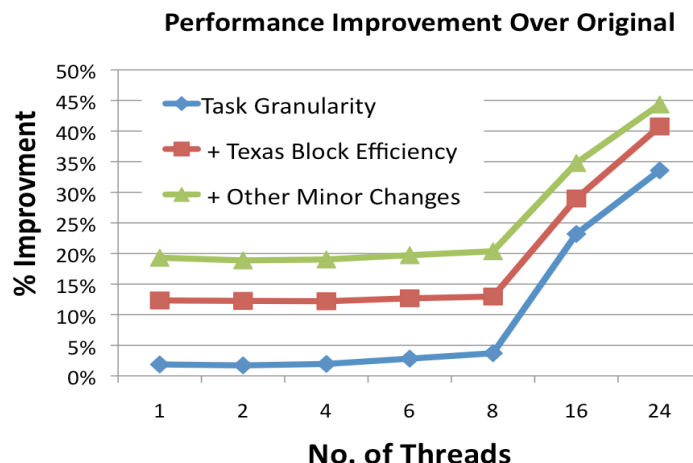


Figure 2.5: NWChem optimization and autotuning on one Hopper node for c20h40 configuration, showing a total speedup of up to 44%.

the high-level scientific goal of developing electron-correlated methods for excited state structure and dynamics in photoactivated systems. Preliminary work has focused on improving the performance of the TEXAS integral package to compute the two-electron repulsion integral, which is of considerable importance to NWchem, due to its computational complexity and large number of integrals. Our performance improvement focused on two approaches: balancing the load across multiple threads and reducing the wall-clock computing time of each individual atom quadruplet integral.

In the TEXAS integral package, the large number of atom quadruplet integrals is assigned to each individual thread via task stealing. One important parameter of task stealing is the task granularity, whose ideal value is difficult to choose. Due to the different types of the atom quadruplet, the computing time needed for each quadruplet integral varies widely. This further complicates the choice of the granularity. By profiling, we discovered that the original task granularity chosen by TEXAS causes load imbalance when more than 16 threads are active. We tested with different methods and found that with finer granularity the load imbalance problem can be solved. Experimental results on the Cray XE6 Hopper system showed that by improving the load balance, the performance can be improved by 23% and 34% for 16- and 24-cores, respectively (see Figure 2.5 *Task Granularity*).

We additionally identified several time-intensive loops and applied auto-tuning techniques to improve their performance. The optimization techniques include loop unrolling, blocking, pre-fetching, and vectorization. Additionally, we discovered potential performance enhancements within several functions, including the sorting operations. Finally, we discovered a more efficient blocking structure, allowing improved reuse of intermediate results and overall runtime improvement (Figure 2.5 *Texas Block Efficiency*). Combining our optimization schemes resulted in a total speedup of 19-44% for various configurations as shown in Figure 2.5.

SUPER researchers have also begun applying autotuning to MPI communication. In particular, we have been looking at tuning configuration parameters within implementations of MPI. Parameters such as the threshold between small, medium, and large messages can have a significant impact on the performance of a program. A second opportunity is to tune the frequency of calls into the MPI library to poll for the completion of asynchronous communication. Calling into the MPI library too often can create excessive overhead. However, calling into the MPI library not frequently enough can also slow down execution by limiting the rate at which asynchronous activities occur. With MPI 3.0, there are now asynchronous collec-

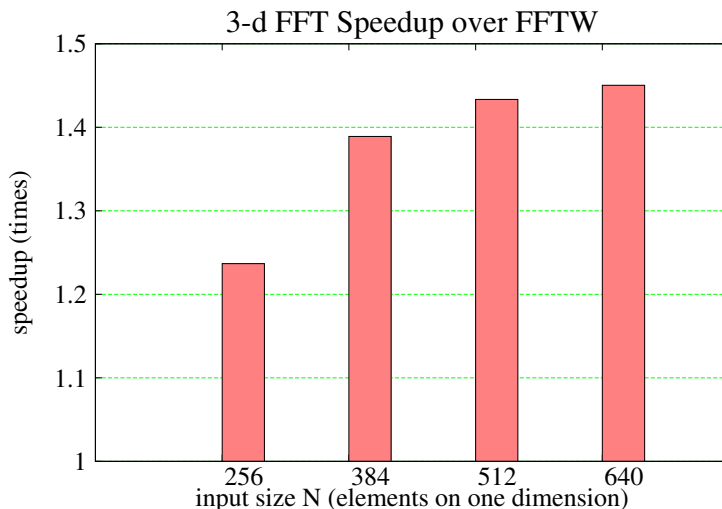


Figure 2.6: Performance gain of communication auto-tuned FFT vs. FFTW.

tive communication operations in addition to asynchronous point to point communication. To explore the ability of auto-tuning to help with these communication parameters, we have been working to develop an auto-tuned version of a 3-D FFT kernel that uses asynchronous collective communication. Figure 2.6 shows the results of using our new FFT kernel vs. the popular FFTW auto-tuned FFT library. Using our new communication optimization techniques we are able to improve performance 25% to 45% depending on the size of the input. Recently, we have been extending this work to support 2-d decomposition of the data. This will allow for better strong scaling. We now have achieved similar results to our 1-d decomposition for matrices that are the same size in all dimensions, but are working to improve performance for arbitrary sized inputs.

SUPER researchers have reached a significant milestone with the CHiLL and CUDA-CHiLL loop transformation and code generation frameworks. We have integrated these tools with ROSE, and the first release of the integrated system was completed in November 2012. Prior to the release, and since then, we have added significant enhancements to CHiLL's capabilities to prevent users from incorrectly transforming the code, improve the performance and the generated code and increase usability of the tools. We are planning a subsequent release in Summer 2013. We are extending the framework to support non-affine computations more broadly, computations requiring run-time support to ensure correctness and profitability of transformations. In addition, we are evaluating extensions to user specification of search space pruning in the presence of code variants. We are also working to extend the reuse analysis capability in PBound.

In collaboration with Martin Head-Gordon's SciDAC effort, our researchers have focused on performance instrumentation and analysis, performance optimization, and developing scalable implementations for leadership-class supercomputers for each of the project's four optimization and parallelization efforts: libtensor, RAS-SF, linear/eigen solvers, and two-electron integrals. To date, our work has focused on analysis, optimization, and planning for a distributed backend for libtensor. After significant preliminary interaction with Krylov and Epifanovsky, we were able to begin instrumenting the code in order to intercept all BLAS calls recording the NUMA node containing each array/vector argument and executing thread, as well as corresponding sizes and access strides. It is our belief that this information will allow us to model performance and identify routines that underperform due to a lack of NUMA locality (today, multisocket SMP's are invariably non-uniform memory access) or memory access pattern. Routines or thread schedulers can be rewritten to mitigate these effects and improve performance. Additionally, this approach may identify patterns of BLAS calls that could be aggregated in order to maximize performance on today's multi-

core architectures. A number of exploratory auto-tuning concepts are being developed to deal with NUMA, complex memory access patterns, and proper exploitation of thread-parallelism.

Although the aforementioned optimization effort will target the current common-use scenario, it is ultimately single-node out-of-core, and thus not appropriate for today's disk-less leadership-class supercomputers. As such, we have had a series of three-way discussions with Edgar Solomonik (UCB) on his Cyclops Tensor Framework (CTF) [47]. CTF is an MPI-based distributed tensor contraction framework. It subsumes full control of tensor creation, redistribution (using a cyclic distribution of elements), and contractions. The cyclic aspect results in a regularized computational pattern that allows efficient exploitation of large-scale supercomputers. It has been demonstrated in standalone CCSD benchmarks and provides excellent scalability and superior performance at large scale. As such, we are working with USC and UCB on the creation of a second (CTF-based, user-selectable) backend for libtensor that would allow users to efficiently target either single-node SMPs or leadership computing resources. CTF performance is heavily dependent on problem configuration and collective performance. BLAS performance is typically a minority component. We will thus focus any auto-tuning efforts on local data marshaling and approaches to MPI communication.

As part of the Orio autotuning infrastructure, we extended CUDA code generation and autotuning of sparse matrix and vector kernels used in the PETSc toolkit [15, 34]. The GPU-specific optimization parameters include numbers of threads, blocks, and streams, L1 cache size (for newer NVIDIA chips), compiler flags, and cascading reductions. Orio generates both the host code and the device function. The GPU autotuning was incorporated into a structured grid PETSc matrix type and a vector type. In the last few months, we also implemented preconditioners that can operate with the autotuned kernels, including a polynomial preconditioner and an ILU preconditioner, with expected improvements in convergence (10-20x over no preconditioning). We also implemented an initial MPI code generation and tuning module in Orio for optimizing MPI implementation parameters and communication/computation overlap.

2.5 Future Plans for Performance Engineering

Say something that correlates with institutional plans.

First-person tools such as PAPI and software layered on top of it use core-local performance counters and attribute cost metrics to specific cores, processes, and threads for the purpose of analysis and tuning. Third-person tools such as RCRTToolkit focus on performance counters for chip-wide resources such as memory bandwidth, shared caches, and power/energy for the purpose of understanding aggregate behavior and to provide feedback to software layers that can institute run-time adaptation policies. SUPER researchers are beginning to explore methods for the integrated of these tools and models. An early experiment used RCRTToolkit memory utilization information in combination with HPCToolkit call stack profiling to help differentiate loops, similar in structure, that are or are not bandwidth limited in the LQCD Chroma code [35].

Another area being explored is the issue of providing a controlled environment for benchmarking and tuning of programs on large shared clusters, especially for auto-tuning for combinations of energy and performance. Early experiments [40] with running benchmarks repeatedly highlight substantial differences in energy consumption due to increases in parasitic currents between a relatively cool system that starts in a quiescent state and one that is running a full operating temperature. Furthermore, the temperature differences between chips based on intrinsic differences in fabrication, or on the cooling available at different physical locations add additional sources of variation. Performance variability from run to run or between chips is also measurable, but is somewhat less. If TurboBoost is enabled, the potential differences are much greater. Experimental frameworks for code optimization need to either control this variability or compensate for it. Run time adaptive methods may be able to exploit this variation. The sampling interval for power and

energy measurements puts a lower bound on the granularity of loops that can be isolated for optimization without introducing sampling errors.

The end-to-end data capture capability will be leveraged in a new effort focusing on I/O, initially examining how to correlate an application's poor I/O performance or I/O performance variability with other activity on the system while the application was running. We will begin by characterizing the I/O behavior of the MPAS-ocean model and the CESM and XGC1 running on DOE Leadership Computing Facility systems. Later, we will use our I/O characterization results and data collection infrastructure to research techniques for runtime scheduling of I/O so as to control the negative impact of contention for I/O and interconnect resources.

The end-to-end data capture capability will also be leveraged in a new effort focusing on communication overhead. We will investigate techniques for using offline MPI communication characterization and online data on allocated nodes to determine a mapping that minimizes the MPI communication overhead, again using CESM and XGC1 as case studies.

3 Energy

SUPER's Energy thrust is charged with understanding how computation and communication patterns affect the overall energy and/or power requirements of HPC applications, and leveraging this understanding to design software- and hardware-aware optimization techniques that either reduce DOE's HPC energy footprint or compute within a power budget. Energy and power efficient HPC requires controlling how much power is committed to a task and understanding what benefit can be expected in return for completing that task. In many cases this presents the opportunity to make informed tradeoffs between performance and power. Towards that end goal of developing energy-efficient HPC, two complementary sub-thrusts have emerged within the Energy thrust: research on power measurement devices and the software solutions that provide fine-grained access to the power measurements; and energy-efficiency research that utilizes the first two to develop system-level optimization strategies. We highlight recent accomplishments in the two research themes.

3.1 Power Measurements

The PAPI group at UTK continues to work with RENC1 and others within SUPER to explore ways to produce and distribute RENC1's PowerMon2 card for high resolution in-band power and energy measurement. In addition to MIC performance counter support, the PAPI 5.1 release also provides access to an on-board power sensor on the MIC card itself. This allows measurement of current and voltage (and computed power) for various subsystems on the MIC card at roughly 50 msec resolution.

As a result of early conversations with our team, IBM engineers have developed a prototype high speed API for Blue Gene Q, called EMON2, which provides integrated power and energy measurements at the node level. We are working with IBM and LLNL to develop a PAPI interface for this API.

3.2 Energy Efficient HPC Research and Tools

The SUPER Energy thrust continues to research new methodologies to reduce the energy consumption of large scale HPC systems and implement these in tools that enable HPC systems to realize these savings. Along these lines we highlight two such efforts from this past year. The Resource Centric Reflection (RCR) tool suite [14] and the Green Queue Framework [39, 48].

The RCR tool suite is designed for monitoring off-core hardware performance counters and other node-

wide metrics and for computing real time models based on those metrics. In addition to performance metrics, RCRTTool is used for monitoring temperature and power consumption to guide real time decisions regarding clock modulation and power states, as well as to guide fine grain thread/task scheduling. At UNC, RCR has been installed on Intel Sandy Bridge and Ivy Bridge E5 clusters and is being used for characterizing the performance of a variety of codes, primarily in computational chemistry and coastal ocean circulation (storm surge) codes.

The primary symptom of the end of Denard scaling is that supply voltages to CMOS circuits are near their practical minimum. The impact of this is that the use of dynamic voltage and frequency scaling (DVFS) to control power states of a chip has little opportunity to vary the voltage, thus limiting the effectiveness of the technique. At UNC we have been using "clock modulation", a mechanism that filters out a fraction of the clock pulses reaching each core. In contrast to DVFS, clock modulation can be done very quickly on a per core basis by setting a machine- dependent mask register. This is being used by a fine grain thread scheduler that uses information from RCRTTool to guide its decisions.

Experiments on energy variability are discussed above in section [2.5](#).

SUPER also continues to develop the Green Queue framework which utilizes both application- and machine-specific details (or characterizations) to optimize large-scale parallel applications for energy efficiency. The Green Queue framework consists of methodologies that accurately identify customized hardware settings for different computational phases of scientific applications that reduce the energy required to run the applications to completion. One can view these methodologies as dynamically adjusting the hardware to *exactly* match the demands an application puts on the hardware; methodologies that can greatly help address the power wall problem faced by the HPC community on our path to the exascale. We evaluated an initial prototype of the Green Queue framework [[39](#), [48](#)] by deploying it to optimize large scale codes of high importance to the DOE (e.g., MILC, LAMMPS and GTC). The results were encouraging – for example, energy required to run LAMMPS on 1024 cores of Gordon (a Sandybridge-based supercomputer at SDSC) saw a reduction of 32% when optimized by the Green Queue framework.

In addition to continuing to evolve and engineer the Green Queue prototype, we have spent a majority of time this year on the following two key components of the framework – 1. accurate models that predict how changes in application workload affects system level power draw and 2. accurate models that predict the sensitivity of an application's execution time and energy behavior when subjected to different hardware settings. These models utilize workload characteristics that affect power draw and performance responses of a system as predictors. The initial prototype relies on measured performance sensitivity analysis, which can limit the prototype's usability because the measured sensitivity is tied directly to the settings (e.g., input data) under which the analysis was performed. In order to fully automate the process of deriving customized hardware settings for different application phases and to enhance its applicability, Green Queue must rely *only* on the models and developing a unified methodology that enables power, performance and energy modeling is the current focus of our work. Details of the models and their accuracy can be found in Section [10.5](#).

3.3 Future Plans for Energy Efficiency

Say something that correlates with institutional plans.

4 Resilience

During this review period, SUPER researchers have developed new formal methods for assessing the resilience of particular regions of code and particular algorithms using fault injection. We have also devel-

oped a novel fault injector that can automate the process of assessing the resilience of particular areas of code. To leverage the results of such campaigns, we have developed programmer-assisted annotations that mark areas of the code with low resilience, and we have begun to automate compiler techniques that can be used to armor regions of the code that are so annotated. We have also made significant strides in improving checkpoint/restart for practical usage at large scales. Finally, we have conducted a detailed study of the overhead and resilience of resilience-enhancing compiler transformations when used independently and together. Together, these further our progress towards a comprehensive, automatic tool set for vulnerability assessment and mitigation.

4.1 Formal Approaches to System Resilience

Formal approaches to resilience are being investigated with a view to build error detectors that are incisive and yet incur low overheads. Our current explorations in this area are in the integer space (non floating-point). This problem-space is important to address, given that many algorithms (*e.g.*, graph algorithms) are inherently integer computation oriented. Many approaches focus on resilience in the space of floating-point computations; these works assume that the integer space can be covered using TMR methods. Our thrust is to avoid the wholesale use of TMR methods, and to arrive at assertion-based checkers that instead reliably detect errors and trigger recomputation.

Our preliminary explorations resulted in the construction and release of the Kontrollable Utah LLVM Fault Injector (KULFI) tool [44]. This tool offers a versatile combination of fault injection options. Using KULFI, studies are underway to understand the behavior of closely related families of algorithms. The targets of study include sorting algorithms, and will soon be expanded to various relaxation methods as well as graph algorithms. Preliminary results from these studies suggest that some of the sorting algorithms are inherently more resilient than the others, as evidenced by the reduced number of silent data corruptions.

In order to explain the differences in the number of SDCs that were empirically observed in the above KULFI-based study, we are studying the *predicate state transition diagrams* associated with these sorting programs. Given a collection of predicates $Pred = \langle p_1, \dots, p_N \rangle$ whose truth values sharply characterize the executions of a given sequential program P , one can define the *abstract states* of P by recording the truth values of the predicates as defined by $Pred$ across program execution steps (at present, program execution steps are being obtained through concrete execution). In our approach, we at present include many of the salient control-flow predicates of P in $Pred$. In our future efforts, we plan to also include interesting invariants of the program generated using state-of-the-art invariant generation techniques.

Predicate state transition diagrams of programs constructed with and without faults indicate that control-flow tracking using a collection of predicates and invariants may lead to inexpensive fault detection. Our future research aims for a precise characterization of the cost of synthesizing these predicates using TMR-based code, and the amount of error detection enhancement offered by well-constructed predicates – as opposed to error detectors that are obtained in a more *ad hoc* manner. Our preliminary results are encouraging, and are being written up for submission.

4.2 Composing resilience techniques

We are working towards a comprehensive algorithmic resilience approach that considers a range of techniques such as the replication of key variables, detection of memory violations and localized rollback on detected errors. We are also investigating techniques to configure these resilience methods to achieve the best performance given user-defined accuracy and confidence targets.

Building on our prior work with AMG, CG, GEMM, and LU factorization, we have demonstrated our

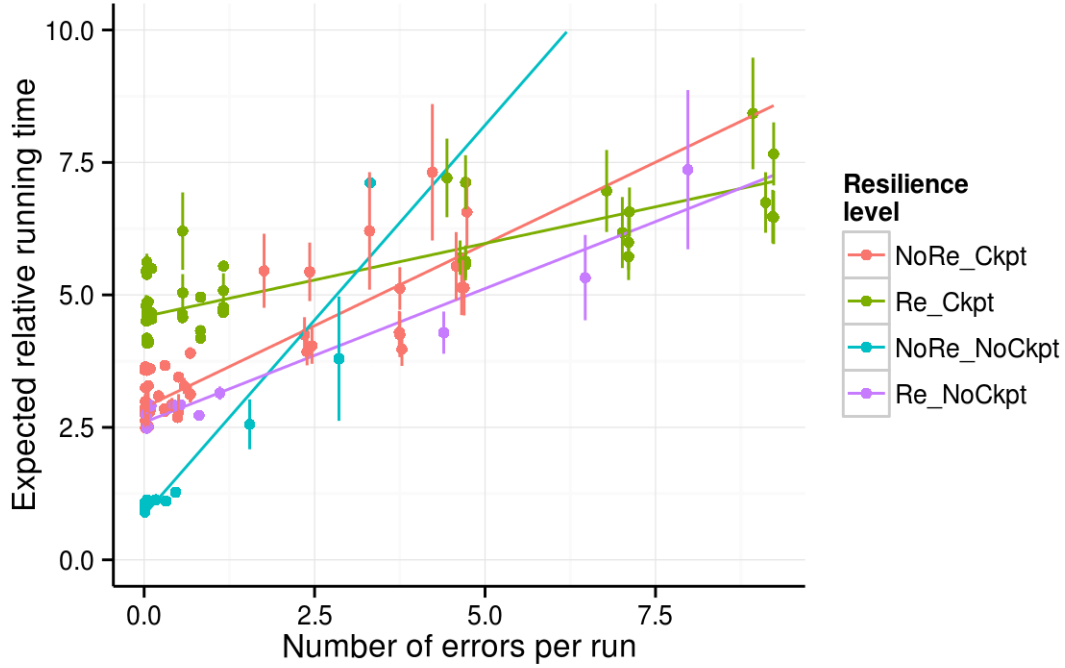


Figure 4.1: Performance/resilience tradeoff for various resilience techniques.

approach on new types of numerical applications: the Alternating Direction Method of Multipliers (ADMM) for Lasso problems, the Hattrick gravitational simulation and the Digital Room Correction (DRC) acoustic correction application. We show how these three applications can be comprehensively protected from soft faults by adding three different resilience mechanisms to the routines where they spend the most time: algorithmic error checks, replication of critical data structures and checkpoint-restart of individual routines. Figure 4.1 illustrates a particular case where the optimal combination of resilience techniques depends on the errors the application suffers per second, thus showing how important is to select the optimal configuration of resilience techniques.

We have demonstrated via fault injection experiments that although none of these techniques can individually protect applications from soft faults, they are highly effective when used together. Further, we have demonstrated how these techniques can be configured to offer the right level of protection at any given fault rate or input type. Our work has identified knobs that can be used to tune the statistical confidence level with which a given overhead or accuracy target will be met, and using this we can produce a wide range of resilience strategies, from low-overhead, lightweight transformations to more aggressive heavyweight approaches.

When using rollback recovery techniques to tolerate soft errors, the period of checkpointing is not the only optimization parameter to consider: if algorithmic checkers are not available, the checkpointed data cannot be certified, and all checkpoints may include data corrupted by a silent error, thus making the application non-recoverable by rollback recovery. This risk can be mitigated by keeping multiple checkpoints in time. The number of checkpoints kept, and the period between two checkpoints impacts both the risk and the efficiency of the overall solution. We have built a performance model ([1]) and proposed a corresponding optimization for this case. On the other hand, if algorithmic checkers, capable of internally validating the data are available, data can be validated before checkpointing, thus ensuring that some checkpoints will

always allow a full valid recovery. But in this case, the number and frequency of validations, as well as the number and frequency of non-validated checkpoints, are all potential optimization parameters. The model proposed allows to consider this case, and to optimize the efficiency, based on the relative costs of validation and checkpointing, and the platform characteristics.

4.3 Resilient Application Programming Interfaces

We are continuing our work towards linguistic mechanisms that allow developers to indicate which sections of their applications code and data structures are tolerant to memory errors. We have also begun to integrate these techniques with compiler transformations. Additionally, we are improving the state of the art with our checkpoint/restart library and interface.

4.3.1 Fault mitigation

SUPER researchers have expanded their previously defined set of resiliency annotations to also include a mitigation mechanism, rather than simply detecting and ignoring the presence of errors. As an example, the developer can now associated with a specific code section a recovery function to be invoked when the error conditions are met.

The new set of annotation can now be coupled to a compiler tool such as ROSE to implement the corresponding source-to-source transformations (as described in point below). At the time of this writing we have experimented with a manual implementation of these annotations in a small set of kernel codes and will present experimental results of its in an upcoming report.

4.3.2 Programmer-guided Source Code Transformations

We support some coupling of our annotation-based approach with compiler-automated resilience strategies. The new set of annotations from USC/ISI can now be coupled to a compiler tool such as ROSE to implement the source-to-source resiliency transformations. We have added an annotation-guided software triplication transformation to ROSE, and our technique currently supports variables with pointer types. We add a `robust` or `critical` pragma annotation to the variable declaration to tell the compiler to perform the transformation. We have validated this specific transformation for a the AMG code mentioned above.

In addition to transformations that are geared towards numeric scientific codes, we have begun to explore transformations based on the software duplication and triplication in the context of discrete heap-allocated and array-allocated graph computations. These involve the replication of selected pointer fields for detection and recovery of data errors as well as the use of storage alignment constraints to aid in the recovery in the presence of duplication only. These newer techniques are still in an evaluation phase. We expect to report on their effectiveness for a selected set of graph computations (*e.g.*, Breath-First-Search and Depth-First-Search traversals) in an upcoming report.

4.3.3 Algorithm-Based Resilience

In addition to automatic and compiler-based approaches, we have extended the set of algorithm-specific approaches, providing new checkers to validate internally the data consistency, and detect silent errors, in larger classes of applications. To the traditional set of one-sided factorizations (LU, LLT, QR), and solves (CG), we added algorithm-based fault tolerance for the two-sided Hessenberg Reduction. Using this approach, we implemented a Hessenberg Reduction [23], that tolerates the loss of data due to permanent crashes in a distributed system. We are extending this work to the case of silent errors (memory corruptions) in distributed and shared memory systems.

4.3.4 Resilience with Checkpoint/Restart

We are improving the state of the art for checkpoint/restart methods. Traditionally, applications simply write checkpoint files to a stable store such as a parallel file system and then read them back in after a failure for recovery. While this is straight-forward, it is not scalable, and can incur overhead of tens of minutes while an application blocks waiting for the checkpoint writes to complete. In prior work, we developed the Scalable Checkpoint/Restart Library (SCR) [?]. SCR improves checkpointing performance because it caches checkpoints on in-system storage, such as RAM disk on the compute nodes, and moves only a select few to the parallel file system. Writing and recovering with checkpoints on in-system storage is scalable. In our experiments, the bandwidth of writing cached checkpoints exceeded that of writing to the parallel file system by up to $1000\times$.

Although SCR has improved checkpoint/restart performance, there is still high overhead when the selected checkpoints are written to the parallel file system to defend against catastrophic failures. Thus, our research in this area focuses on reducing this source of overhead. We are exploring asynchronous checkpoint transfer, checkpoint compression, and improvements to in-system storage. Our checkpoint compression framework, mcrEngine, uses semantic information in checkpoint files written in commonly used file formats, such as NetCDF and HDF5, to merge similar data from checkpoints across processes and then compresses them. Our experiments show that we can achieve compression ratios of up to 70% for a multi-physics code and reduce overhead of writing checkpoints by up to 87%. In our in-system storage improvement work, we developed CRUISE, a light-weight file system specifically for checkpointing [?]. We developed this because on current HPC systems, the only in-system storage that exists is RAM disk which can be slow; additionally, some systems do not even provide RAM disk functionality. Using CRUISE, we achieved 1 PB/s when checkpointing 3 million tasks on the Sequoia BG/Q system at LLNL.

SCR and related research efforts have made checkpoint/restart viable at large scales on today's machines. However, as we look forward to extreme scale systems, our current approaches will likely not suffice. We focus our future research directions on minimizing the amount of data that needs to be transferred to the parallel file system for fault tolerance, and designing techniques for I/O on future hierarchical storage systems.

4.4 Automating Resilience

In addition to the annotation-driven resilience strategies mentioned above, SUPER researchers continued their push towards fully automatic compiler transformations for resilience. We have developed a compiler-based approach using Triple Modular Redundancy to provide HPC software with fault tolerance against transient faults, as we expect them to manifest themselves on future Exascale architectures. Performance results using ROSE-based source translation showed that for a randomly selected subset of benchmarks the overhead of this extra layer of support is about 20%. Prior results had shown 0% to 30% overhead for the Jacobi iteration on common hardware; and 20%, 40%, 26%, and 2% overhead for a randomly selected subset of benchmarks from the Livermore Loops [31]. We expect that may be competitive with future approaches to fault tolerance using check-point restart that may be much more expensive or maybe even intractable for Exascale. This work is released as a framework within ROSE to support research work in this area by ourselves and collaborators.

We have also extended the ROSE Outliner to generate CUDA kernels from sequential loops. This new feature helps to automate the auto-tuning efforts focusing on heterogeneous computing using GPUs. It also enables us to explore new language features (some of which might be tunable) for OpenMP support for accelerators including GPUs and Intel MIC processors. We are preparing a paper for this work to be submitted to International Workshop on OpenMP (IWOMP) 2013.

4.5 Coarse Grain Resilience

In addition to hardware faults that may corrupt results without bringing down an entire compute node, there are many sources of failure that must be dealt with at a higher level. As suggested above in Section 2.1, large scale ensemble campaigns must be prepared to deal with compute node failure, networking and I/O interruptions, system software problems, algorithmic bugs, and failure of the scientific models when presented with inputs outside their domain of applicability. SUPER researchers are exploring autonomic computing methods that deal with failure of an ensemble member at any of these levels.

4.6 Future Plans for Resilience

Say something that correlates with institutional plans.

UTK: For the Future, we plan to look into adding more checkers (functions that can validate a current state of an algorithm progress in order to validate the current data), and continue to improve the existing as well as develop new on-line strategies to deal with soft-errors.

LLNL: In the area of checkpoint/restart for extreme scale computing, we plan to research methods for minimizing the amount of data transferred to the parallel file system for fault tolerance, and design scalable strategies for I/O on future hierarchical storage architectures.

UNC: We are planning to continue our work on frameworks for large ensembles with an emphasis on integrating leadership class system resources.

5 Overall Optimization of Performance, Energy, and Resilience

When a single objective, such as execution time, is available, the autotuning search problem can be posed as a numerical optimization problem. However, it is increasingly common to have multiple objectives, such as execution time, energy consumption, resilience to errors, peak power demands, and memory footprint. When the relative weights or constraints on these objectives are not known at search time, one must pose the autotuning search problem as a multi-objective optimization problem.

5.1 Optimization Research

In initial SUPER work, we developed a common framework for collecting and exchanging data for multiple metrics, created a collection of test problems, and set up an infrastructure for collecting power and execution time data on these problems for multiple platforms.

We have now deployed this infrastructure for a wide variety of autotuning decision spaces and kernels. We have examined HPC kernels from the SPAPT autotuning suite [2] (`mm`, `atax`, `trmm`, `bicgkernel`, `lu`, `adi`, `jacobi`, `fdtd`), the TORCH suite [24], and CSPARSE [16], as well as the mini-applications from the Mantevo project [19] (`miniGhost` and `miniFE`). Our decision spaces have included tunable decision parameters related to:

- code transformations readily processed by ORIO [38] (e.g., loop unroll/jamming, cache tiling, register tiling, scalar replacement, array copy optimization, loop vectorization);
- parallelization strategies (OpenMP/number of threads, MPI/number of nodes); and
- dynamic voltage and frequency scaling.

We have examined objectives based on power, energy, and execution time on a wide variety of architec-

tures. Our results in [5] are based on data collected on

- a first-generation Intel Xeon Phi (Many Integrated Core) architecture;
- the Intel Xeon E5530 architecture, with detailed power measurement capabilities; and
- one IBM Blue Gene/Q rack at Argonne.

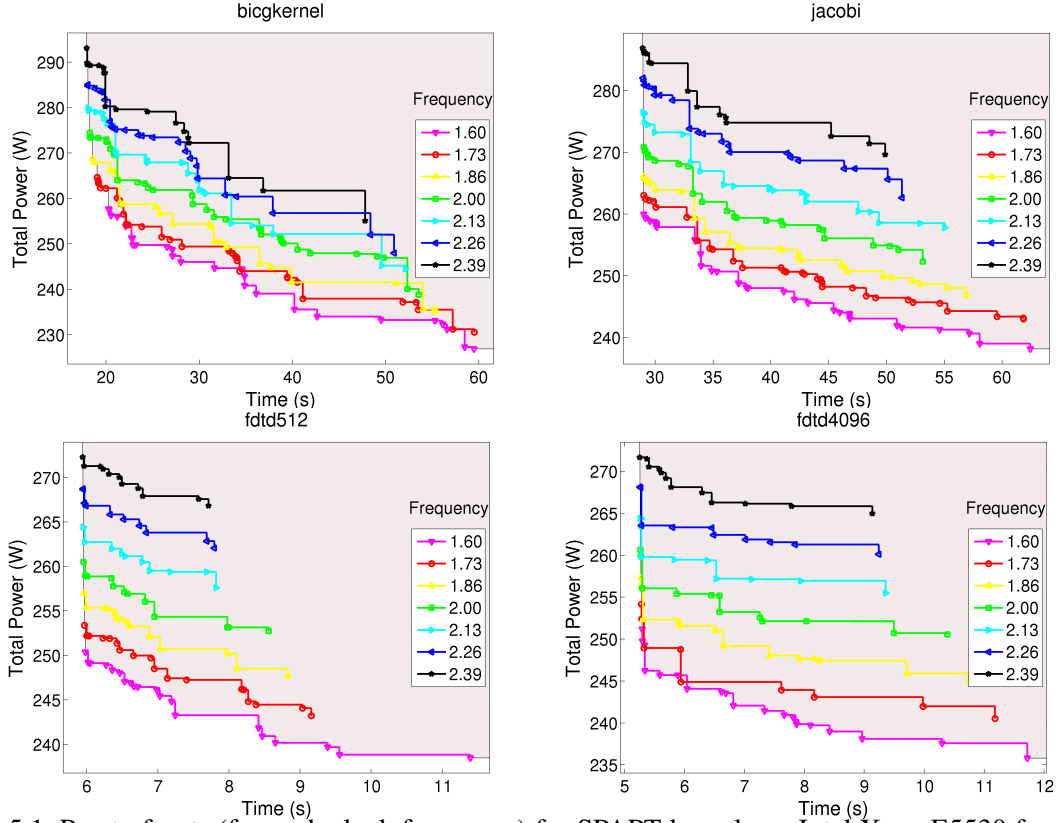


Figure 5.1: Pareto fronts (for each clock frequency) for SPAPT kernels on Intel Xeon E5530 for the objectives time and total system power as measured with WattsUp. The shaded area shows the Pareto front across all frequencies. (Plots from [5].)

Figure 5.1 demonstrates the tradeoffs between execution time and total system power (as measured at the wall) for four different kernels from SPAPT. For each of the seven clock frequencies shown, 300 code variants were generated from the SPAPT decision space. The lines mark the resulting *Pareto fronts* for each frequency, which correspond to the time and power for variants that are considered optimal from a multi-objective perspective. For each of the code variants defining the Pareto front, there is no code variant that both runs in less time and draws less power. The shaded region shows the Pareto front (essentially the Pareto front of the set of frequency-based Pareto fronts) if clock frequency is also treated as a decision parameter that can be set by a user or system administrator.

Such Pareto fronts provide solutions for a wide variety of decision-based problems in high-performance computing. In addition to revealing the code variants that minimize each objective in isolation (given by the extreme points of the Pareto front), they also provide the optimal code variants for all linear combinations of objectives. A Pareto front can also be used to quantify the effect of using one of the objectives as a constraint (and minimizing the others). For example, for the `bicgkernel` at 1.6GHz in Figure 5.1, decreasing a power cap from 250W to 240W would increase the minimal execution time from 23s to 36s.

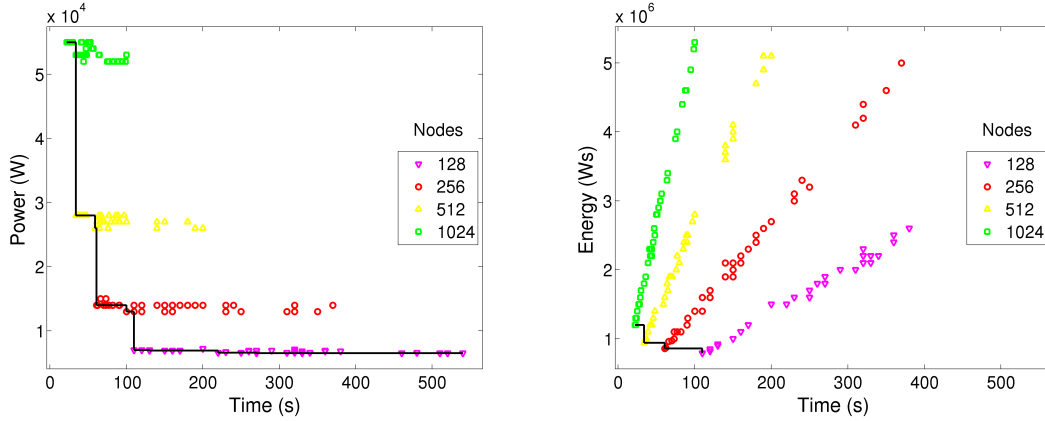


Figure 5.2: Power, energy, and time for the mini-application `miniFE` on an IBM Blue Gene/Q. (Plots from [5].)

Figure 5.2 demonstrates this multi-objective approach on Blue Gene/Q for `miniFE`, a finite element mini-application that implements kernels representative of unstructured, implicit finite-element applications. Here, the decision space consists of four parameters: two generic parameters that control the scaling behavior and two application-specific parameters. We highlight (through different colors and markers) the parameter corresponding to the number of nodes, since this decision parameter has the greatest effect on power and energy usage. Figure 5.2 (left) shows the tradeoffs between power and time, with power draws clearly being driven by the node counts. Figure 5.2 (right) shows that there are also tradeoffs between energy and time. The resulting Pareto front could be used to quantify the energy price associated with obtaining additional speedups.

5.2 Future Plans for Optimization

Say something that correlates with institutional plans.

6 Engagement

6.1 Application Partnerships

SUPER personnel participated in writing SciDAC-3 SAP proposals for each of the five Office of Science programs. The funded projects that explicitly include SUPER investigators are listed below.

Office	Project	PI	SUPER Inst.
BER	Applying Computationally Efficient Schemes for BioGeochemical Cycles (ACES4BGC)	F. Hoffman (ORNL)	ORNL
BER	Multi-scale Methods for Accurate, Efficient, and Scale-Aware Models of the Earth System	W. Collins (LBNL)	LBNL
BER	Prediction of Ice and Climate Evolution at Extreme Scales (PISCEES)	W. Lipscomb (LANL)	LBNL ORNL
BES	Developing Advanced Methods for Excited State Chemistry in the NWChem Software Suite	C. Cramer (Minnesota)	LBNL
BES	Optimizing Superconductor Transport Properties through Large-scale Simulation (OSCon)	A. Glatz (ANL)	ANL
BES	Simulating the Generation, Evolution and Fate of Electronic Excitations in Molecular and Nanoscale Materials with First Principles Methods	M. Head-Gordon (LBNL)	LBNL
FES	Partnership for Edge Plasma Simulation (EPSi)	C.-S. Chang (PPPL)	ORNL
FES	Plasma Surface Interactions (PSI)	B. Wirth (UTK/ORNL)	ORNL
HEP	Community Petascale Project for Accelerator Science and Simulation (ComPASS)	P. Spentzouris (Fermi)	ANL
NP	A MultiScale Approach to Nuclear Structure and Reactions	W. Haxton (UCB)	LBNL
NP	Computing Properties of Hadrons, Nuclei and Nuclear Matter from QCD	F. Karsch (BNL)	UNC
NP	Nuclear Computational Low Energy Initiative (NUCLEI)	J. Carlson (LANL)	ANL

SUPER is currently collaborating with the two additional SciDAC-3 SAPs as well as FASTMath on white papers outlining further opportunities for SUPER personnel to have immediate impact on these projects.

Recent SUPER engagement progress includes the following.

1. For the EPSi (FES) SciDAC-3 SAP we contributed to the recent performance optimization and scaling study of the XGC1 code on the Cray XK7 at OLCF, demonstrating excellent full system scaling (to almost 300,000 cores), effective utilization of GPU accelerators, and a $4\times$ overall speed-up compared to previous versions of the code. Similar excellent scaling was demonstrated on the IBM BG/Q at ALCF (to over 500,000 cores) and on the Cray XC30 at NERSC. These results will be presented as a poster at SC13. The end-to-end and holistic performance data collection scripts are also being used and further developed to track performance changes during development.
2. Within the PSI (FES) project, SUPER is contributing expertise in designing and integrating a lightweight “always-on” performance monitoring infrastructure to XOLOTL, a new plasma surface interactions code being developed within the PSI project.
3. For the ACES4BGC (BER) SciDAC-3 SAP we introduced the end-to-end and holistic performance data collection scripts into production versions of the CESM code, and are using this capability to track performance changes during development. By optimizing the coupled model configuration and tuning available performance options for the target system, we recently enabled a $1.6\times$ speed up for one production science run on the Cray XK7 at OLCF, and a $2\times$ speed up for another.
4. For the PISCEES (BER) SciDAC-3 SAP we augmented the performance instrumentation available

in Trilinos and contributed to a study demonstrating good scalability out to nearly 10,000 cores on the Cray XE6 system at NERSC and on the Cray XK7 system at OLCF for a new ice sheet model dynamical core (FELIX). We also demonstrated that the more “modern” solver technologies being evaluated in FELIX are slower than the original, and that additional optimization work was needed.

We also worked with another new ice sheet model dycore (BISICLES), which is built on the CHOMBO framework. BISICLES is currently transitioning from a 2D to a 2.5D or 3D formulation. In anticipation of the reformulation we used a multigrid benchmark from the CACHE project in order to understand the potential performance benefits of various optimizations.

5. For the Multiphysics (BER) SciDAC-3 SAP we are collaborating in the integration of a threading model into the Model for Prediction Across Scales (MPAS) Ocean code, to allow for hybrid (MPI/OpenMP) parallelization, with the goal of enabling on-node scalability for emerging many-core platforms such as the Intel MIC architecture. Results on over 8000 cores of the Cray XE6 system at NERSC show that the hybrid implementation achieves comparable performance and scalability to MPI.

We also worked with the implicit solvers research group to integrate the instrumented Trilinos developed in the PISCEES work into their codes, enabling the performance diagnosis and partial elimination of a long-standing performance problem. Further optimizations are ongoing.

6. For the electronic excitation materials (BES) SciDAC-3 SAP we have focused on optimizing and distributing Anna Krylov (USC) et al’s libtensor tensor contraction library for CCSD. We recently completed an analysis of the code and identified several opportunities for optimization. We are also collaborating on the performance optimization and further parallelization of the RAS-SF (Restricted Active Space Spin-Flip) code.
7. For the NWCHEM (BES) SciDAC-3 SAP we determined that, by introducing finer parallel granularity, the load imbalance that occurs in the TEXAS integral package when using more than 16 threads can be essentially eliminated. This resulted in between 25% and 35% improvement in performance on the Cray XE6 system at NERSC.
8. For the CalLAT (NP) SciDAC-3 SAP we collaborating with the developers to improve I/O performance by incorporating support for parallel-HDF5.

6.2 Institute Awareness

The FASTMath, QUEST, and SDAV Institutes all have strong emphases on performance and on being architecture-aware. In consequence, SUPER should be able to contribute to the success of the other institutes. The recently funded Roofline collaboration is the only directly funded collaboration between the Institutes at the current time, SUPER is pursuing opportunities for collaboration within the context of SciDAC-3 SAPs in which both SUPER and another Institute are engaged. A current example is the performance optimization of PETSc iterative linear solvers used in the solution of nonlinear PDEs discretized on a structured grid. Other likely targets, currently under discussion with the developers, include optimization of kernels in the Trilinos library (FASTMath) and Chombo library (FASTMath) and optimization of a QUEST job management system, all within the context of the PISCEES project.

6.3 Outreach

Important proving grounds and delivery mechanisms for SUPER technologies are DOE SC's computing centers, ALCF, NERSC, and OLCF. Each center has its own technologies and methodologies that it promotes to its users and that are used internally for performance tracking. To promote SUPER technologies to the centers, we are currently taking the approach of demonstrating these technologies in the SciDAC-3 application project codes running at the centers. As more projects adopt these technologies, and as we can demonstrate their utility, we will work with the centers to identify common interests and collaboration opportunities. Note that much of the SUPER work in end-to-end and holistic performance measurement and analysis is focused on the centers, and will thus be immediately useful to the centers. Also, some of this work requires support from system architects, especially in the areas of I/O and batch scheduling, and so will require collaboration with the center staff.

7 Augmentation

To date, SUPER has been augmented twice to address additional research problems for ASCR and HEP.

7.1 Transforming Geant4

SUPER investigators submitted a supplement proposal "Exploring the Transformation of Geant4 for the Future", which was funded and started in FY13. The Geant4 augmentation is lead by Hovland at ANL, and involves Oregon, USC and UNC SUPER participants as subcontractors. High Energy Physics has similarly augmented the Fermi National Accelerator Laboratory and the SLAC National Accelerator Laboratory to provide Geant4 domain experts to this collaborative effort. Furthermore, experts from CERN are also participating.

Initial work focuses on analysis of the multi-threaded version of Geant4 using HPCToolkit and detailed performance analysis of the emerging GPU implementations of portions of Geant4. The performance insights gained in this effort will be used to guide the long term transformation of Geant4 to enable it to scale and exploit the performance potential of emerging, multicore processors.

7.2 Roofline Toolkit

Using the super augmentation received in late FY13, we began the construction of the Roofline Toolkit. This effort spans LBL, ANL, and Oregon and covers three major facets: hardware performance characterization, source code analysis, and integration/visualization. At LBL, we have codified the requisite functionality of the hardware characterization component needed to quantify the memory, cache, and in-core performances of various multi- and manycore CPU architectures. We are currently in the process of integrating the various ad-hoc benchmarks we've written into a single portable benchmark. Additionally, we are collaborating with researchers at NERSC to verify the methodology using Crays performance counter tools. Thus far, coarse grained application characterization (not function characterization) coincides well with Roofline predictions for a variety of miniApps. We are planning on expanding this activity to include some load-level fine-grained performance counter instrumentation for additional verification and validation of the Toolkit. **Boyana - would you like to add anything?**

8 Management

Bob Lucas of USC serves as the overall manager of the SUPER Institute, assisted by Lenny Oliner of LBNL. Management duties include setting the overall research direction, conducting teleconferences, over-

seeing project meetings, preparing reports and reviews, connecting to other SciDAC-3 institutes and application projects, and representing SUPER in various community activities. This section of the Spring 2013 Director’s report discusses the organization of SUPER, mechanisms used to coordinate our research, and the strategy used for outreach to the other SciDAC-3 institutes and application partnerships. A more detailed description of is available in the SUPER Operating Plan.

8.1 Organization

The organization and goals of the SUPER SciDAC-3 institute are based on the collective experience that our team members have acquired over the past ten years of SciDAC. We have chosen to organize a broadly-based project with expertise in compilers and other system tools, performance engineering, energy management, and resilience, and with the prerequisite skills in measurement, source analysis, and execution modeling and optimization. Leadership within SUPER is distributed to reflect this broad range of expertise. SUPER’s flat organizational chart, depicted in Figure 8.1, is possible due to both our modest size, as well as to the decade-long history of working together enjoyed by most of its investigators.

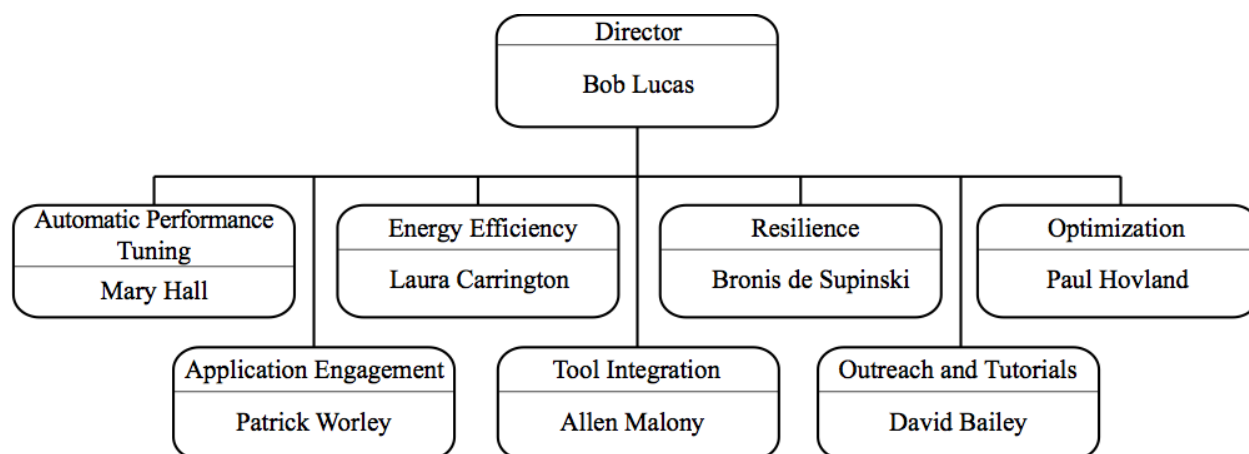


Figure 8.1: SUPER organizational chart.

SUPER investigators have expertise in using a broad range of commercial and open source performance analysis tools, in particular those deployed at NERSC, ALCF, and OLCF. In addition, we have developed our own research software that allows us to make unique contributions. Furthermore, we are closely collaborating with Prof. John Mellor-Crummey at Rice, the primary developer of the HPCToolkit and with developers of other performance analysis tools such as Scalasca, Paraver, and PerfExpert. We are following the successful strategy developed in the SciDAC-2 Performance Engineering Research Institute (PERI) of developing interfaces between these tools, integrating them, and applying the resulting synergistic apparatus to real DOE applications. This approach will produce a robust and comprehensive end-to-end application optimization infrastructure that would otherwise be beyond the reach of any one research group. Ultimately, our goal is to use it to optimize DOE applications in terms of performance, energy consumption, and resilience.

SUPER and its collaborators are geographically distributed. Therefore, we have developed mechanisms to facilitate communication and collaboration among our investigators. Every Wednesday, at noon in the Eastern time zone, a one-hour telephone conference call is scheduled. We have buttressed this with an additional one hour time slot on Tuesdays, at 2 PM Eastern. The first Wednesday of every month is an all-hands call in which we discuss the direction of the project and other management issues. The subsequent Wednesday phone calls are devoted to software integration, resilience, and optimization. The Tuesday time

slot is allocated to auto tuning, end-to-end performance, energy, and measurement and modeling.

To further ensure that SUPER collaboration is effectively maintained, we schedule a two-day, all-hands, face-to-face project meeting every six months. The first five such meetings were hosted by Oregon, UNC, ANL, Rice, and LBNL. A tentative schedule for the next five meetings has been established. Finally, we are taking advantage of conferences and PI meetings that attract a significant number of our investigators to opportunistically schedule shorter SUPER meetings. The best example of this is the SC conference series, held every November, at which LBNL organizes a meeting every Monday.

For outreach to the broader scientific community, SUPER will use the Web site www.super-scidac.org, created and maintained by UTK and UTEP. It includes links to the participating institutions, recent news and publications. In addition, SUPER personnel conduct tutorials at conferences and other venues to teach computational scientists how to optimize performance, energy consumption, and resilience.

8.2 SciDAC Application Partnerships

The SUPER research strategy does not merely encourage collaboration with DOE computational science investigators, it demands such interactions, further stimulating our research work. The SciDAC-3 SAPs serve as precisely the kinds of computational science challenges we are looking for, and we are pleased that SUPER investigators are active participants in twelve of the eighteen projects. Details were presented above in section 6.1. Other SUPER investigators will engage in direct collaborations with the SAPs on a more selective basis, either when there are mutually beneficial research objectives, or when such contributions are identified as being of high priority by our DOE program manager. We expect to apply the most advanced autotuning and multi-objective optimization techniques to DOE's most strategic applications, thus pushing the envelope on both computer science and computational science advances on emerging petascale platforms.

8.3 Institute Directors Plan

It is also a goal of SUPER to identify opportunities to work with the other SciDAC-3 Institutes, FASTMath, QUEST, and SDAV. Optimization of widely used mathematical libraries, in terms of performance, energy consumption, and resilience, will help maximize SUPER's impact on DOE computational science. Towards that end, we have concluded the first year of a collaborative effort with FASTMath and QUEST to investigate the ParaDis code, an extension of our institutes funded by NNSA.

The leaders of the initial three SciDAC-3 institutes (Diachin, Najm, Lucas, and Bailey), met in Livermore, CA, on Nov. 10, 2011, to begin developing plans for communication and coordination amongst our institutes at both the management and technical levels. We held a second meeting, including SDAV's Arie Shoshani, at the SciDAC-3 PI meeting on July 26, 2013. We also meet telephonically. These meetings tend to be scheduled opportunistically, at technical conferences or DOE PI meetings, to therefore minimize unnecessary travel and the expense that would incur.

Technical collaboration between the institutes is facilitated by assigning liaisons to ensure that opportunities to interact are not missed. We will provide each other access to our software repositories, as appropriate, and our Institute web sites will point to each other. We will provide educational opportunities so that all of our investigators are aware of the research being conducted by their peers in the other institutes. This has already included inviting members of other institutes to participate in SUPER all-hands meetings and teleconferences for exchanging information on topics of mutual interest.

9 Summary

Summary

Two years into the SciDAC-3 program, SUPER is performing well. As described above in the overall report, as well as in the following institutional progress reports, significant technical progress has been made on a broad range of performance, energy, resilience and optimization challenges.

The team is functioning smoothly, as one would expect given that many of its investigators have over a decade of SciDAC experience. We meet regularly, both telephonically and in person. Eight phone calls are scheduled every month, and others occur as needed. We have had team meetings at Oregon in September 2011, at SC in November of 2011 and 2012, at UNC in March 2012, at ANL in September 2012, Rice in April 2013, and most recently at LBNL in September, 2013. We are actively collaborating with other Institutes, SAPs, DOE computing centers, and the broader DOE SC scientific computing community.

10 Institutional Progress Reports

10.1 Argonne National Laboratory

Argonne’s work in SUPER focuses on multi-objective optimization (Balaprakash, Hovland, Wild), performance modeling (Narayanan, Norris, Bui), and autotuning (Norris, Mametjanov).

In September, we had a personnel change, with Norris moving to the University of Oregon.

10.1.1 Optimization

In initial SUPER work, we developed a common framework for collecting and exchanging data for multiple metrics, created a collection of test problems, and set up an infrastructure for collecting power and execution time data on these problems for multiple platforms.

We have now deployed this infrastructure for a wide variety of autotuning decision spaces and kernels. We have examined HPC kernels from the SPAPT autotuning suite [2, 7] (`mm`, `atax`, `trmm`, `bicgkernel`, `lu`, `adi`, `jacobi`, `fdtd`), the TORCH suite [24], and CSPARSE [16], as well as the mini-applications from the Mantevo project [19] (`miniGhost` and `miniFE`). Our decision spaces have included tunable decision parameters related to:

- code transformations readily processed by Orio [38] (e.g., loop unroll/jamming, cache tiling, register tiling, scalar replacement, array copy optimization, loop vectorization);
- parallelization strategies (OpenMP/number of threads, MPI/number of nodes); and
- dynamic voltage and frequency scaling.

We have examined objectives based on power, energy, and execution time on a wide variety of architectures. Our results in [5] are based on data collected on

- a first-generation Intel Xeon Phi (Many Integrated Core) architecture;
- the Intel Xeon E5530 architecture, with detailed power measurement capabilities; and
- one IBM Blue Gene/Q rack at Argonne.

Figure 10.1 demonstrates the tradeoffs between execution time and total system power (as measured at the wall) for four different kernels from SPAPT. For each of the seven clock frequencies shown, 300 code variants were generated from the SPAPT decision space. The lines mark the resulting *Pareto fronts* for each frequency, which correspond to the time and power for variants that are considered optimal from a multi-objective perspective. For each of the code variants defining the Pareto front, there is no code variant that both runs in less time and draws less power. The shaded region shows the Pareto front (essentially the Pareto front of the set of frequency-based Pareto fronts) if clock frequency is also treated as a decision parameter that can be set by a user or system administrator.

Such Pareto fronts provide solutions for a wide variety of decision-based problems in high-performance computing. In addition to revealing the code variants that minimize each objective in isolation (given by the extreme points of the Pareto front), they also provide the optimal code variants for all linear combinations of objectives. A Pareto front can also be used to quantify the effect of using one of the objectives as a constraint (and minimizing the others). For example, for the `bicgkernel` at 1.6GHz in Figure 10.1, decreasing a

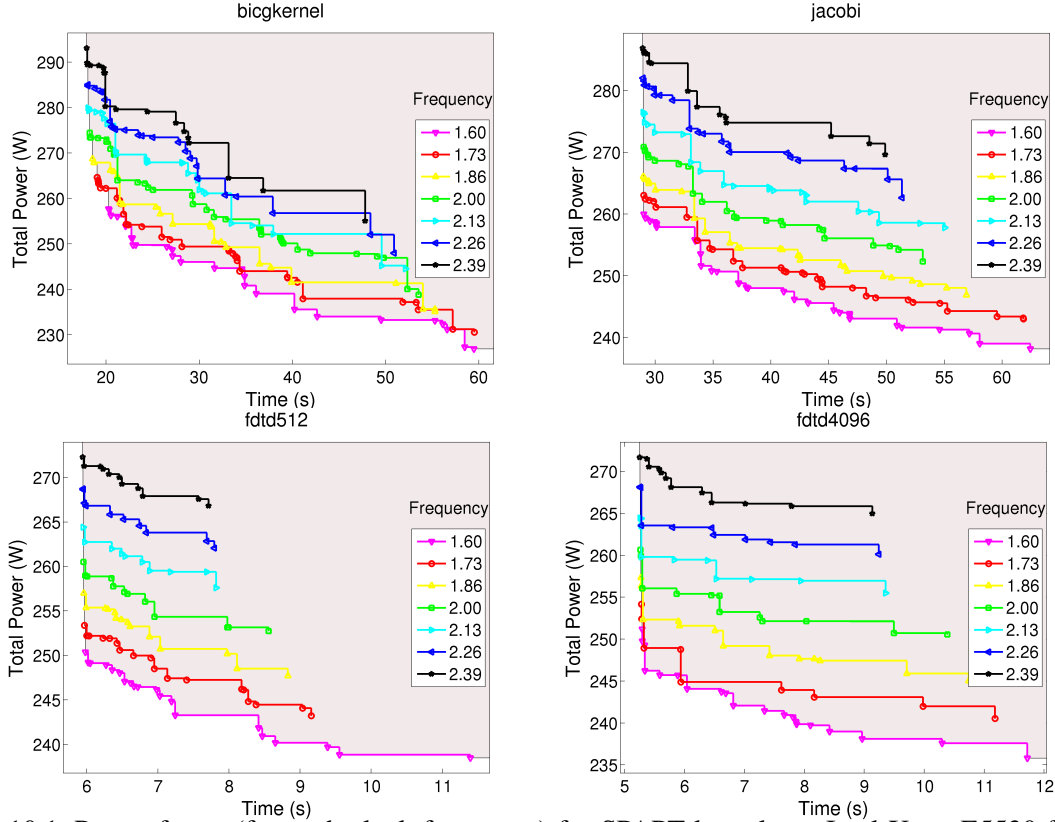


Figure 10.1: Pareto fronts (for each clock frequency) for SPAPT kernels on Intel Xeon E5530 for the objectives time and total system power as measured with WattsUp. The shaded area shows the Pareto front across all frequencies. (Plots from [5].)

power cap from 250W to 240W would increase the minimal execution time from 23s to 36s.

In addition to the above multi-objective work (described in [5] and the poster [6], which has been selected as a finalist for best poster at SC13), we also begun exploring active learning-based approaches for building surrogates [3, 4]. We expect to build on such methodology when deriving efficient algorithms for performing multi-objective optimization.

10.1.2 Performance modeling

PBound. We have improved the modelling of reuse within PBound. We have refined the reuse analysis within PBound. By implementing per-reference analysis we are able to improve the accuracy of analysis. Pbound now provides self-spatial and self-temporal reuse and allows the relative importance of loops and references to be gauged. In order to achieve this, the logic to how reuse is evaluated was considerably revised. Based on these changes an integration of PBound to CHiLL via a *decision algorithm* has been completed. Using this integration, model driven autotuning for key kernel was performed. We have begun to model parallelism in the code, computation partitioning between CPU and GPU, and are extending PBound's capabilities for Fortran input.

10.1.3 Autotuning

We extended GPU code generation and autotuning support in Orio as part of the restricted C input syntax of the existing Loop module [15, 34]. The GPU-specific optimization parameters include numbers of

threads, blocks, and streams, L1 cache size (for newer NVIDIA chips), compiler flags, and cascading reductions. We integrated autotuning into a new grid-based PETSc matrix datatype and vector type for GPUs. In the last few months, we also implemented preconditioners that can operate with the autotuned kernels, including a polynomial preconditioner and an ILU preconditioner, with expected improvements in convergence (10-20x over no preconditioning).

Based on our experiences with the detailed analysis described in Sec. 10.1.2, we have begun work on an autotunable MPI benchmark that overlaps communication and computation and extensions to Orio to support MPI code generation and tuning. The autotuning module generates parallel code that overlaps communication and computation. The computation and communication in the generated code are parameterized in order to explore different overlap strategies. Initial experiments on the Vesta Blue Gene/Q system at Argonne varying the performance and input parameters to the autotuning module showed that (1) increasing the amount of computation performed allows for larger messages to be overlapped with the computation and (2) the rendezvous protocol can handle the transfer of large messages better than the eager protocol. The MPI autotuning module is able to quantify how much overlap is possible for a given problem and provides an optimized implementation. As the next step, we will extend this MPI autotuning module to support MPI nonblocking collectives and apply machine learning techniques for reducing the optimization search space.

We are also adding compiler-independent SIMD vectorization pragmas to Orio. An annotated loop is transformed into a set of loop variants with compiler-specific pragmas (e.g. Intel's `icc` and GNU's `gcc`). The loop variants are empirically evaluated and the top-performing candidate is chosen as the result of autotuning specialization. This approach improves performance portability by abstracting away from specific compiler pragmas and increases performance (up to 2.3x speedup, see Figure 10.2) through empirical evaluation of a set of low-level SIMD pragmas [33].

10.1.4 Application Partnerships

Argonne was an active participant in many SciDAC-3 application partnership proposals. In FY12, we received funding for and began representing SUPER in the following SciDAC-3 application partnerships:

BES Optimizing Superconductor Transport Properties through Large-scale Simulation (OSCon); PI A. Glatz (ANL)

HEP Community Petascale Project for Accelerator Science and Simulation (ComPASS); PI: P. Spentzouris (Fermi)

NP Nuclear Computational Low Energy Initiative (NUCLEI); PI: J. Carlson (LANL).

10.1.5 Geant4 Supplement

The Geant4 collaboration involves work lead by Hovland at ANL, and involving USC and UNC SUPER participants. Until August, 2013 the work was led by Norris at ANL. As part of the electromagnetics code review (with Fermilab), we reviewed the PhysicsVector data structures and studied the effects of several refactorings, including using or not using STL vectors and algorithm, different types of memory allocation methods and reordering of data members for better locality in the interpolation functions. The performance analysis employed both HPCToolkit and TAU and compared hardware counters to quantify the cache performance differences between different code versions.

10.1.6 ComPASS Supplement

We also submitted a supplement proposal "Parallel Optimization for Leadership Class Accelerator Design", which was awarded at the end of FY13. This work is lead by Wild at ANL and involves HEP

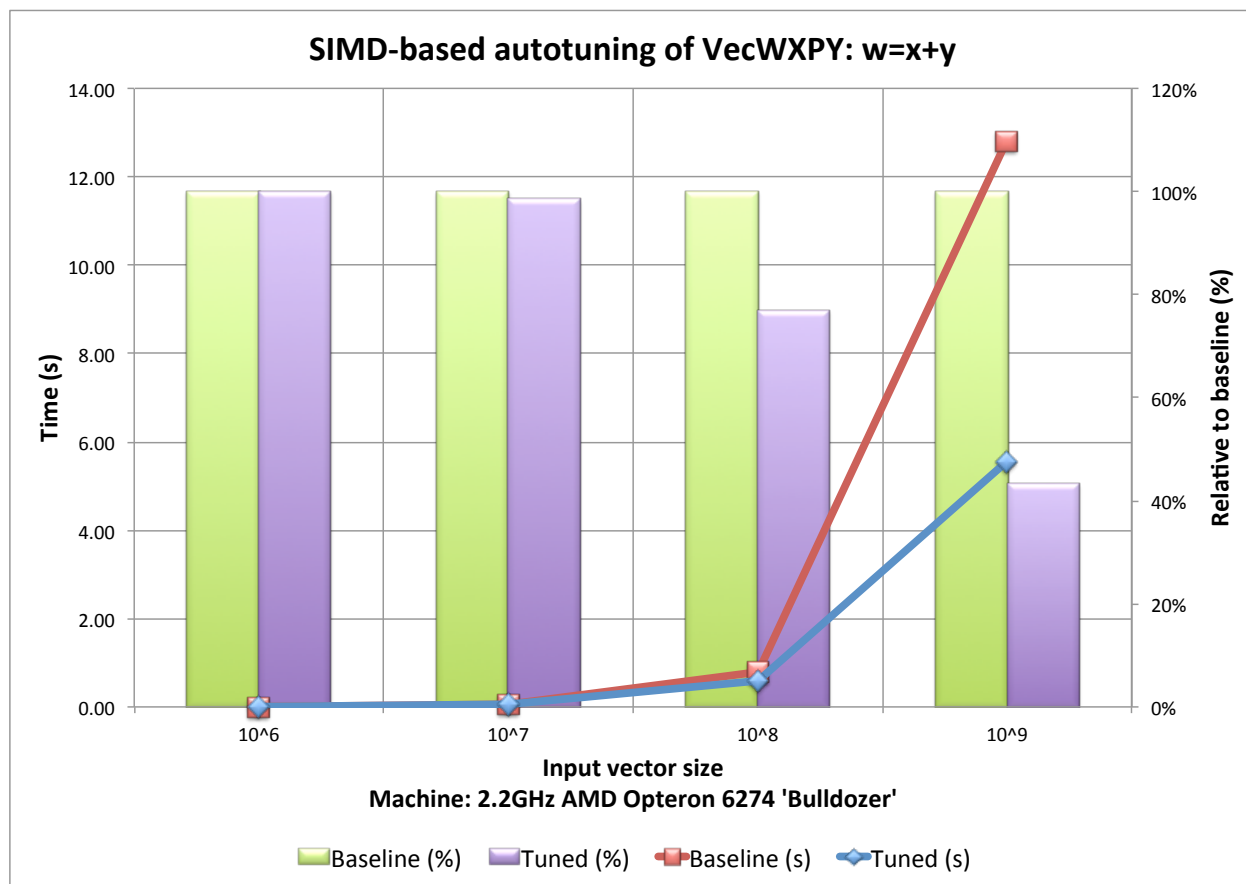


Figure 10.2: Auto-vectorized (baseline) vs. autotuned kernel's performance with Intel's ICC 13.1.3. Each input vector size is listed with the kernel's absolute execution time (in seconds, lines) along the left vertical axis as well as the relative speedup of autotuned kernel's time with respect to the baseline kernel's time for that size (in %, bars) along the right vertical axis. The baseline kernel contains no pragmas and is compiled with '-O3' flag, which turns on auto-vectorization. Autotuning delivers up to 43% or 2.3x speedup.

SciDAC-3 partners from the ComPASS project from Fermi National Laboratory and Tech-X. Fermi has a real need for optimization of a machine that will be planned to commence operation for the neutrino program some time in 2014. ANL is interviewing postdoc candidates for this work in October and November of 2013.

10.2 Lawrence Berkeley National Laboratory

LBNLs SUPER efforts are currently focused on performance instrumentation, performance analysis, and performance optimization of six different SciDAC partnerships as well as the roofline augmentation. Below we discuss some recent highlights.

1. *Martin Head-Gordon's (LBNL) electronic excitation materials project.* We focused on optimizing and distributing Anna Krylov (USC) et al's libtensor tensor contraction library for CCSD. The analysis of libtensor revealed multiple bottlenecks, including the lack of handling NUMA and cache locality, which impacts the performance of BLAS routines. We also found that the task scheduler does not always make optimal decisions for load balancing tasks, especially when the number of cores per node increases. Our tuning efforts advanced in two fronts. The first involves a better choice of scheduling quanta to achieve better load balancing with the increase of core count. The second aims at improving the performance on NUMA architectures by a better distribution mechanism of the dataset. Overall we achieved better scaling for the code with the number of cores, achieving speedup up to 82% on 32-core Carver nodes, 54% on 24-core Hopper nodes, at NERSC. We are also actively helping Dan Haxton in his code development efforts for Coulomb Grid Method for N-electrons. We improved the performance of his code by 36% in distributed environment, on NERSC Hopper, using multiple techniques. We optimized the memory management (creation and destruction) strategy for frequently allocated objects. We also reduced the amount of communication needed by the algorithm using persistent storage that caches previous communications.
2. *The TEXAS integral package (as part of the NWChem project).* In recent work, we address tuning the performance of NWChem on three currently deployed high performance computing platforms, the Cray XE6, the Cray XC30, the IBM BG/Q, as well as one emerging platform, the Intel MIC. Although NWChem provides extensive capabilities for large scale chemical simulations, we focus on one critical component: the effective evaluation of two-electron repulsion integrals with the TEXAS integral module that are needed for the construction of the Fock matrix needed in the Hartree- Fock (HF) and Density Functional Theory calculations. Overall results indicate that load balancing has significant potential for improving performance, attain up to 50% speedup. When combined with our additional optimization strategies, results in an overall speedup of up to 2.5x. On platforms that supports simultaneous multithreading, running multiple threads can improve the performance significantly. On the IBM BG/Q platform, running 2 and 4 threads per core can improve performance by 1.7x and 2.2x, respectively. Finally, extracting the full performance potential from the vector processing units is a significant challenge for NWChem. Via substantial programming effort, we obtained a vectorized version running approximately 25
3. *Bill Collins Multiscale Climate project.* Work to date has focused on integrating a threading model into the Model for Prediction Across Scales (MPAS) Ocean code, to allow for hybrid (MPI/OpenMP) parallelization, with the goal of enabling on-node scalability for emerging many-core platforms such as the Intel MIC architecture. In the development version, we have implemented several OpenMP parallelization schemes in order to evaluate their performance. We have completed detailed performance profiling and analysis of this code, both on-node and scaling across multiple nodes, to identify the major bottlenecks. Apart from implementing shared-memory OpenMP support, a number of optimization opportunities have been identified, and we have pinned down two of these to work on in the next few months. The first is to implement support for variable ocean depth memory usage in the grid data structure. An analysis of different real-world input grids have revealed large potential of performance improvement. The second is implementation of an intelligent grid cell ordering schemes to maximize memory localities and reduce data transfers. A scheme to utilize space-filling curves on the grid cells for ordering is being implemented. This implementation will be compared in terms of data transfers at on-node cache level as well as halo-exchanges among partitions across multiple nodes.

4. *Phil Jones PISCEES ice-sheet modeling project.* To date, we have focused on the AMR BISICLES code. BISICLES is currently a 2D, finite-volume, adaptive mesh refinement multigrid code that uses a viscous tensor operator and a 4-color Gauss-Seidel relaxation scheme (4 round trips to memory per relax) written in CHOMBO. The code is currently in the process of two transitions one to a 3D (2.5D) formulation and another transition that is replacing geometric multigrid (GMG) with algebraic multigrid (AMG). The latter also entails a transition from multi-color Gauss-Seidel smoothers to Chebyshev polynomial smoothers. While the code transitions, we've used a MG benchmark from the CACHE project (miniGMG) in order to understand the potential performance benefits and challenges associated with a chebyshev smoother. We've constructed a geometric multigrid solver that uses a communication-avoiding Chebyshev polynomial smoother. For our more challenging synthetic problems, the chebyshev smoother has superior algorithmic properties. However, as our reference implementation moves more data than it needs to, the performance benefit is being reduced. These insights are being addressed and simultaneously conveyed to the application designers to ensure the actual smoothers do not suffer similar ill effects. Finally, as the code transitions from GMG to AMG, the Chebyshev smoother operations will transition from stencils on a structured grid to sparse matrix vector multiplications (SpMV) — a nominally memory and bandwidth-intensive operation. We are thus in the process of creating a testbed for exploring the benefits of communication-avoiding Chebyshev smoothers for AMG to help address the emerging data movement performance bottleneck.
5. *Wick Haxtons QCD project.* Various lattice QCD, such as Chroma and MILC, make use of the QDP++ API, which provides a data-parallel programming environment for computations on a Lattice. Currently QDP++ has support for three kinds of file I/O formats: ASCII text, XML text, and flat binary. In our on-going effort, we are incorporating support for parallel-HDF5 file I/O. The advantages of using parallel-HDF5 are: it provides a compact standard scientific data storage format, improving data portability; it is built for large and complex hierarchically organized data, which fits Lattice QCD data types; it is a high-performance I/O model and supports massively parallel systems. Further, metadata in HDF5 uses XML format, which is also used by QDP++. A separate repository has been setup to work on this, which will be merged into the main QDP++ repository at Jefferson Lab once complete. We have designed the I/O API for HDF5 data read, write, and conversion to and from other formats used in QDP++. We are currently implementing the hooks for this API into QDP++ as an optional configuration feature, and writing a data conversion routine which reads a binary/XML QCD configuration file, converts it into HDF5 format and writes it out to disk in parallel. We plan to implement the full API over the next several months and perform detailed I/O performance analysis and compare with the existing I/O support.
6. *NUCLEI's MFDn code.* In MFDn, a traditional eigensolver like Lanczos is used to calculate the properties of light atomic nuclei. The very large, sparse symmetric matrix in question is the configuration interaction many-body Hamiltonian. In collaboration with FastMath and NUCLEI, SUPER is exploring the algorithmic and performance benefits and challenges of replacing Lanczos with a block eigensolver like LOBPCG. At the core of LOBPCG is both a Sparse Matrix-Dense Matrix (block of m vectors) multiplication (SpMM) and a complimentary transposed multiplication (SpMM.T). In essence, we can replace the bandwidth-bound SpMV of Lanczos with a compute-intensive SpMM in LOBPCG. Unfortunately, the implementation and algorithmic parameterizations are non-obvious and required a good deal of performance tuning and performance modeling in order maximize performance and understand the fundamental bottlenecks. Thus far, we have improved the combined SpMM, SpMM.T performance by 3x (1.5x for SpMM, 4x for SpMM.T). Additionally, our Roofline model highlights the transition from a DRAM-bandwidth bound state not to peak flops, but to either a L2 or L3 bandwidth-bound state (depending on matrix sparsity) as one increase the number of vectors (m) in the block of vectors (dense matrix). We have codified the results and submitted them to IPDPS14. On going work is exploring optimization of the other routines in LOBPCG to ensure our performance gains on SpMM are not amortized.

7.Roofline Toolkit. At LBL, we have codified the requisite Roofline functionality of the hardware characterization component needed to quantify the memory, cache, and in-core performances of various multi- and manycore CPU architectures. We are currently in the process of integrating the various ad-hoc benchmarks we've written into a single portable benchmark. Additionally, we are collaborating with researchers at NERSC to verify the methodology using Crays performance counter tools. Thus far, coarse grained application characterization (not function characterization) coincides well with Roofline predictions for a variety of miniApps. We are planning on expanding this activity to include some load-level fine-grained performance counter instrumentation for additional verification and validation of the Toolkit.

10.3 Lawrence Livermore National Laboratory

Bronis R. de Supinski is the leader of the resilience research and development thrust of SUPER described in Section 4), and he is also the SUPER PI at Lawrence Livermore National Laboratory. He is ensuring that the resilience techniques developed at collaborating institutions work together to form a comprehensive resilience evaluation tool set, and that they eventually lead to viable resilience auto-tuning capability that will simplify the development of applications that can withstand faults.

During this period, LLNL has again both coordinated and actively participated in the resilience thrust of SUPER. The lab has begun to develop a tool set for automatically characterizing the resilience properties of code regions in real world applications. Our preliminary design includes facilities that will allow the same region in a real-world run to be executed repeatedly and to have faults injected at a much higher rate than would be seen in a real system. The goal of this work is to accelerate the process of assessing application fault tolerance, and to enable the techniques described in Section 4 to be applied more rapidly to real-world codes. We have also begun preliminary work on a silent bit error detection framework that will allow us to characterize the actual silent fault rates of large machines.

LLNL has made significant strides in assessing the feasibility of using individual resilience strategies together in the same application. We have extended the techniques we used for AMG in the previous reporting period, and we have examined the performance/resilience tradeoff in more detail by combining techniques. In doing this, we have found that composite techniques can be *more* effective than using any single resilience technique. Going forward, this will guide our efforts to create automated compiler transformations that implement these hand-transformations.

LLNL has lowered the overhead of our ROSE compiler-based Triple Modular Redundancy Techniques to around 20% for real-world applications. Such techniques will be effective both with programmer-supplied language annotations, *and* with automated compiler analysis. This will integrate well with our auto-tuning efforts, and will eventually allow code to be adaptively made more robust at runtime.

LLNL has also advanced the state of the art in checkpoint/restart in the Scalable Checkpoint/Restart Library (SCR). Our research directions include asynchronous transfer of checkpoints; mcrEngine, a framework for checkpoint compression; and a light-weight file system called CRUISE for caching checkpoints on compute nodes. We found that mcrEngine achieved compressed checkpoints from a multi-physics code by up to 70% smaller than the original files and significantly reduced the overhead of writing checkpoints by up to 87%. Our file system CRUISE demonstrated write bandwidth of 1 PB/s when checkpointing 3 million tasks. These efforts will enable applications to reach solutions more quickly by removing the high overhead of checkpointing.

In addition to these directions, LLNL remains active in SUPER's performance engineering work. We have added facilities in ROSE that allow the generation of CUDA kernels from sequential loops. This feature will help us to auto-tune for heterogeneous computing platforms and to integrate our resilience language features and transformations with GPUs. In line with this, we have extended our research on tunable compiler transformations for OpenMP. We are currently preparing an IWOMP submission on this work, and we are investigating integrating our tunable OpenMP transformations with other ongoing work at LLNL on noise-resilient scheduling.

Additionally, LLNL is engaging the ParaDiS team to improve the performance of the application. We have investigated the OpenMP threading performance of ParaDiS, the characteristics of ParaDiS with respect to resilience, and collaborated with other SciDAC institutes to improve the performance of ParaDiS. Due to our efforts, we improved the performance of ParaDiS by $2\times$ at unprecedented scales on the Sequoia BG/Q

system. Our collaborations with FASTMath researchers have improved the remote force calculation times by 31%. We expect our continued collaboration to result in further improvements on the order of 45-50%.

Finally, LLNL is actively investigating new optimizations that can be performed on power-bound machines. We are looking into the energy efficiencies of techniques like power-balancing (as opposed to *load balancing*) for large applications. Power balancing would monitor the critical path of applications and move power to the few slowest nodes. If done correctly, has the potential to reduce the degree of application-level load balance needed for performance at scale, and to reduce the amount of data applications need to transfer over the network to achieve perfect balance (and therefore good parallel efficiency).

10.4 Oak Ridge National Laboratory

SUPER has five major thrust areas: performance portability, energy minimization, resilience, multi-object optimization and engagement. The ORNL team’s ongoing efforts are focused on the performance engineering subtask of performance portability and on engagement. Recent progress is described below.

10.4.1 Performance Engineering

We participated in an application characterization study of a workload containing DOE Co-Design Center proxy applications, other full applications, and benchmarks. For this characterization, we modified the mpiP lightweight MPI profiling tool to additionally output point-to-point communication data as an adjacency matrix, allowing users to easily understand an application’s logical communication topology by visualizing this matrix. For both collective and point-to-point operations, the modified tool also generates a histogram of the data volume used in these operations. We also used tools built on the Machine Independent Application Models for performance Insight (MIAMI) framework, a collection of tools for automatic construction of application-centric, single node performance models, to study instruction mix and the fraction of vectorized code in applications. This work will be presented at the 4th International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS13) [53]. We plan for this application characterization work to continue, and to include new and updated versions of the Co-Design Center proxy applications.

We are developing an approach for tracking the influence of a value on later computation. It is a direct approach that operates on-line during the run of the program being analyzed. In contrast to Taint Analysis, a similar approach that returns a binary status (i.e., “tainted” or “not tainted”), our approach indicates how much a value contributes (directly or indirectly) to a later computation. We are implementing the approach in a tool called *VIT*, and have completed support for single- and multi-threaded applications. This work was presented at the 6th Workshop on Productivity and Performance (PROPER 2013) [43]. We are currently implementing support for MPI-based applications.

Earlier in the SUPER project we developed and demonstrated the feasibility of low overhead scripting approaches for capturing and archiving end-to-end and holistic performance data for production application runs at the ALCF, NERSC, and OLCF. This augmented instrumentation (both application and capture of relevant system information) was developed further and implemented in versions of the Community Earth System Model (CESM) and in the XGC0 and XGC1 fusion simulation codes throughout FY13. Data from production runs were archived and are being used in continuing research on diagnosing performance variability and other topics of interest requiring the correlation of application and system performance data. In particular, in collaboration with Lawrence Livermore National Laboratory and University of Oregon, the multiple streams of performance data and of related metadata are being merged and archived within the TAUdb performance database and machine learning techniques are being applied to identify potential sources of performance variability, driving further development of TAUdb and the associated analysis tools.

We recently began new efforts focusing on I/O and interprocess communication. Our initial goal for the I/O work is to correlate an application’s poor I/O performance or I/O performance variability with other activity on the system while the application was running. We will begin by characterizing the I/O behavior of the MPAS-ocean model and the CESM and XGC1 running on DOE Leadership Computing Facility systems. Later, we will use our I/O characterization results and data collection infrastructure to research techniques for scheduling I/O so as to control the negative impact of contention for I/O and interconnect resources. For interprocess communication, we are investigating techniques for using offline MPI communication characterization and online data on allocated nodes to determine a mapping that minimizes the MPI communication overhead, again using CESM and XGC1 as case studies.

10.4.2 Engagement

SUPER personnel at ORNL are active participants in four SciDAC-3 application partnership projects (SAP), two from Biological and Environmental Research (BER) and two from Fusion Energy Science (FES), and play an advisory role in another BER SAP. ORNL also holds a management role with respect to engagement activities: collecting and summarizing data on progress in reports, presentations, and highlights, and helping coordinate multi-institutional activities.

EPSi. For the EPSi (FES) SciDAC-3 SAP we developed and delivered a stripped down version of the XGC1 application to SUPER, and continue to maintain and update it to track developments in the science code. The XGC1 development team is particularly interested in exploiting SUPER technologies to evaluate and optimize the recent hybrid CPU/GPU port to the Cray XK7 (Titan) at the OLCF.

We contributed to the recent performance optimization and scaling study of the XGC1 code on Titan, demonstrating excellent full system scaling, effective utilization of GPU accelerators, and a $4\times$ overall speed-up compared to previous versions of the code. Similar excellent scaling was demonstrated on the IBM BG/Q at ALCF and on the Cray XC30 at NERSC. These results were presented at the 2013 SciDAC PI meeting, submitted as a potential DOE highlight in August 2013, and will be presented as a poster at SC13. The end-to-end and holistic performance data collection scripts are also being used and further developed to track performance changes during development.

PSI. For the PSI (FES) SciDAC-3 SAP we contributed in two ways: by providing guidance on what performance data to collect and how it should be collected from XOLOTL, a new application being developed as part of the project; and by conducting performance analysis studies on existing plasma surface interaction codes such as the SOLPS application. With the primary XOLOTL developer, we designed an approach for integrating a lightweight “always on” performance data monitoring infrastructure into the code, and have begun implementing the approach. For SOLPS, which is a legacy code, we are researching how best to determine which parts of the code can be adapted to expose large degrees of parallelism needed to make use accelerators such as a GPU or Intel Phi.

ACES4BGC. For the ACES4BGC (BER) SciDAC-3 SAP we introduced the end-to-end and holistic performance data collection scripts into production versions of the CESM code, and are using this capability to track performance changes during development. By optimizing the coupled model configuration and tuning available performance options for the target system, we recently enabled a $1.6\times$ speed up for one production science run, and a $2\times$ speed up for another.

PISCEES. For the PISCEES (BER) SciDAC-3 SAP we are working with A. Salinger of FastMATH and PISCEES to evaluate the performance of a new ice sheet model dynamical core (FELIX) being developed using the Trilinos libraries and with K. Evans on defining performance benchmarks to be included with a V&V test suite. We recently augmented the performance instrumentation available in Trilinos and contributed to a study demonstrating good scalability for FELIX out to almost 10,000 cores on Hopper at NERSC and Titan at OLCF, results for which were presented at the CESM Workshop in July 2103 and at the 2013 SciDAC PI meeting. We also demonstrated that the more “modern” solver technologies (Belos, ML) being evaluated in FELIX are slower than the original (AztecOO), and that additional optimization work was needed.

Multiphysics. For the Multiphysics (BER) SciDAC-3 SAP, a project with which ORNL does not have a funded participant, we worked with the implicit solvers research group to integrate the instrumented Trilinos developed in the PISCEES work into their codes, enabling the performance diagnosis and optimization of a long-standing performance problem.

10.5 University of California San Diego

UCSD's contributions to advancing SUPER's overall research fall into two main categories – Energy Efficiency in HPC via the Green Queue framework and Multi-objective optimization of power, performance and energy.

10.5.1 Green Queue – Energy Efficiency in HPC

At UCSD we are working to develop an automated framework, PMaC's Green Queue, that can utilize both application- and machine-specific details (or characterizations) to optimize large-scale parallel applications for energy efficiency. The Green Queue framework consists of methodologies that accurately identify customized hardware settings for different computational phases of scientific applications that reduce the energy required to run the applications to completion. One can view these methodologies as dynamically adjusting the hardware to *exactly* match the demands an application puts on the hardware; methodologies that can greatly help address the power wall problem faced by the HPC community on our path to the exascale. We evaluated an initial prototype of the Green Queue framework [39, 48] by deploying it to optimize large scale codes of high importance to the DOE (e.g., MILC, LAMMPS and GTC). The results were encouraging – for example, energy required to run LAMMPS on 1024 cores of Gordon¹ saw a reduction of 32% when optimized by the Green Queue framework.

In addition to continuing to evolve and engineer the Green Queue prototype, we have spent a majority of time this year on the following two key components of the framework – 1. accurate models that predict how changes in application workload affects system level power draw and 2. accurate models that predict the sensitivity of an application's execution time and energy behavior when subjected to different hardware settings. These models utilize workload characteristics that affect power draw and performance responses of a system as predictors. The initial prototype relies on measured performance sensitivity analysis, which can limit the prototype's usability because the measured sensitivity is tied directly to the settings (e.g., input data) under which the analysis was performed. In order to fully automate the process of deriving customized hardware settings for different application phases and to enhance its applicability, Green Queue must rely *only* on the models and developing a unified methodology that enables power, performance and energy modeling is the current focus of our work.

One of the key challenges in designing a unified methodology is related to the construction of a single training set (consisting of micro-benchmarks) that is rich enough to encapsulate various power, performance and energy interactions between the hardware and software. Our current training set consists of the following – 1. a set of automatically generated loops designed to stress different components of a system from the pcubed [27] suite, 2. set of widely used HPC kernels available in the SPAPT autotuning suite [2] and 3. a set of idioms [12] (different compute and memory access patterns) prevalent in HPC. We note that this is an evolving set; as we test our models on larger set of real applications, more types of computations are likely to be added. For a given target platform, the training micro-benchmarks are evaluated using different available hardware settings (e.g., different clock frequencies) to capture the interplay of power and performance for different computations. Associated with each micro-benchmark is a characterization profile, which consists of cache statistics, computational intensity metrics, and value dependence distances for integer and floating point operations. The measured power and performance characterization data from the micro-benchmarks serve as the training set to develop system level power and performance models. The models can be then used to identify the types of computations amenable to saving energy and to map these computations to phases in real applications for customized hardware settings.

¹Gordon is a Sandybridge-based supercomputer at the San Diego Supercomputer Center.

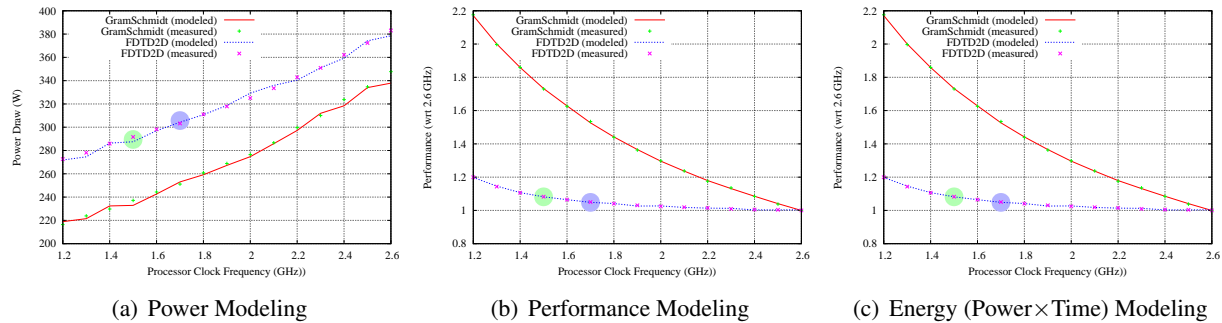


Figure 10.3: Power, performance and energy modeling for a memory bound (FDTD) and a compute bound kernel (Gram-schmidt) kernel. Green and blue circles in the graphs provide a flavor of performance-energy tradeoff decisions that can be made with models.

We present a preliminary example to illustrate the idea of choosing energy optimal CPU clock frequency settings through the use of power and performance models. The example uses two HPC kernels – two-dimensional Finite-Difference Time-Domain (FDTD) kernel from the stencil computation domain and Gram-Schmidt kernel from the dense linear algebra domain. FDTD is a memory bound computation, i.e., number of floating point calculations done per byte of data moved from the memory is low. As such FDTD spends majority of its time waiting for the data motion from main memory to complete. Therefore, the execution time of FDTD should show relatively less degradation when run on lower clock frequencies, thereby exposing opportunities to save energy. On the flip side, Gram-schmidt is a compute bound kernel and should show poor performance when run on lower clock frequencies.

Figure 10.3 shows a series of plots for experiments conducted with these two kernels on a 16-core Intel Sandybridge node with 15 available clock frequencies. Figure 10.3(a) shows the accuracy of our power models for these two kernels. As seen in the graph, the model tracks measured power draw fairly accurately for all clock frequencies. Figure 10.3(b) shows the interaction between performance and CPU clock frequency. For Gram-schmidt, as expected, the impact that clock frequency has on performance is fairly large. Again the modeled performance impact tracks measured impact accurately. Finally, Figure 10.3(c) combines the power and performance models to predict energy usage. We see that changing hardware settings to lower power draw does not help in the case of Gram-schmidt, i.e., it is best run at the highest frequency for the lowest energy usage. For FDTD, running the kernel at 1.5GHz clock frequency as opposed to 2.6GHz saves 18% in energy usage. This gain in energy comes with 8% in performance penalty. Models also allow making energy and performance tradeoff decisions. For example, in the case of FDTD, one could choose to run the kernel at 1.7GHz. operating frequency to only allow a 5% performance penalty while still saving 16% in energy usage – tradeoff decision graphically depicted in Figure 10.3 with green and blue circles.

The current modeling technique provides fairly accurate predictions for power and performance, however, there are some outstanding issues relating to the accuracy of the models when predicting for large scale applications. We attribute this to the lack of sufficient diversity in our training set and are looking to enrich that space with more types of computations. Perhaps the issue is also related to the characterization profiles we generate for the benchmarks – we can be overlooking some important predictors. The agenda this year is to continue investigating these issues.

10.5.2 Multi-objective optimization of power, performance and energy

In the spirit of advancing collaboration within SUPER as well as advancing our own research agenda of looking into the multi-pronged energy, performance and resiliency challenges that upcoming exascale

systems will face, we have also been collaborating with ANL and UMD researchers to develop multi-objective optimization framework that can identify and exploit the tradeoffs that exist between application performance and energy consumption. We recently proposed a framework that examines a rich decision space consisting of software and hardware related parameters (e.g., compiler optimization parameters, CPU clock frequency and concurrency) on a variety of different architectures to provide evidence that tradeoffs between these competing objectives do exist. The impact of this framework will be high. In particular, the framework allows one to explore these tradeoffs will help architects and software developers make informed design space choices for the next generation systems – e.g., the framework will allow one to quantify how much performance will have to be traded to be within the mandated power cap. We refer the readers to Section 5 and Section 10.1 for more details on this collaboration.

10.5.3 Outreach

SUPER’s Energy thrust lead Laura Carrington co-chaired a Birds of a Feather session titled “Power and Energy Measurement and Modeling on the Path to Exascale” at SuperComputing 2012 conference. PI Ananta Tiwari represented SUPER on another SuperComputing BoF session title “Cool Supercomputing”.

10.6 University of Maryland

In the period since our last report, the University of Maryland effort on the Super project has concentrated in the areas of releasing a new version of Active Harmony, exploring multi-variable optimization and applying auto-tuning to MPI communication related parameters.

10.6.1 Integration

Our major integration effort has been connect auto-tuning to other performance tools. As mentioned in the last report, to enable this integration, we developed the Active Harmony plugin system. The plugin system allows both different auto-tuning search strategies as well as processing modules to be easily developed and used. In this reporting period, we have completed implementation of the plugin system and released a new version of Active Harmony. Notable from the standpoint of Super integration, the TAU plugin is now a standard part of the Active Harmony release, moving that integration effort from an experiment to production.

We also developed a new constraint plugin this period. The constraint plug allows users to express affine constraints on the relationships of parameters in the active harmony system. Constraints can be used to express things like the size of the elements for two unrolled loops should not exceed cache capacity. Our constraint system uses Pressberger arithmetic and the Omega library to efficiently constrain our search space. To make searching constrained spaces efficient, we developed a technique that uses penalization to return arbitrarily high values for the objective function at infeasible points. Infeasible points are not actually run, the constraint plugin simply returns a high value to the core search strategy. The core Harmony search strategies use a simplex (a $n+1$ point object for a n -dimensional search) to narrow in on an optimal configuration. Our penalization approach to constrained optimization allows the natural ability of the simplex-based search algorithms to work their way back to feasible regions. We decided on this approach after considering alternatives such as using a nearest neighbor algorithm to find the nearest feasible point to the proposed infeasible one.

Figure 10.4 shows the simple constraint $y < x$ and the possible ways to deal with an infeasible point (in the red region). In this example, the simplex starts as the triangle a, b_1, c . The search algorithm tries to reflect the simplex across the ac edge to form the triangle a, b_2, c . However, the point b_2 is in the infeasible region. A nearest neighbor approach would generate the point b_2' . However, our penalization approach results in the search abandoning expansion at this step and performing a contraction by replacing b with b_2'' in the simplex. The penalization helps to prevent the simplex from deforming and converging too soon.

10.6.2 Multivariate Optimization

One of the goals of enhancing auto-tuning as part of the Super project is to develop ways to handle multi-variable optimization. Example of variables that might be optimized include: runtime, compile time, power consumption, and binary size.

At Maryland, we have been exploring different ways to treat multi-variable optimization. Simple approaches such as combining the various goals into a single weighted objective function are unsatisfactory since they place too large of burden on the user to devise an appropriate hybrid objective function. Instead we are looking at an approach that allows users to express the relative importance of various objectives plus an error tolerance from optimal that they are willing to allow for each objective. We then use this information to construct a penalization function similar to the one described above for constraints. Initial implementation of this approach is complete, and trials are being conducted on synthetic objective functions. In the next six months, we will evaluate this approach using real application programs.

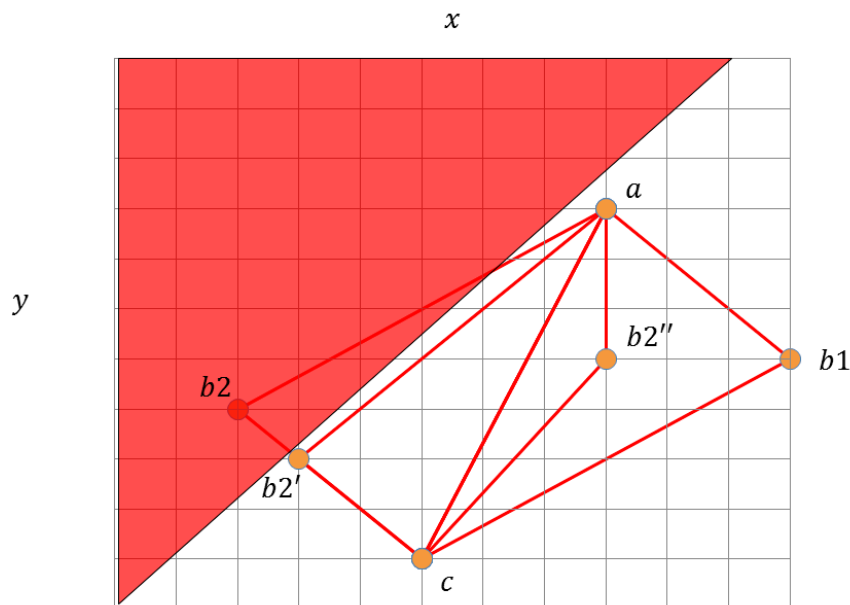


Figure 10.4: Active Harmony Constraints.

The third area that UMD has been working on is applying auto-tuning to MPI communication. In particular, we have been looking at tuning configuration parameters within implementations of MPI. Parameters such as the threshold between small, medium, and large messages can have a significant impact on the performance of a program. A second opportunity is to tune the frequency of calls into the MPI library to poll for the completion of asynchronous communication. Calling into the MPI library too often can create excessive overhead. However, calling into the MPI library not frequently enough can also slow down execution by limiting the rate at which asynchronous activities occur. With MPI 3.0, there are now asynchronous collective communication operations in addition to asynchronous point to point communication.

To explore the ability of auto-tuning to help with these communication parameters, we have been working to develop an auto-tuned version of a 3-D FFT kernel that uses asynchronous collective communication. As reported last time, our new communication optimization techniques we are able to improve performance 25% to 45% depending on the size of the input. Recently we have been working to take our initial 1-d decomposition of data to 2-d decomposition to improve strong scaling of the algorithm when scaling to larger core counts without increasing the dataset size. Initial results have shown our technique is superior for FFTW for inputs with the same number of elements in each dimension. However, work remains to be done for the case when the number of elements differs in each dimension.

10.7 University of North Carolina

We describe progress on the SUPER project at RENCI at the University of North Carolina at Chapel Hill. We begin with a summary of our work plan.

Node-wide performance measurement and tuning is becoming critical given current trends towards more cores per chip and less memory and interconnect bandwidth per core. UNC is working to develop and deploy Resource Centric Reflection tools (RCRTool) to provide node-wide measurements and utilization models for shared resources (memory, interconnects, energy) on computer architectures of interest to DOE.

The plan is to support off-line analyses suitable for inclusion in the SUPER off-line autotuning framework, as well as on-line analyses to guide introspective adaptation at several levels: operating system kernel, user-level thread packages, and adaptive applications. The “third person” offline analysis tools will be coupled with existing “first person” performance tools to provide an integrated performance model.

To provide relatively high frequency (at least 1KHz sample rate) power monitoring capabilities capable of resolving changes in system power states due to changing program phases, we have redesign our prototype PowerMon device to be suitable for wide-scale and flexible deployment in a wide range of systems, including those with accelerator devices. PowerMon data is provided through a USB interface to either the monitored system or an external monitor. With an anticipated marginal cost of less than \$200 per unit (plus the cost of cabling) when built in a batch of at least 50 units, a goal is to deploy PowerMon to SUPER researchers to the broader community of DOE researchers.

While memory, interconnect, and I/O performance have been improving, they have not kept pace with the rapid increase in the number and speed of the cores on each chip. Furthermore, with the end of Denard scaling, the speed at which cores can be run in practice is constrained by the energy budget of the package, not the design and silicon implementation of the cores themselves. For example, if only one or two cores per package are active they can be run at a speed up to 40 percent faster than if all are active. “TurboBoost” or “computational sprinting” capability under hardware control can thus have a tremendous performance advantage, but it makes performance issues much harder to understand. To meet these new challenges, SUPER researchers at UNC are working on node-wide performance tools (RCRToolkit) performance tool extensions for tracking these dynamic behaviors. These efforts necessitate real time monitoring of hardware in the “uncore” part of the chips. In contrast to past work that has focused on attributing costs to particular program constructs in individual threads, this effort will require new models and analyses suitable for system-wide performance characterization. Experimentation and prototype software is being developed for Intel Xeon X86_64, Intel MIC, and AMD chips.

A further challenge in multi-socket and multi-core systems is that because I/O and communication devices are being shared by an increasing number cores and are in danger of becoming bottlenecks. UNC will be addressing these issues at several levels. RCRTool will be used to characterize shared device utilization and correlate it with application activities. To mitigate the effects of shared resource constraints, we are extending extend our ongoing work on lightweight adaptive thread scheduling to include compile time support to optimize thread layout in NUMA systems and to provide co-scheduling, as well as to exploit cooperative dynamic adaptation between the application and the scheduler.

Although systems may be able to handle the average bandwidth required for interprocess communication and file I/O, such loads exhibit intense bursts of activity when simulations produce periodic output data sets, or checkpoint/restart files. I/O bottlenecks during bursts have the effect of serializing applications and limiting scalability. We will investigate methods and programming models at levels ranging from applications to file systems down to operating systems kernels for smoothing these loads through the use of asynchronous,

buffered operations. One variant of this approach is to use in-memory (including NVRAM) file systems for temporary files coupling multi-physics, multi-scale simulations and workflows. While in-memory storage can be very fast, it can expose the application to potential failures. We will explore mechanisms for trading off performance versus reliability in this area. We propose a co-design approach to this research in which we work with one application team that is specifically focusing on a code or codes for future systems.

10.7.1 PowerMon2

The RENCI PowerMon2 power monitoring was previously designed using funding from a variety of sources, including SciDAC-2 PERI. After hand building a prototype, using State of North Carolina funds, we built 12 more units. During the Fall of 2011, we distributed 6 of these to SUPER partner institutions (2 at UCSD, 4 at UTK). In addition, we have loaned units to non-SUPER researchers at Georgia Tech, Imperial College London, Laboratoire Informatique Grenoble, and Microsoft Research. Our current estimate of new demand is for approximately 304 more units within SUPER and as many as 60 by the other institutions. The original proposed budget for UNC under SUPER had supplies and staff salary earmarked to build additional units for use by SUPER researchers; these items were eliminated in the final budget. We have investigated the costs and methods for paying for a new fabrication run to put additional units in the hands of users. As there is a substantial setup cost and there are price breaks for the parts, including orders from, and recovering costs from, non-SUPER researchers will reduce the unit cost for all.

Based on feedback from SUPER researchers at UCSD and UTK, this year we redesigned the PowerMon2 firmware to include specific features needed for their applications. For example, to better meet the needs of UTK's effort to integrate PowerMon2 measurements with component PAPI, the firmware was extended to reduce host system overhead by extending PowerMon2 so in addition to providing instantaneous power measurements, it can integrate power over a measurement period to directly return energy consumption for that period. The firmware has also been updated to allow the device to be re-flashed in the field, thus moving software development opportunities to the community. Hardware improvements for the next fabrication batch have been designed to improve sampling rates as well as to support better communication protocols with the host to further reduce processing overhead.

At UNC, we have begun measurements of the energy consumption patterns of systems that include Nvidia GPU accelerator devices. These showed that the GPU (a Tesla 2050) can remain in a quiescent state consuming little energy until a computation is actually started on it. Furthermore, power consumption is roughly proportional to the parallelism of the computation. While power ramps up quickly, on this system the return to the low power state after the GPU computation was very slow, measured in minutes. We have begun discussions under a non-disclosure agreement with Nvidia regarding this behavior, which is characteristic of units of that generation. We have been assured that newer generation hardware and firmware does better.

UNC is collaborating with UTK, which has found funds to have a new batch of PowerMon devices built for their projects.

10.7.2 RCRToolkit

In the reporting period, we worked on packaging and documenting the Resource Centric Reflection tool (RCRTool) in preparation for testing by external friendly users.

RCRTool uses a system-wide third person approach to performance monitoring to focus on utilization (bottleneck) problems at shared resources. Attribution of performance problems, however, is fairly coarse-grained. In contrast first person libraries such as PAPI and tools built on it such as TAU, HPCToolkit, and PerfExpert cannot even access the hardware performance monitoring devices for the shared resources, but

they can provide accurate and detailed attribution of on-core costs and performance events to fragments of an application code. We developed an experimental coupling of RCRTool with Rice University's HPC-Toolkit, which is extensively used by SUPER. In this design, HPCToolkit can query information shared by RCRTool and can use that information to refine its first-person reporting. For example, performance events such as L2 cache misses can be predicated by RCRTool's model of whether or not memory is currently fully utilized, thus separating those cache misses that are solely dominated by memory latency from those that also incur queuing time at memory controllers or DRAM. The combination is also used to diagnose hitherto unanalyzed cases in which computation may be well-balanced between cores, but data allocation imbalances cause hot spots in some memory controllers while others are nearly idle. We have used the coupled tools to analyze the SciDAC USQCD Chroma code, the LBL magnetohydrionmics code, several standard benchmarks, and a health systems simulation. The results are reported in System-wide Introspection for Accurate Attribution of Performance Bottlenecks (Anirban Mandal, Rob Fowler, and Allan Porterfield), which has been submitted for publication.

RCRToolkit has been extended to monitor Intel Sandy Bridge power state information, including the RAPL model discussed in the UTK section of this report. This information is currently being used to start to extend the implications of energy and performance controls in the hardware. It is also being used as input to guide adaptive scheduling and power management in extensions to the Sandia QThreads system. The principal manifestation of the end of Denard scaling is that it is no longer possible to reduce supply voltages linearly with clock frequency. The implication of this is that dynamic voltage and frequency scaling (DVFS) no longer has as much potential to reduce power consumption as in earlier systems. Furthermore, power state changes under DVFS are relatively costly and need to be applied chip-wide. We are therefore exploring the application of "clock modulation", a mechanism that leaves frequency unchanged, but filters the clock to allow only a fraction of the cycles to be active. In addition to being very fast to apply, method can be applied on a per-core basis, thus allowing non-critical threads to run at a small fraction of the speed and power of other threads.

We have made progress with the our Resource Centric Reflection (RCR) tool suite for monitoring off-core hardware performance counters and other node-wide metrics and for computing real time models based on those metrics. In addition to performance metrics, RCRTool is used for monitoring temperature and power consumption to guide real time decisions regarding clock modulation and power states, as well as to guide fine grain thread/task scheduling. We have shared our code base with UTK and University of Oregon. At UNC, it has been installed on an Intel Sandy Bridge E5 cluster and is being used for characterizing the performance of computational chemistry and ocean circulation codes. A paper describing the application of RCRToolkit in XPRESS will appear at the ROSS13 workshop.

10.7.3 Memory System Characterization

We have continued our work characterizing memory performance using the pChase benchmark. pChase uses concurrent pointer chasing loops to characterize the effective memory latencies and bandwidths of multi-socket, multi-core systems as functions of offered load. At low loads, system response is dominated by latency. At high-loads such as might be generated by highly optimized codes that use either hardware or software prefetching, performance is dominated by bandwidth limitations. During this period, we evaluated an Intel Sandy Bridge system and a Dell system based on the AMD Interlagos chip. The latter is similar to the nodes in the latest update to Jaguar/Titan. In both cases, overall system bandwidth has improved over previous generations of these chips, though this corresponds to modest decreases in per core bandwidth. Note that in order to achieve this, we found that it is necessary to fully populate the systems with dual-rank DIMMS. Fully populating the systems also keeps the per-core memory at about 4GB/core. Failure to include enough DIMMS in the system configuration severely constrains memory performance.

10.7.4 End-to-end performance

We have worked on adaptive ensembles of simulations for a variety of inverse problems. The approach being taken is an extension to the Pegasus workflow system. This is being done in coordination with the Pegasus group at ISI and with the RENCi networking research group. A prototype of this was demonstrated in the UNC/RENCi booth at SC11. The demonstration involved a distributed ensemble running on Hopper, RENCi's internal cluster, and a cluster at Duke. In addition to reserving computational resources to rapidly run the ensemble, bandwidth was reserved on ESNET and Internet2 links between RENCi and NERSC. The current version will be demonstrated at SC12. We have begun experiments with the use of in-memory storage for temporary files in simulation pipelines and ensembles.

10.7.5 Application Outreach

We are continuing work begun under PERI and the SciDAC-e project in support of the search for catalysts for solar fuels. This application is driving system software debugging efforts, optimization exercises, and is serving as a driving application for improving the capabilities of performance tools.

The parallel spin-orbit configuration interaction (PSOCI) chemistry code developed in part under the SciDAC-e award is being applied to larger problems in electronic structure theory. The current goal is a problem of size 500 million double group adapted functions which is fairly large. Improvements to the code are occurring on Hopper at NERSC and on Jaguar at ORNL (made possible through the PEAC INCITE project). To accomplish the chemical goals requires not only computational efficiency but faster communications. PSOCI uses Global Arrays (ga++, v5) for managing the data layer. GA uses ARMCI for performing the transfer of data. Two options exist for ARMCI. The fastest (on Cray systems) is the version of ARMCI out of Argonne which uses the one-sided communications provided by MPICH2. However there is a problem. PSOCI has uncovered a failure mode where MPICH attempts an unsafe allocation that ultimately gets interpreted as a segfault; it seems more likely to be a problem in flow control. Collaborators from Argonne (J. Hammond, J. Dinan), Cray (Pritchard), and NERSC (support staff) are actively involved in setting up a runtime environment to determine if this problem is in the MPICH or Cray gemini layers. Resolution of this problem will have immediate effects for PSOCI and will ensure the problem cannot occur for other GA-based codes such as NWChem.

This effort is driving development work on new version of HPCToolKit by N. Tallent (PNNL). The improved capabilities are in the area of collection of metrics for Global Array operations and data transfer. This work is being done on Hopper.

SUPER tools are being applied in our work as part of the *Computing Properties of Hadrons, Nuclei and Nuclear Matter from Quantum Chromodynamics* SciDAC project.

10.7.6 Emerging SciDAC Engagement

During the reporting period, we began dialogues with researchers in the SciDAC QUEST Institute and attended the QUEST PI meeting at Duke University on April 6. Uncertainty quantification entails the solution of a number of inverse problems and this entails running large, dynamic ensembles of simulation and experimental data analysis codes. These computational campaigns have a lot in common with the chemical inverse design problems needed for identification of new materials such as solar catalysts. Such campaigns need to be run rapidly and robustly to boost the productivity of the researchers and to obtain timely results. The campaigns need to be computationally efficient. Our discussions with Mike Eldred of SNL are investigating their problems in this area. We are having further discussions with Omar Knio of Duke regarding UQ algorithms that may be incorporated in the search phases of materials search campaigns to optimize the

end problem by limiting the number of simulation runs needed.

At the September 2012 SciDAC PI meeting, these discussions continued. We identified the PISCEES model as a prime candidate for further work.

A group of SUPER partners (ANL, USC, and UNC) have received supplementary funding to work with physicists at FNAL, SLAC, and other sites on the particle simulation framework GEANT4. At UNC we have been analyzing a full-scale simulation of the Compact Muon Solenoid (CMS) at CERN to identify opportunities for improvement. Thus far we have determined that on this example the program executes approximately one instruction per clock cycle, which, while not horrible, still leaves adequate headroom for improvement. Analysis of memory cache performance indicates acceptable miss rates for both L1 and L2 caches for both data and instructions. While there are some data structures and regions of the code that have relatively high miss rates, these regions are contained and make only a small contribution to overall performance. We are currently working on quantifying the relative cost of the overhead of the GEANT4 object-oriented structure.

10.7.7 SUPER Education and Outreach at UNC

Stephen Olivier graduated with a Ph.D. from the UNC Computer Science Department in May 2012. His dissertation, entitled Locality Awareness for Task Parallel Computation, entailed collaborative work with the SUPER and SciDAC PERI researchers at UNC and at LLNL. Although he did not derive his stipend through SUPER (He was on a UNC Alumni Fellowship for the funding period.), he used SUPER performance tools and equipment previously purchased under SciDAC PERI for his experimental work. He is currently a staff member at SNL, Albuquerque.

Priyadarshi Sharma is a first year graduate student in the Computer Science Department at UNC-CH. He joined the SUPER effort in September and is working on node-wide performance monitoring.

10.8 University of Oregon

To date, the University of Oregon effort has continued to focus on maintaining and populating the performance database, updating, integrating and distributing associated measurement and analysis tools for the SUPER project, benchmarking partner applications and kernels, and autotuning integration. The SUPER TAUdb database continues to grow, with hundreds of new application engagement profiles added from the MULTISCALE engagement.

10.8.1 TAU enhancements

Over the last twelve months, extensive work has gone into improving the measurement of OpenMP applications with TAU. Previously, TAU has relied on source instrumentation using OPARI and POMP [25] to provide context-relevant measurement of OpenMP applications. *OpenMP Runtime API* (ORA, or "Collector API") [36] support was added to TAU, in order to tie in to OpenMP runtimes that provide that interface. ORA is a proposed specification for compilers and runtimes to provide thread state query support for sampling tools and event-based callback support for probe-based measurement tools. The list of compilers/runtimes that support ORA is limited (OpenUH, Solaris), so the TAU team implemented ORA support to the GNU OpenMP runtime (GOMP) [18] by use of a library wrapper generated by the TAU tools [46]. The wrapper intercepts the external GOMP library calls and provides the necessary state management and event callback mechanism to implement the ORA specification. In July of 2013, a new proposed tools interface specification for OpenMP 4.0 runtimes was submitted for review to the OpenMP Forum. The OpenMP Tools (OMPT) interface [17] is an update of OPARI/POMP and ORA, updating the list of events and states and better supporting newer features such as tasks. IBM has provided a reference implementation in an experimental, limited availability compiler on the BG/Q systems at Argonne. Intel open-sourced their OpenMP runtime in April of 2013 [22], and we joined an effort started by Rice University to implement OMPT support in the Intel runtime. The University of Oregon has subsequently added OMPT support to TAU, and performed extensive testing with the NAS 3.2.1, BOTS 1.1.2 and SPEC 2012 OpenMP test benchmarks. The enhanced OpenMP measurement support has been instrumental in our engagement work, as described in 10.8.3. A full survey publication of supported OpenMP methods in TAU was submitted to IPDPS 2014 [20].

Also over the last twelve months, the event based sampling support in TAU has been improved and enhanced support for *hybrid* measurement has been added. Hybrid measurements in TAU are performed with minimal, high-level instrumentation of the code combined with sampling for finer detail in uninstrumented regions such as leaf functions or library calls. The instrumentation can include but is not limited to MPI, OpenMP, I/O, static phases or dynamic phases (iterations in a time-stepped simulation). TAU provides a context-aware unwinding of the sample program stack to merge it with the TAU timer event stack. The enhanced hybrid measurement support has also been instrumental in our engagement work, as described in 10.8.3, and a paper describing the work was submitted to PDP 2014. A new collaboration within the SUPER group has recently started to integrate TAU timers into the General Purpose Timing Library (GPTL) [42] to add hybrid measurement support. GPTL timers are currently integrated into the XGC application, so this work will contribute directly to the FES/EPsi engagement effort.

TAU has added support for the RAPL [21] layer within PAPI [52] to access power and energy measurements on Intel Sandybridge systems. The power information is captured in Watts for a socket (power plane 0), the cores plus DRAM (energy package), and an embedded GPU (power plane 1). TAU has added new API calls for power profiling, tracking power usage at a given location in the code, enabling and disabling tracking power, and setting an interrupt interval. Marker events trigger when values exceed max/min observed values by a given threshold.

TAU has enhanced its CUDA and CUPTI support for GPUs with enhanced attribution of CUPTI counters

to multiple kernel executions without additional synchronization. TAU now recognizes when multiple kernel invocations occur between synchronization points, and annotates the combined measurements accordingly. New CUDA 5.0 features include GPU Kernel Tracking, where TAU tracks the number of global memory accesses and/or branches and associates it with kernel source code lines. New CUDA 5.5 features include the explicit measurement of GPU to GPU memory transfers and CUDA Dynamic Parallelism (CDP) support. Finally, TAU also has enhanced support for the Intel Xeon Phi (MIC) accelerators, adding support for PAPI hardware counters.

10.8.2 Autotuning Integration

The University of Oregon team has also integrated TAU in to the Orio autotuning framework, in a collaboration between SUPER members. The Orio autotuning framework has been extended to include TAU measurement support and integrate with the TAUdb database. We have integrated Orio with the TAU framework to provide performance hardware measurements, using the PAPI support integrated in TAU. The hardware counters enable more detailed analysis of the effects of transformations using different performance metrics other than just time. The performance measurements of each code variant are stored in the TAUdb database, providing infrastructure for both exhaustive and guided searches. A cross-language (OpenCL and CUDA) and cross-architecture comparison study was performed using the integrated frameworks, the result of which was submitted to IPDPS for publication [13]. Figure 10.5 shows the results of guided search for five target architectures when executing FormJacobian3D, a PETSc based application that uses non-linear solvers.

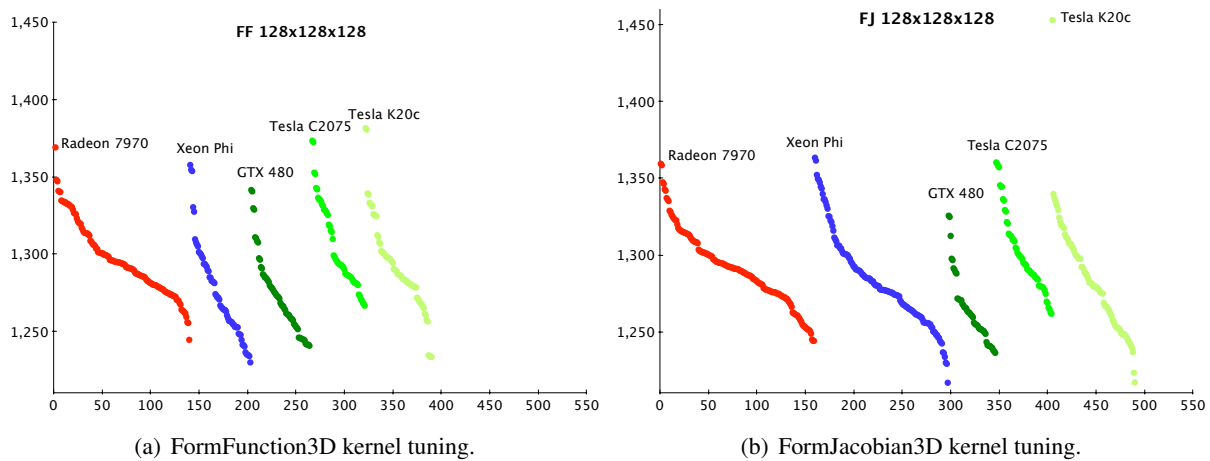


Figure 10.5: Performance (execution time in milliseconds) of evaluated variants for FormFunction3D and FormJacobian3D kernels of size 128x128x128. The sorted variant executions are represented on the x-axis, and the y-axis represents execution time.

10.8.3 Application Engagements

The team at the University of Oregon has been actively involved in performing parametric studies of the MULTISCALE application engagement partner, the MPAS-Ocean code. The MPAS-Ocean code is a CPU only code written in Fortran using MPI for parallelism. Scaling has been identified as an issue with the code, along with reducing the amount of time spent communicating. Over the last six months, the developers have been investigating OpenMP as a way to increase fine-grain parallelism within the code in order to target clusters of multicore nodes as well as new hardware such as the Intel Xeon Phi. The MPAS developers have integrated TAU measurement into the code, augmenting the existing application timers. The existing MPI measurement, enhanced OpenMP measurement, OMPT support and hybrid profiling support in TAU have been instrumental in confirming performance improvements within the code due to both OpenMP and re-

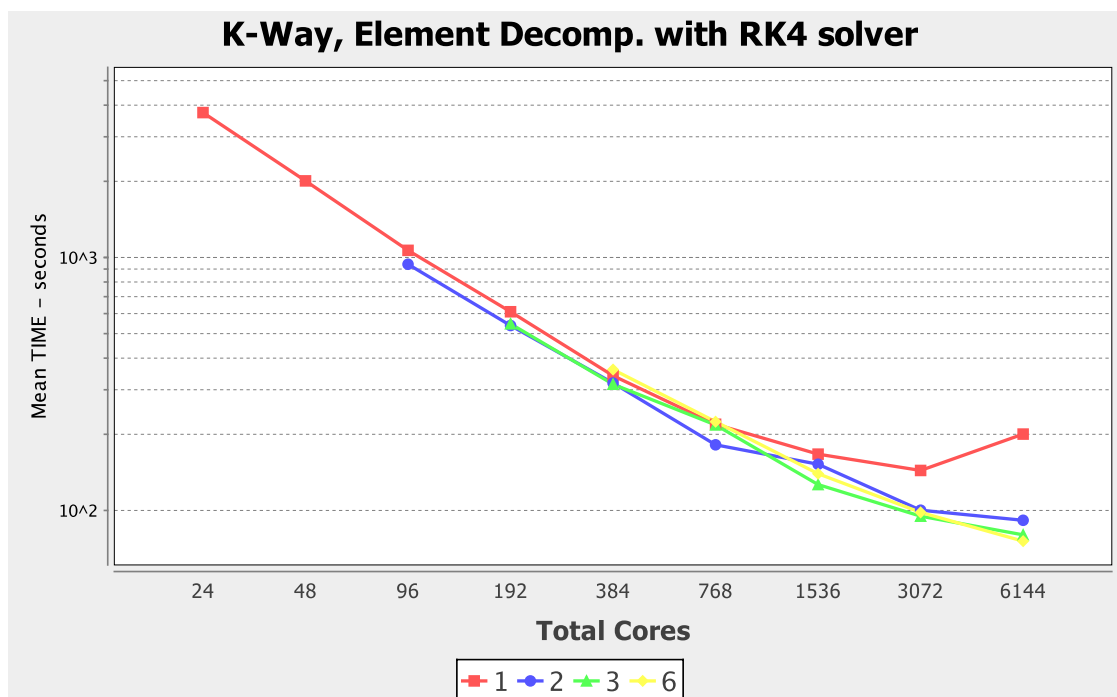


Figure 10.6: Scaling performance of MPAS-ocean using K-way data partitioning, element decomposition, the RK4 solver and a guided OpenMP schedule. Each series represents the number of threads per process. Scaling performance tails off for single-threaded executions with greater than 768 processes.

structuring to allow more vectorization. The MPAS developers recently performed a scaling study combined with a parametric study to confirm the improvements due to OpenMP in different solver implementations and decomposition methods. Figure 10.6 shows the strong scaling results from one combination. As can be seen in the figure, single threaded performance tails off at higher core counts, whereas the OpenMP-MPI combined code continues to scale.

10.9 University of Southern California

Bob Lucas of USC serves as the overall manager of the SUPER Institute, assisted by Lenny Olikier of LBNL. Management duties include setting the overall research direction, conducting teleconferences, overseeing project meetings, preparing reports and reviews, connecting to other SciDAC-3 institutes and application projects, and representing SUPER in various community activities. A detailed description of how this management function is being carried out is available in the SUPER Operating Plan. Lucas also coordinates SUPER's interactions with the other SciDAC-3 institutes. Most recently this manifested itself in a telephone conference amongst the four SciDAC-3 institute directors, who next plan to meet in person at the SciDAC-3 PI meeting in July. Interactions also included participation of investigators in each others conference calls and PI meetings.

From a technical perspective, the USC team has engaged in the SUPER resilience and SUPER autotuning teams. Under the resilience theme, We have pursued two major technical approaches, corresponding to two distinct levels of programming abstraction. This work builds on earlier research performed in the Multiscale Systems Center, funded by the Semiconductor Research Corporation's Focus Center Research Program.

Intrinsic resilience is a new concept, developed at USC, in which underutilized resources in an individual central processing unit (CPU) can be exploited to perform redundant operations, and hence detect the presence of soft errors in logic that could otherwise lead to catastrophic failure of a scientific calculation, or worse, erroneous results. Today's microprocessors can typically retire as many as eight instructions per cycle (IPC), yet in practice are lucky to achieve an IPC of one. The key is the compiler-derived schedule, which determines which operations are performed, in what order, for the architecture of a specific CPU. This schedule is unique to each application, the CPU it's compiled for, and the level of optimization chosen. Pipeline bubbles and other idle resources can be exploited to detect soft-errors with pairing, or even correct them with triple-modular redundancy. The figure below depicts an automated analysis generated by a tool developed in the context of this project. Developers can direct the compiler to generate schedules that trade-off execution time (performance) for redundant operation coverage (resilience). This compiler approach relies on the use of a hybrid-TMR concept as described in detail in one of our FTP publications.

USC is also studying the question of how a user's knowledge of the ability of a scientific program to tolerate soft faults in logic can be exploited to reduce the number of times such errors unnecessarily halt a computation. We are motivated by the fact that there are many examples of applications today that tolerate errors and successfully complete, often without the knowledge of the user. We have initially focused our research on uncorrectable, double-bit failures in memory. We have developed type qualifiers, and other mechanisms for identifying regions of memory where errors can be explicitly tolerated. We have created a test bed in which faults can be randomly injected into codes, and shown that four small benchmarks and kernels can survive those errors that hit the tolerant sections of memory and run successfully to completion. As a way to promote the early experimentation and hence adoption of these techniques we have engaged other researchers in the SUPER community, in particular, Bronis de Supinski and Marc Casas Guix at LLNL as well as Prof. Ganesh Gopalakrishnan at the University of Utah. A paper based on this work has been submitted to SC13.

Along with the autotuning team, USC is studying the performance of the USQCD code, and in particular of a SU(3) QCD solver where the basic objects are discretized 3D complex vector fields. The solver has been manually optimized by its developers and in its current state shows few optimization opportunities. However there are optimization opportunities once the basic data structures are reorganized, and USC is developing versions of the solver which use different representations of the 3D complex vectors. Once these versions are completed they will be further optimized with SUPER optimization tools.

In the next six months of SUPER, USC will continue to direct the overall institute, and contribute to research in autotuning and resilience. This will include organizing telephone calls, meetings hosted by LBNL and Utah, and opportunistic meetings at venues such as SC13 and the SciDAC-3 PI meeting. Our performance engineering research will focus on new compiler decision algorithms, further integration with modeling, and integration into the autotuning system. This work is an ongoing collaboration with ANL, LLNL, Maryland, Oregon, Rice, and Utah. Our resilience research will be directed towards a tighter collaboration with SUPER colleagues at Lawrence Livermore National Laboratory. Bronis de Supinski leads the overall SUPER resilience theme, and is investigating the resilience of DOE applications. Our resilience API will provide him with a way to express the knowledge he gains. Our research will now turn to the question of how an introspective system can react to this knowledge, and prevent applications from failing unnecessarily.

Finally, Dr. Jeremy Abramson defended his thesis entitled “Resiliency-aware Scheduling” in January 2013 and graduated on May of 2013. The last year of his research was supported by SUPER and directed by Pedro Diniz. His thesis work has focused on the development of a novel resiliency-aware scheduling algorithm targeting both FPGA-based computing engines as well as VLIW architectures. Resiliency-aware scheduling combines traditional compiler techniques such as critical path and dependency analysis with the ability to potentially modify the target architectures resource configuration. This new approach can, in many cases, offer operational coverage similar to traditional schemes, while enjoying a performance advantage and area savings.

10.10 University of Tennessee, Knoxville

During this reporting period, UTK focused primarily on activities related to three thrust areas of SUPER: performance engineering, energy minimization and resilience. These areas will be discussed in further detail below.

10.10.1 Performance Measurement

The primary vehicle through which UTK provides performance measurement support to SUPER and the larger HPC community is the Performance API (PAPI), an ongoing project providing access to hardware performance counters on cpus and other hardware components of interest. PAPI 5.2 was released in August 2013, providing initial support for Intel Haswell and IBM POWER8 processors. Intel MIC (Xeon Phi) support has been hardened and PAPI support for the offload pragmas for MIC software development has been incorporated. We have worked with the TAU group and with ORNL researchers to test and validate this important feature enhancement. Other enhancements to PAPI 5.2 include runtime loading of external libraries for components. This further enhances PAPI's portability and maintenance on heterogeneous hardware platforms.

Another PAPI release, 5.3, is anticipated in advance of this year's SuperComputing Conference in mid-November. This release will include further enhancements to support for MIC, bug fixes in the support of the Blue Gene / Q networking capability, and better support for end user reuse of our example code base.

Although no longer directly funded by SUPER, the development of the MIAMI performance tool has continued through the efforts of Gabriel Marin. He has recently made a beta version of the tool available to "friendly users" within the SUPER community to begin using it for real-world applications.

Future directions for PAPI development activities include support for ARM64 architectures, which should be of particular interest for the HPC community, support for AMD northbridge counters that measure system level performance characteristics, and broader support for I/O measurement in virtual environments.

10.10.2 Energy Measurement

Power and energy measurement activities continue to be of importance for both PAPI and the SUPER research thrust.

The PAPI 5.1 release discussed in the previous report provided access to an on-board power sensor on the MIC card which allows measurement of current and voltage (and computed power) for various subsystems on the MIC card at roughly 50 μ s resolution. The PAPI 5.2 release provides additional support for reading MIC power from the host cpu. This makes it much easier to measure power consumption of MIC code at fairly high resolution without actually instrumenting the MIC code directly. We are currently working to validate and verify the power measurements obtained from this instrumentation to insure that it correlates with software activity on the hardware.

We reported earlier that we were in conversation with IBM engineers to implement a PAPI interface for a high speed power measurement API for Blue Gene Q, called EMON2. This API provides integrated power and energy measurements at the node level. After some frustration at obtaining access to the required resources, we are making progress toward a PAPI implementation of this code. We are working with systems at LLNL (Sequoia), Argonne (Mira), and Juelich (JuQueen) to develop and test a PAPI interface for this API.

10.10.3 Resilience

UTK's approach to resilience involves the combination of multiple existing techniques, ranging from checkpoint/restart fault-tolerance, to automatically tolerating failures, to algorithm-specific approaches, that involve resilient middleware, and resilient algorithms that exploit internal redundancy to detect and correct failures. In this light, we continued our effort on the Resilience area in complementary directions.

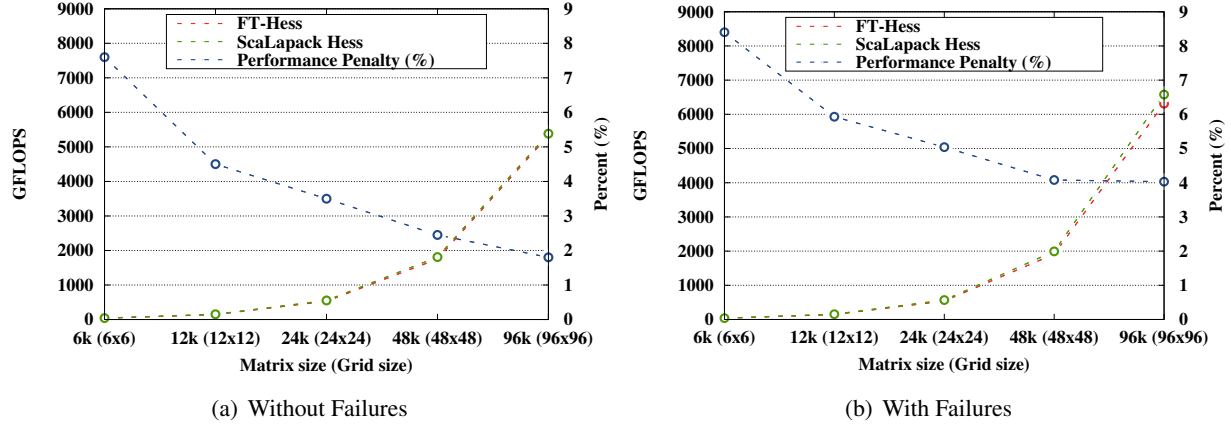
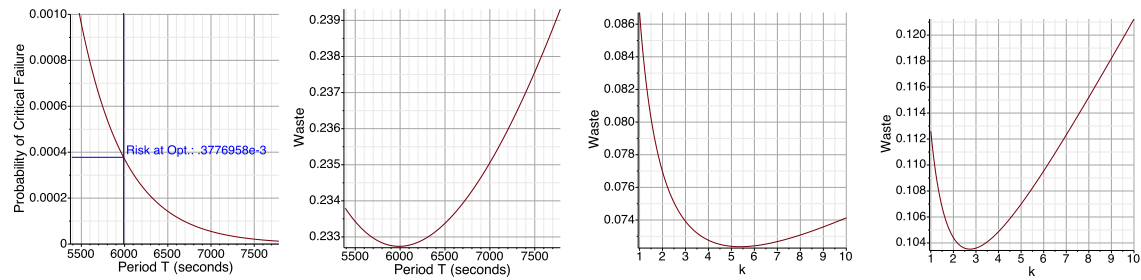


Figure 10.7: Overhead of FT-Hessenberg without failures and with one failure. Platform: Titan, NB = 80

Motivated by predictions of a unifying performance model of traditional rollback recovery techniques that we designed to project the efficiency of these techniques at extreme scale ([10]), we pursued the development and evaluation of the fault-tolerant extension proposed to the MPI standard, ULFM [8], upon which we built a new algorithm-based fault-tolerant (ABFT) scheme for the Hessenberg Reduction [23]. This algorithm combines a traditional ABFT technique with diskless checkpointing to fully protect the data: the trailing and proceeding matrix is protected with checksums, while the finished panels in the panel scope are covered by diskless checkpointing. Performance modeling shows that the overheads follow a decreasing trend as the size of the matrix or the size of the process grid increases. Experiments on the Cray XT6 evaluate these failure-free overheads to be as low as 1.8% for a $96,000 \times 96,000$ matrix on a 96×96 process grid (see Figure 10.7(a)). Failure injection (see Figure 10.7(b)) demonstrates a small penalty, linear with the number of failures.

Considering the case of silent errors (memory corruptions that are not corrected, or signaled, by the hardware), we evaluated in [1] how rollback recovery strategies can be combined with checkers to minimize the waste induced by such errors. Assuming there is no sure way to validate the consistency, of the data, except by continuing the computation to its end, we evaluate how the risk is linked with the number of checkpoints kept, and the period of these checkpoints. As illustrated in Figures 10.8(a), and 10.8(b), if the 3 last checkpoints are kept, on a typical platform (MTBF of each component of 10 years, number of components 10^5 , and costs of checkpoint and rollback of 10 minutes), the optimal period, with respect to waste, induces a risk of unrecoverable failure (i.e. all the checkpoints kept are corrupted) of 4×10^{-3} . By increasing the period between checkpoints, the waste increases, but the risk decreases significantly. Then, we illustrated how checksumming techniques, like the ones developed in ABFT, can be utilized to detect the silent errors, providing a checker to validate the data, and build a performance model that predicts the optimal number of checkpoints and validations, depending on the characteristics of the platform, the application and the relative costs of validation and checkpointing. As illustrated in Figures 10.8(c), and 10.8(d), depending on these relative costs, different frequencies of checkpoint and verification provide different minimal waste. Because the data can always be validated before it is checkpointed, the risk of unrecoverable scenarios is



(a) Risk of unrecoverable failure, when keeping only the last 3 checkpoints, as function of the checkpointing period	(b) Waste due to Fault-Tolerance, when keeping the last 3 checkpoints, as function of the checkpointing period	(c) Waste due to Fault-Tolerance, as function of the number of validations / checkpoints, with k validations of the data per checkpoint; Cost preemptive checkpoint / Cost Validation = 1/30	(d) Waste due to Fault-Tolerance, as function of the number of validations / checkpoints, with k checkpoints between each validation; Cost Validation / Cost Checkpoint = 16

Figure 10.8: Using rollback/recovery and checkers, to tolerate silent error corruptions.

entirely avoided in that case.

10.11 University of Texas El Paso

Funding at the University of Texas at El Paso (UTEP) began September 1, 2012. Progress so far at UTEP has involved hiring two graduate students to work on the project, getting the graduate students up-to-speed, and collaborating with colleagues on performance modeling and on automated program transformations and performance tuning. Initial work in the performance and energy areas is described below. UTEP has also continued to maintain the SUPER website and help publish the SUPER newsletter.

10.11.1 Performance

Work in the performance area has focused on four topics. One topic is the use of automated program transformation tools to outline performance-critical kernels and to perform automated loop transformation. Tools being investigated include the ROSE outliner [29] for C/C++ codes, the Photran (www.eclipse.org/photran) plug-in for the Eclipse Parallel Tool Platform (PTP) for Fortran codes, and the PerfExpert and MACPO tools from Texas Advanced Computing Center [11, 41]. We are currently installing these tools and testing them with the provided examples and with computational chemistry applications. We attempted to use the ROSE outliner to extract kernels from the MPQC computational chemistry code and the C++ version of the GAMESS computational chemistry code. We have been unsuccessful so far, but we are working with the Rose developers to resolve the problems we have encountered. We have been successful in working together with colleagues at TACC to integrate ROSE-based program transformations with PerfExpert and MACPO recommendations so as to implement automated loop transformations. We are applying this approach to computational chemistry applications (integral computations in NWChem and Hartree Fock computations in GAMESS) and expect to publish results soon. The goal is to eventually integrate PerfExpert and MACPO with the SUPER auto-tuning framework as additional third-party tools.

A second topic is using the Tuna command-line tuning application to tune parameters for computational chemistry codes GAMESS+TigerCI and MPQC. For GAMESS+TigerCI, these parameters include OpenMP settings such as chunk size and application-specific parameters such as buffer sizes. We have been working with University of Maryland to get Tuna working with GAMESS, and we will be placing the results in a TAUdb data base. We have previously used handwritten scripts to search the space of application-specific parameters settings for MPQC, and we are attempting to modify the MPQC code so that this search can be carried out by Tuna.

A third topic is using machine-independent data access metrics such as reuse distance and strides to model the cache performance of applications. We have been working with the PEBIL-based data access measurement tool from SDSC and with MACPO from TACC. So far we have written micro-benchmarks to validate the results reported by these tools, and we have also instrumented our micro-benchmarks with PAPI to sanity-check the results. Our micro-benchmarks carry out simple vector and matrix operations for which we can determine the expected results. We have found bugs in both tools and have reported them to the tool developers. SDSC has delivered a fixed version of their tool that we are now testing. The MACPO team is working on the bug fixes. Once we are convinced that the tools report correct results, we will apply them to the XGC-1 and QCD kernels to extend the analyses already carried out by other SUPER researchers.

A fourth topic is using profiling and data access analysis tools to identify portions of applications amenable to GPU implementation and to predict the GPU performance. Profiling tools we are investigating for this purpose include TAU [45] and PerfExpert [11]. Data access analysis tools include MACPO [41] and a similar tool under development at SDSC. The idea is to carry out the following steps:

1. Optimize for multicore. Optimization for multicore tends to generate SPMD and streaming parallelism/vectorization that are directly mappable to SIMD/SIMT execution. Additionally, optimization for

multicore generates good memory locality.

2. Identify the most important (time consuming) kernels in the optimized code.
3. Eliminate those time consuming kernels that are not straightforwardly mappable to SIMT/SIMD execution because of frequent TLB misses, high fraction of branches, cache conflicts across cores, or irregular access strides to important data structures. item Characterize the following execution behaviors of those code segments that are straightforwardly mappable to SIMT/SIMD execution in order to assess the speedup to be obtained:
 - a) Computational intensity
 - b) Pure SPMD parallelism
 - c) Streaming parallelism/vectorization
 - d) Regular access strides through data structures
 - e) Data reuse and data transfer volume.

The idea is to have our analysis feed into the SUPER auto-tuning framework which could then generate and evaluate possible GPU implementations.

We have not worked on this topic since our previous report, because the graduate students have been getting up-to-speed on GPU programming. They are now ready to begin work on this topic during the summer and apply the techniques to computational chemistry codes and the the XGC-1 kernel.

10.11.2 Energy

We have reviewed our previous work on using cycle breakdown models [37] and linear regression models [30], both of which use hardware performance counter data, to construct predictive power-performance models of applications. Cycle breakdown models suffer from the weakness that the counters needed are not always available. Linear regression models suffer from the weakness that they are very dependent on the training data. We are currently reviewing the literature on application of machine learning techniques to this problem to determine if this would be a fruitful avenue of research. We have not worked on this topic since our previous report, but we plan to return to it this summer.

10.12 University of Utah

In the second half of the second year, the University of Utah team focused on five key issues: (1) improved capability of CodeGen+ to handle non-affine codes; (2) initial analysis of MPAS application code on Edison; (3) detailed analysis of a key computational kernel from QCD. (4) analysis of XGC code; and, (5) identify highly vulnerable fault sites in programs.

10.12.1 Enhancements to CodeGen+.

We have enhanced the code generation capabilities of CodeGen+ to support: (i) non-affine loop bounds, and (ii) non-affine transformations. With the capability to handle non-affine loop bounds, CodeGen+ can be used in conjunction with CHiLL and CUDA-CHiLL to tune applications involving sparse computations. The support for non-affine transformations in CodeGen+ provides users with an abstraction to express runtime inspection routines and data structures in a polyhedral framework setting. Additionally, the code generation statement macro interface was extended to update array subscripts correctly during the application of non-affine transformation.

10.12.2 Initial analysis of MPAS code.

The Model for Prediction Across Scales (MPAS) code (need to stress its importance) is written in Fortran and MPI and is used in weather studies. The code uses unstructured grids, leading to indirect addressing. Further, extensive use of array notation, structures, and pointers, makes the code difficult to analyze using existing compiler tools. As a step towards understanding the performance behavior of the MPAS code, we have successfully evaluated its performance on Edison for two small test inputs, `overflow_1km_100layer` and `baroclinic_channel_4000n_20levs`. Our initial experiments indicate that the code has sub-linear speedup and most of execution time is spent in a few function calls (`se_btr_vel`, `adv`, `MPI_Wait` and `Se_timestep`). We also noticed degradation in performance when the MPI processes were distributed across different nodes instead of running on a single node. Although we sense an opportunity to improve and scale MPAS code performance, we would need further analysis to identify the data access patterns of the code and assert if such access patterns are amenable to locality and parallelization optimizations.

10.12.3 Detailed analysis of QCD code.

TODO

10.12.4 Analysis of XGC code.

Did some analysis, but did not lead to any performance improvements. NEED TO EXPLAIN

10.12.5 Resilience: Current and Future Work.

The Utah team is investigating ways to identify highly vulnerable fault sites in programs. In particular, we are investigating the use of machine-learning techniques to optimize the placement of invariant-based error detectors. Static analysis methods are also under consideration to determine how errors may propagate. The overall objective is to minimize error detector induced overheads without sacrificing coverage.

A. References

- [1] G. Aupy, A. Benoit, T. Herault, Y. Robert, F. Vivien, and D. Zaidouni. On the combination of silent error detection and checkpointing. In *Proceedings of the 19th IEEE Pacific Rim International Symposium on Dependable Computing, PRDC'13*, Vancouver, Canada, December 2013.
- [2] P. Balaprakash, S. M. Wild, and B. Norris. SPAPT: Search problems in automatic performance tuning. *Procedia Computer Science*, 9:1959–1968, 2012. doi: 10.1016/j.procs.2012.04.214.
- [3] P. Balaprakash, R. Gramacy, and S. M. Wild. Active-learning-based surrogate models for empirical performance tuning. In *Proceedings of IEEE Cluster*, September 2013. Available at <http://www.mcs.anl.gov/papers/P4073-0513.pdf>.
- [4] P. Balaprakash, K. Rupp, A. Mametjanov, R. B. Gramacy, P. D. Hovland, and S. M. Wild. Empirical performance modeling of gpu kernels using active learning. Preprint ANL/MCS-P4097-0713, Argonne National Laboratory, Mathematics and Computer Science Division, July 2013.
- [5] P. Balaprakash, A. Tiwari, and S. M. Wild. Multi-objective optimization of HPC kernels for performance, power, and energy. In *Proceedings of the 4th International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS13)*, November 2013. To appear, available at <http://www.mcs.anl.gov/papers/P4069-0413.pdf>.
- [6] P. Balaprakash, A. Tiwari, and S. M. Wild. Framework for optimizing power, energy, and performance. In *Posters of the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC13)*, November 2013. To appear.
- [7] P. Balaprakash, S. M. Wild, and P. D. Hovland. An experimental study of global and local search algorithms in empirical performance tuning. In *High Performance Computing for Computational Science - VECPAR 2012, 10th International Conference, Kobe, Japan, July 17-20, 2012, Revised Selected Papers.*, Lecture Notes in Computer Science, pages pp. 261–269. Springer, 2013. ISBN 978-3-642-38717-3. doi: 10.1007/978-3-642-38718-0_26.
- [8] W. Bland, A. Bouteiller, T. Herault, J. Hursey, G. Bosilca, and J. Dongarra. An evaluation of user-level failure mitigation support in MPI. *Computing*, May 2013. doi: 10.1007/s00607-013-0331-3.
- [9] S. Borkar and A. A. Chien. The future of microprocessors. *Communications of the ACM*, 54(5):67–77, May 2011.
- [10] G. Bosilca, A. Bouteiller, E. Brunet, F. Cappello, J. Dongarra, A. Guermouche, T. Herault, Y. Robert, F. Vivien, and D. Zaidouni. Unified model for assessing checkpointing protocols at extreme-scale. *Concurrency and Computation*, 2013. To Appear.
- [11] M. Burtscher, B. Kim, J. Diamond, J. McCalpin, L. Koesterke, and J. Browne. Perfexpert: An easy-to-use performance diagnosis tool for HPC applications. In *SC'10 International Conference for High-Performance Computing, Networking, Storage and Analysis*, Nov. 2010.
- [12] L. Carrington, M. M. Tikir, C. Olschanowsky, M. Laurenzano, J. Peraza, A. Snively, and S. Poole. An idiom-finding tool for increasing productivity of accelerators. In *Proceedings of the international conference on Supercomputing, ICS '11*, pages 202–212, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0102-2. doi: 10.1145/1995896.1995928. URL <http://doi.acm.org/10.1145/1995896.1995928>.

- [13] N. Chaimov, B. Norris, and A. Malony. Multi-target autotuning for accelerators. *submitted*, 2014.
- [14] B. D. F. R. Choi, J.W. and R. Vuduc. A roofline model of energy. In *IPDPS 2013*, 2013.
- [15] C. Choudary, J. Godwin, J. Holewinski, D. Karthik, D. Lowell, A. Mameetjanov, B. Norris, G. Sabin, P. Sadayappan, and J. Sarich. Stencil-aware GPU optimization of iterative solvers. *SIAM Journal on Scientific Computing*, 2013. URL <http://www.mcs.anl.gov/uploads/cels/papers/P3008-0712.pdf>. Also available as Preprint ANL/MCS-P3008-0712.
- [16] T. A. Davis. *Direct methods for sparse linear systems*, volume 2. SIAM, 2006.
- [17] A. Eichenberger, J. Mellor-Crummey, M. Schulz, N. Copt, J. DelSignore, R. Dietrich, X. Liu, E. Loh, and D. Lorenz. OMPT and OMPD: OpenMP Tools Application Programming Interfaces for Performance Analysis and Debugging. April 2013.
- [18] GNU. GNU libgomp. <http://gcc.gnu.org/onlinedocs/libgomp/>, 2013.
- [19] M. A. Heroux, D. W. Doerer, P. S. Crozier, and J. M. Willenbring. Improving performance via mini-applications. Technical Report SAND2009-5574, Sandia National Laboratories, September 2009.
- [20] K. Huck, S. Shende, A. Malony, and D. Jacobsen. Integrated measurement for cross-platform OpenMP performance analysis. *submitted*, 2014.
- [21] Intel. *Intel® 64 and IA-32 Architectures Software Developer’s Manual*, volume 3B: System Programming Guide, Part 2. Intel, June 2013.
- [22] Intel. Intel open source OpenMP runtime. www.openmpRTL.org, 2013.
- [23] Y. Jia, G. Bosilca, P. Luszczek, and J. J. Dongarra. Parallel reduction to hessenberg form with algorithm-based fault tolerance. In *Proceedings of Supercomputing’13*, 2013. to appear.
- [24] A. Kaiser, S. Williams, K. Madduri, K. Ibrahim, D. Bailey, J. Demmel, and E. Strohmaier. TORCH computational reference kernels: A testbed for computer science research. Technical Report UCB/EECS-2010-144, EECS Department, University of California, Berkeley, December 2010. URL <http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-144.html>.
- [25] A. Knüpfer, C. Rössel, D. an Mey, S. Biersdorff, K. Diethelm, D. Eschweiler, M. Geimer, M. Gerndt, D. Lorenz, A. Malony, W. Nagel, Y. Oleynik, P. Philippen, P. Saviankou, D. Schmidl, S. Shende, R. Tschüter, M. Wagner, B. Wesarg, and F. Wolf. Score-P - A Joint Performance Measurement Runtime Infrastructure for Periscope, Scalasca, TAU, and Vampir. In *Proc. 5th Parallel Tools Workshop*, pages 79–91. Springer, 2012.
- [26] M. Laurenzano, M. Tikir, L. Carrington, and A. Snively. Pebil: Efficient static binary instrumentation for linux. In *Performance Analysis of Systems Software (ISPASS), 2010 IEEE International Symposium on*, pages 175–183, 2010. doi: 10.1109/ISPASS.2010.5452024.
- [27] M. A. Laurenzano, M. Meswani, L. Carrington, A. Snively, M. M. Tikir, and S. Poole. Reducing energy usage with memory and computation-aware dynamic frequency scaling. In *Proceedings of the 17th international conference on Parallel processing*, Euro-Par’11, pages 79–90, 2011. ISBN 978-3-642-23399-9. URL <http://dl.acm.org/citation.cfm?id=2033345.2033356>.
- [28] Lawrence Livermore National Laboratory. ROSE compiler infrastructure. <http://rosecompiler.org>, April 2013.

- [29] C. Liao, D. J. Quinlan, R. W. Vuduc, and T. Panas. Effective source-to-source outlining to support whole program empirical optimization. In *Workshop on Languages and Compilers for Parallel Computing (LCPC)*, pages 308–322, University of Delaware, 2009.
- [30] C. Lively, X. Wu, V. Taylor, S. Moore, H.-C. Chang, C.-Y. Su, and K. Cameron. Power-aware predictive models of hybrid (MPI/OpenMP) scientific applications on multicore systems. In *International Conference on Energy-Aware High Performance Computing (EnA-HPC 2011)*, Hamburg, Germany, Sept. 2011.
- [31] LLNL. Livermore loops benchmark. http://www.wikipedia.org/Livermore_loops.
- [32] A. Malony, M. Hall, J. Hollingsworth, B. Norris, and G. Marin. Tool integration and autotuning for super performance optimization. Poster presentation at DOE SciDAC Fall PI Meeting, September 2012.
- [33] A. Mametjanov and B. Norris. Autotuning of Vectorization in Stencil Computations. Workshop on Optimizing Stencil Computations (WOSC’13), Indianapolis, Indiana, October 2013.
- [34] A. Mametjanov, D. Lowell, C.-C. Ma, and B. Norris. Autotuning stencil-based computations on GPUs. In *Proceedings of IEEE Cluster 2012*, Sep. 2012. URL <http://www.mcs.anl.gov/uploads/cels/papers/P2094-0512.pdf>. Also available as Preprint ANL/MCS-P2094-0512.
- [35] A. Mandal, R. Fowler, and A. Porterfield. System-wide introspection for accurate attribution of performance bottlenecks. In *Workshop on High-performance Infrastructure for Scalable Tools (WHIST)*, Venice, Italy, June 2012.
- [36] Marty Itzkowitz, Oleg Mazurov, Nawal Copt, Yuan Lin, Sun Microsystems. An openmp runtime api for profiling. <http://www.oracle.com/technetwork/server-storage/solaris10/omp-api-141059.html>, February 2007.
- [37] S. Moore and J. Ralph. User-defined events for hardware performance monitoring. In *International Conference on Computational Science (ICCS)*, Singapore, June 2011.
- [38] B. Norris, A. Hartono, and W. Gropp. Annotations for productivity and performance portability. In *Petascale Computing: Algorithms and Applications*, Computational Science, pages 443–462. Chapman & Hall / CRC Press, 2007. URL <http://www.mcs.anl.gov/uploads/cels/papers/P1392.pdf>.
- [39] J. Peraza, A. Tiwari, M. Laurenzano, L. Carrington, and A. Snavely. Pmac’s green queue: A framework for selecting energy optimal dvfs configurations in large scale mpi applications. In *Submission to CCPE Special Issue on Analysis of Performance and Power for Highly Parallel Systems*.
- [40] A. Porterfield, R. Fowler, S. Balachandra, and W. Wang. OpenMP and MPI application energy measurement variation. Nov. 2013.
- [41] A. Rane and J. Browne. Enhancing performance optimization of multicore chips and multichip nodes with data structure metrics. In *21st International Conference on Parallel Architectures and Compilation Techniques (PACT 2012)*, Minneapolis, MN, Sept. 2012.
- [42] J. Rosinski. GPTL - General Purpose Timing Library. <http://jmrosinski.github.io/GPTL/>, 2013.

- [43] P. C. Roth. Tracking a value’s influence on later computation. In D. an Mey et al., editors, *Proceedings of the 6th Workshop on Productivity and Performance (PROPER 2013)*, Aachen, Germany, Aug. 2013.
- [44] V. Sharma and A. Haran. A Kontrollable Utah LLVM Fault Injector (KULFI), 2013. <https://github.com/vcsharma/KULFI>.
- [45] S. Shende and A. Malony. The TAU Parallel Performance System. *International Journal of High Performance Computing Applications*, 20(2, Summer):287–311, 2006. ACTS Collection Special Issue.
- [46] S. Shende, A. D. Malony, W. Spear, , and K. Schuchardt. Characterizing I/O Performance Using the TAU Performance System. *Exascale Mini-symposium, ParCo 2011*, 2011.
- [47] E. Solomonik, D. Matthews, J. Hammond, and J. Demmel. Cyclops tensor framework: reducing communication and eliminating load imbalance in massively parallel contractions. Technical Report UCB/EECS-2013-11, EECS Department, University of California, Berkeley, Feb 2013. URL <http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-11.html>.
- [48] A. Tiwari, J. Peraza, M. Laurenzano, L. Carrington, and A. Snively. Green queue: Customized large-scale clock frequency scaling. In *Proceedings of the Second International Conference on Cloud and Green Computing, CGC ’12*, 2012.
- [49] University of Maryland and University of Wisconsin. Paradyn/dyninst - putting the performance in high performance computing. <http://www.dyninst.org>, April 2013.
- [50] University of Oregon. PDT - Program Database Toolkit. <http://www.cs.uoregon.edu/research/pdt/>, April 2013.
- [51] University of Oregon. TAU - Tuning and Analysis Utilities. http://tau.uoregon.edu/tau_releases, April 2013.
- [52] University of Tennessee Knoxville. Performance Application Programming Interface. <http://icl.cs.utk.edu/papi/>, 2013 April.
- [53] J. S. Vetter, S. Lee, D. Li, G. Marin, C. McCurdy, J. Meredith, P. C. Roth, and K. Spafford. Quantifying architectural requirements of contemporary extreme-scale scientific applications. In *Proceedings of the 6th International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS13)*, Denver, Colorado, Nov. 2013 (to appear).