Institute for Sustained Performance, Energy, and Resilience (SUPER): University of Utah Final Report

Mary Hall (Utah PI)
Ganesh Gopalakrishnan (Senior Personnel)

DE-SC0008682
DOE Scientific Discovery through Advanced Computing (SciDAC) Program

Over the course of this project, computational scientists working on behalf of the Department of Energy's Office of Science (DOE SC) were exploiting a new generation of petascale computing resources to make previously inaccessible discoveries in a broad range of disciplines including physics, chemistry and material science. The computational systems underpinning this work increased in performance potential from tens to hundreds of Pflop/s, and evolved significantly from those in use at the beginning of the project in terms of available concurrency; wider use of accelerators such as graphics processing units(GPUs) ; and changes to the memory hierarchy including the incorporation of a new generation of persistent devices (e.g., phase change memory).

To ensure that DOE's computational scientists can successfully exploit this emerging generation of high performance computing (HPC) systems, the University of Southern California (USC) assembled a broad team of computer scientists with the expertise to address the most pressing challenges of DOE's computing systems: (a) end-to-end performance optimization, including single-node performance, interprocessor communication, load balancing and I/O; (b) performance portability for heterogeneous systems, including accelerators; (c) minimizing energy consumption; (d) resilient computation; and, (e) exploiting new memory technology.

The leadership on SUPER organized a broadly-based project with expertise in compilers and other system tools, performance engineering, energy management, and resilience. This comprehensive coverage allowed us to integrate tools from numerous researchers, both within our institute and in other SciDAC-3 computer science institutes. We worked 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. It is a collaboration with University of Southern California (lead), Lawrence Berkeley National Laboratory (co-lead), Lawrence Livermore National Laboratory, Argonne National Laboratory, Oak Ridge National Laboratory, Massachusetts Instute of Technology, University of California San Diego, University of Maryland, University of North Carolina, University of Oregon, and University of Tennessee.

Utah's contribution to SUPER was led by Mary Hall (Utah PI), who served as the area lead for autotuning and performance optimization. Within Hall's research group, her work on SUPER extended autotuning technology, and the CHiLL and CUDA-CHiLL transformation, code generation and autotuning compiler. Senior Personnel Ganesh Gopalakrishnan led Utah's contribution to resilience research, with the development of FUSED and SORREL to apply formal methods to detect errors in codes.

Final Report: SciDAC3 SUPER Performance Institute*

1 Introduction

The SciDAC-3 Institute for Sustained Performance, Energy, and Resilience (SUPER) focused on developing technology and leveraging expertise towards ensuring that computational scientists could successfully exploit existing and emerging DOE high performance computing systems. To achieve this goal, SUPER was organized into research thrusts focusing on performance engineering, auto-tuning, performance tools, energy efficiency, resilience and multi-objective optimization, with a target of facilitating effective utilization of near term HPC systems. As described in this summary document, our teams' significant insights into both application- and architecture-awareness drove numerous innovations across our stated research areas, while significantly improving the performance, portability, and functionality of numerous SciDAC partnership applications, enabling scientific simulation at unprecedented scale. Additionally, the tools, methodologies, and collaborations developed under this project will continue paying dividends as we face the challenges of forthcoming multi-petascale and exascale computing technologies.

2 Performance Engineering

At the core of SUPER's performance engineering was its engagement with the SciDAC applications. This included 16 funded partnerships (SUPER researchers received partnership funding) and several unfunded collaborations (funded exclusively through SUPER). In addition, we found it necessary to supplement several funded partnerships with augmentations or SUPER funds in order to provide sufficient resources.

Although there were some common challenges and solutions across applications, we were often required to employ a menagerie of application-specific code transformations, threading enhancements, GPU-acceleration, changes to data structures, leveraging distributed memory, and fundamental algorithmic transformations. We highlight some of the more substantial performance engineering efforts below.

In order to mitigate performance penalties that arise from non-contiguous node allocation on torii, we developed techniques to identify suitable mappings of processes to allocated nodes. This involved instrumentation and collection of application performance data using mpiP. Accelerated with our mpiAproxy tool, we evaluated several reordering algorithms for MPAS-O, ACES4BGC, and XGC1 on Titan [65].

We developed a number of performance solutions for the fusion code CGYRO (ATOM SAP). These included providing additional scratch memory for the vendor-tuned BLAS routines (enabled better blocked implementations), the use of OpenACC and cuFFT for GPU-acceleration, and the replacing of multiple MPI collectives with point-to-point communication. Overall, these improved performance by more than $3\times$.

Instrumentation and analysis of the EPSi FES SAP code XGC1 identified bottlenecks in the PETSc solvers. Disabling vendor-tuned collective implementations improved solver performance by $8\times$. Additionally, we developed a novel, load balancing technique to address load imbalance in XGC1's new nonlinear collisional operator. Overall, we were able to improve XGC1 performance by $2\times$ on 256K cores of Titan.

Within the ParaDIS NNSA SAP, we exploited efficient OpenMP configurations (MPI vs. OpenMP balance, placement, compiler flags, etc.) to attain a $2\times$ speedup at 512K cores on Sequoia. Additionally, we collaborated with FastMath to integrate and analyze the use of their new KINSOL solver in ParaDIS. Overall, KINSOL improved performance by about 10% at 256K cores. Finally, we developed a new domain decomposition and load balancing algorithm capable of detecting and correcting load imbalance at scale. This improved performance by nearly 20%.

MFDn leverages a stored-matrix Lanczos eigensolver to calculate the properties of light atomic nuclei. To save memory, symmetry and reduced precision are leveraged. As such, the SAP was in need of high-performance implementations of SpMV and SpMV_T. Moreover, as SpMV is inherently memory bandwidth limited and Lanczos requires global dot products, SUPER collaborated with FastMath on the optimization and integration of a block (s-step) eigensolver (LOBPCG) that transforms multiple SpMV's into a single SpMM and transforms multiple global dot products into a single matmul. The resultant $4\times$ local SpMM speedup enabled an overall speedup of $1.8\times$ at scale [3, 5].

Unlike MFDn, BIGSTICK (CalLat NP SAP) is a matrix-free (factored matrix) implementation of nuclear CI. Unfortunately, at high concurrency, load balance becomes increasingly challenging. In a two-step method, we first benchmark MFDn to create a cost model, then use these empirically-derived weights to affect better load balance. The result was a $4\times$ speedup on Mira at 64K cores [58]. Similar load balancing emerged on manycore processors like

*PI Contact: Lenny Oliker, loliker@lbl.gov, Lawrence Berkeley National Laboratory, 510-990-5328

KNL at high thread concurrency (128-256 threads). To that end, we implemented a fundamentally different approach to threading and attained a 7× speedup on KNL [60].

Although hybrid programming models (MPI+X) are obviously necessitated by GPU-acceleration, they can also be necessitated on manycore architectures. When attempting to run CCSD(T) from NWChem on the Knights Corner (KNC) manycore accelerator (predecessor to Cori's KNL), we found that there was only sufficient memory to run one MPI process (one core) per card. To remedy this extreme underutilization, we developed an OpenMP implementation that produced a 65× increase in KNC performance [59].

Prior to SciDAC3, the LibTensor framework (used in QChem within the Excited States BES SAP), had been a single-node, out-of-core tensor contraction library. Such an implementation was not appropriate for large-scale DOE supercomputers. To that end, we collaborated with UCB on the integration of the Cyclops Tensor Framework (CTF) [64] as a means of providing a distributed memory back end. The result allowed for a 150× speedup, as well as portability to BGQ-, KNL-, and GPU-based supercomputers [36].

Perhaps the most impressive speedup was attained on the FES code Xolotl (PSI FES SAP) where accelerated comparisons and pruning redundant work resulted in a more than 10,000× speedup on the reaction network initialization routine (falling from nearly a day to 6 seconds) and overall speedup of over 1,000× on the Cray XC30 Eos.

3 Autotuning

While SciDAC2's PERI pioneered the development of autotuning tools, SUPER advanced this work in several ways, increasing the capability of tools and their ability to optimize SciDAC applications. Early in the project, we integrated various autotuning, performance analysis and compiler, and code generation tools, including Active Harmony, TAU, PAPI, Orio, CHiLL and CUDA-CHiLL, and ROSE. We completed an an automated, end-to-end optimization of the SMG2000 benchmark, a semi-coarsening multigrid on structured grids [69]. This demonstration combined *outlining* using the ROSE compiler, transformation and code generation using CHiLL, and search space navigation with Active Harmony. A subsequent experiment combined TAU with CHiLL, Active Harmony, and Orio to demonstrate end-to-end autotuning using performance data (via PAPI), a performance database TAUdb, and machine learning [18]. An integration with TAU and Orio's OpenCL code generation investigated autotuning to facilitate porting to different GPUs [19].

Throughout the project, we extended these tool capabilities, inspired by SciDAC applications requirements. For Active Harmony, a new user interface was developed that simplified specifying the autotuning search space, as well as extensions for multi-objective autotuning (see Section 7). Orio was extended to optimize and tune for modern architecture features: SIMD pragmas [46], and GPU code generation and autotuning of sparse matrix and vector kernels used in the PETSc toolkit [23, 47]. CHiLL has been extended to improve the quality of generated code [21], generate GPU code [38], and most notably, employ loop and data transformations for sparse matrix computations using an inspector/executor approach [73, 74, 75].

With these enhancements, we also applied autotuning to a several SciDAC application codes. As a group, the SUPER autotuning team examined key computations from three SciDAC applications: (1) the PUSHE computation from the EPSi project XGC1 code; (2) a DSlash operator example extracted from the USQCD MILC code containing a conjugate gradient solver and hand-optimized SU(3) primitives; and, (3) the MPAS Ocean model from the Multipscale climate project. The first of these, XGC1, did not yield improvements as the application developers were simultaneously improving the code. For USQCD MILC, the Dslash operator was manually tuned to target multi-core architectures with SSE/AVX SIMD units. It was heavily unrolled and restructured. As an exercise in demonstrating programmer productivity of compiler-directed autotuning, we started with a high-level version of this code and used CHiLL's transformations to generated a version very similar to the manually-tuned code. The MPAS 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. We developed a custom transformation tool that replaces this indirection inside loop nests with a pointer buffer that points directly to the array being accessed, we demonstrating a 1.4× speedup on *velocity corrector step* function of MPAS, and a 1.08× overall speedup on the application. The application developers modified their data structures as a result of this finding.

In a collaboration with FASTMATH, sparse matrix support in CHiLL was used to optimize the Locally Optimal Block Preconditioned Conjugate Gradient (LOBPCG) solver [2], and was shown to outperform manually-tuned code for the same algorithm [4]. Using an inspector-executor approach, a data transformation converted the very large, symmetric sparse matrix from a CSR format to a *compressed sparse block (CSB)* format. To reduce the data movement associated with indices of the matrices, a short integer was used as the type for the matrices that pointed to the

beginning of each CSB block. Targets for AVX SIMD code generation were marked with pragmas for the native Intel ICC compiler.

4 Performance Tools

Tuning and Analysis Utilities (TAU)

SUPER made significant extensions and improvements to the TAU Performance System [63] that enabled TAU to be applied effectively in performance studies on several institute collaborations. A TAUdb repository was established to store performance data and share it among the SUPER team, and TAU analysis software was configured to provide access to the repository. The source code auto-instrumentation support in TAU was enhanced throughout the project, as support for updated parsers from EDG (C++) was updated and a new Rose [41] (Fortran) parser was integrated. Binary instrumentation support was also continuously enhanced, as new versions of DyninstAPI [72] were supported and new support using the PEBIL binary rewriter [40] was integrated into TAU. Event-based sampling and hybrid timing/sampling measurement in TAU was greatly enhanced throughout the project, providing measurement options for complex applications that are a challenge to auto-instrument [44]. OpenMP support in TAU was also advanced during the project – including a GOMP library wrapper for GCC 4.9 – eventually leading to support for the OMPT (OpenMP Tools) interface in the proposed OpenMP 5.0 standard [26]. TAU has support for an early prototype implementation of OMPT [32], and is being updated to support the eventual "reference" implementation as provided by the OMPT draft subcommittee. TAU was improved and extended for GPUs and manycore devices throughout the SUPER project. CUDA was fully-integrated in TAU using NVIDIA's CUPTI interface [45]. TAU was ported to early and current generations of the Intel Xeon Phi (MIC) architecture. Our efforts reduced TAU measurement overhead for high thread counts on manycore systems. TAU measurement and analysis, together with TAUdb, were incorporated in both CHiLL/Active Harmony and Orio autotuning frameworks. We demonstrated how autotuning could be use to optimize code for multiple architecture targets [20]. Finally, performance data was mapped to the application domain, as TAU data was integrated into MPAS-Ocean output data to be visualized in VisIT. [33] All of the aforementioned TAU enhancements were used to study SciDAC applications, including CCSM, XGC, MILC and MPAS-Ocean.

Roofline Toolkit

Perhaps one of the more user-intensive and challenging activities is understanding the capabilities of the myriad of emerging and novel architectures. One might rely on vendor marketing to quantify the performance potential of a new processor, but this method often overestimates real bandwidths or compute capabilities (STREAM bandwidth is often lower than pin bandwidth, KNL down clocks to 1.2GHz for AVX-heavy computations, etc.). Alternately, one might try to digest the dry and voluminous architectural documentation provided by each vendor. This is a time consuming operation and can fail similarly to the aforementioned approach. A third approach is to construct benchmarks to characterize machines. To that end, as part of the Empirical Roofline Toolkit, we developed a series of benchmarks designed to produce Roofline models for multicore, manycore, and GPU-accelerated node architectures. These benchmarks leverage MPI+OpenMP+CUDA and can thus characterize any of DOE's supercomputers including the BGQ-based Mira, the KNL-based Cori, or the multi-GPU accelerated SummitDev [42]. We also developed a benchmark to characterize NVIDIA's new Unified Memory technology that hides the hierarchical memory nature of GPU-accelerated systems from users [43]. In doing so we were able to identify certain performance pitfalls and mediation strategies for Titan when using CUDA Unified Memory.

We have developed an application to support the visualization of roofline data beyond generation of static images [52]. The roofline visualizer is implemented in Java, using the JavaFX toolkit. It reads and renders Roofline summary data from Json files generated by the Empirical Roofline Toolkit. It also allows browsing and display of rooflines from online roofline data repositories. Such repositories facilitate organized storage and sharing of roofline data. Since multiple rooflines can be loaded simultaneously the visualizer allows filtration of rooflines according to the metadata attributes included in the Json output. Display of multiple rooflines at once facilitates comparative analysis of system performance. Through integration with the TAUdb profile database the Roofline Visualizer is able to display selected application performance profile data on a corresponding Roofline figure. This enables application developers to identify compute kernels with potential for significant performance improvement. The Roofline Visualizer is optionally available as a plugin to the Eclipse IDE. In this mode it allows selection of application source code elements from the IDE's user interface. If performance data for the selected code is available it will be rendered into the selected roofline chart.

5 Energy Efficiency

Future HPC systems will most likely be over-provisioned and power constrained to be operated within strict power limits [53, 56], i.e., not all components of the system can be run at maximum power draw and, therefore, maximum performance. For these classes of systems, the objective of performance optimization, which has traditionally been to minimize time-to-solution, will morph into a more complicated objective maximizing performance under a power budget. Therefore, SUPERs energy thrust was charged with 1) understanding how computation and communication patterns affect the overall power requirements of HPC applications, 2) leveraging this understanding to design application- and architecture-aware optimization techniques that enable the reduction of DOE HPC energy footprint or efficient computing within a power bound and 3) working with the SUPER team towards the overall goal of the multi-objective optimization combining power, performance, and resilience.

Initial tool development work to measure fine-grain power usage was done to support the research. Development on PAPI to enable on-board Xeon Phi power measurement and use of the RAPI for Xeons was performed. In addition, the Resource Centric Reflection (RCR) tool suite was designed for monitoring off-core hardware performance counters and other node-wide metrics and for computing real time models based on those metrics. RCR Tool was also used for monitoring temperature and power consumption to guide real time decisions regarding clock modulation and power states, as well as fine grain thread/task scheduling.

The Green Queue framework [54, 70] is a framework designed to automate the development and deployment of customized architecture- and application-aware power savings recipes for large-scale HPC applications achieving up to 21% and 32% energy savings on HPC production applications run at scale. Additional work focused on the memory sub-system involved our methodology that uses application and machine characterization information to build predictive machine learning models that can accurately quantify phase-level sensitivity to the reduced per core memory bandwidth resulting from changes in the memory bus frequency to reduce the power. We evaluated the predictive capability of the model on real applications and validated them at a fine grain level by looking at 43 individual computational phases or application hotspots as well as the whole application. For more than 91% of the application hotspots, the prediction error is less than 10% [71]. Building from these validated performance and power models collaborations among SUPER team members developed an automated end-to-end system to reduce the complexity of developing and deploying machine learning models for performance, power, and energy. The new framework Automatic Multi-objective Modeling with Machine Learning (AutoMOMML) enabled multi-objective optimizations (power and performance) for HPC workloads [13].

6 Resilience

During the years of SUPER, we have made major advances for resilience. Our approaches covered two major thrusts: detection and correction of errors by formal methods; and improving the performance of application failure recovery mechanisms. Along the first thrust, SUPER investigators focused on understanding how hardware-level errors can be detected and corrected in software. Our efforts exposed intrinsic properties of algorithms for validation and correction and developed the theoretical foundations allowing other researchers to identify the sections of their algorithms that present such properties to develop mathematical invariants. As an example, we augmented dataflow task-based programs with specialized validators or invariant checkers and found these approaches not only can detect soft errors and, in some cases, recover the lost data, but also guarantee a correct result in all instances. Additionally, we developed FUSED [61], a tool-flow based on ROSE that inserts control-flow based error detectors into applications and developed a tool-flow called SORREL [1] to support our method to approximate a given stencil computation using linear regression models [62] to serve as a proxy for detecting errors.

In the second thrust, SUPER researchers also made significant strides to improve recovery mechanisms for applications. We used multi-threading techniques to detect and possibly to correct errors in computing logic [34, 35]. We explored the error sensitivity of algebraic multigrid and found that low-cost pointer triplication could detect and eliminate almost all injected errors [17]. We tested the efficacy of roll-forward and roll-back recovery for two Conjugate Gradient (CG) programs and found an average of 70% of all executions converge correctly for a rate that injects an error every 5 minutes. We also made major strides in improving the User Level Fault Mitigation (ULFM), a proposed extension for fault tolerance to the MPI standard. We investigated different fault management models built on ULFM, including containment domains, Algorithm-Based Fault Tolerance (ABFT), and integration into large scientific applications via Fenix [27, 31] and LFLR [68]. We also improved the state-of-the-art in checkpoint/restart by evaluating the power usage of applications during checkpointing to shift power during low I/O power phases to other applications in computation phases [57], yielding up to 32% improvement for a single, high-priority application.

7 Multi-Objective Optimization

The multi-objective optimization effort in SUPER focused on the optimization of multiple metrics for performance, energy, and resilience by treating collections of these metrics as objectives to be simultaneously optimized and/or constraints to be satisfied. Foundational to this work was development of search formulations and algorithms that exploit mathematical optimization capabilities to address the search problem in autotuning [7, 8, 10, 30]. The collaboration also enhanced the capabilities of search frameworks such as Active Harmony [67] by developing a new search strategy, ANGEL [22], for multi-objective search by focusing on a constrained region defined by available metrics. A particular focus of the collaboration was on the simultaneous optimization of metrics relating to run time, power, and energy [9, 11, 13]. We characterized situations where tradeoffs among these metrics occur and demonstrated that nontrivial tradeoffs can exist between energy consumption and established run time. Additionally, we considered the effects of addressing resiliency and increased fault rates when performing multi-objective optimization. Several efforts addressed checkpointing in particular, establishing optimal frequencies of different types of checkpoints [6, 29, 66]. We also examined the effect of the idle power and/or filesystem energy consumption when considering multi-objective optimization of multi-level checkpointing schemes [12]. We applied our search and modeling methodology to several emerging architectures, including GPUs [51] and FPGAs [48, 49]. We also examined the use of this modeling to improve performance portability across different leadership-class architectures [55]. Under SUPER, we also examined I/O performance and its modeling and optimization to address situations where an application is I/O intensive [15, 16, 37]. This work included inter-institute collaborations between SUPER and SDAV institute.

8 Engagement

Broadly speaking, SUPER's engagement activities can be categorized into application partnerships, inter-institute, facilities, and vendors. SUPER's primary performance engineering activities can be categorized as collaborations between researchers from SUPER and application partnership as described in Section 2.

Most of SUPER's inter-institutional collaborations were with FastMath researchers. We collaborated with FastMath on the optimization and integration of LOBPCG (a replacement for Lanczos as an eigensolver) into the Nuclear Configuration interaction codes MFDn (NUCLEI SAP) and BIGSTICK (CalLat SAP). Whereas Lanczos is based on matvec (SpMV in MFDn), LOBPCG is based on matrix-block vector products (SpMM in MFDn). This activity produced several joint publications and code releases [3, 5].

Additionally, we explored optimization of linear solvers including direct solvers based on HSS factorizations [28], coarse grid solves for 2-level algebraic multigrid [25], and geometric multigrid solvers using HPGMG-FV. HPGMG-FV is a highly-optimized full multigrid solver developed as an alternative to HPL and HPCG for benchmarking supercomputers. The BoxLib-based cosmology code Nyx requires a highly scalable gravity (Poisson) solve. Unfortunately, the existing BoxLib-based solution lacked scalability above 10K cores. To rectify this, SUPER researchers worked with FASTMath researchers to integrate HPGMG into the production BoxLib so that Nyx could leverage its performance capabilities. Researchers from Nyx have since demonstrated scalability to nearly 0.5M cores.

In order to facilitate the performance analysis requirements for FastMath researchers, we collaborated with FastMath on the development of aforementioned Empirical Roofline Toolkit and Visualizer [42] as described in Section 4, as well as conducting an investigation of performance counter infrastructure on the Cray XC series and standalone x86 systems. Eventually this activity expanded to include researchers from NERSC's NESAP application readiness activity who leveraged Roofline across their applications to drive optimization and analyze performance on KNL [14, 24, 50]. This strong collaboration with NERSC also included the development of a performance counter methodology on the Cray XC series (using Intel's VTune and SDE).

Buoyed by success of the NESAP activities, SUPER and NERSC began a collaboration with the Intel Advisor team with the goal of integrating Roofline visualization directly into advisor. Newer releases of Intel advisor now include a Roofline visualization of every large loop nest in an application. These can be analyzed in terms of performance, data movement, SIMDization, and overall fraction of application run time. In March of 2017 we conducted a successful Intel Advisor Roofline Hackathon at LBL [39] with participants from LBNL, NERSC, and LANL, and we are exploring a tutorial at Supercomputing 2017. Commensurate with our NESAP collaborations, our team also collaborated with OLCF's CAAR team on the optimization of XGC1 resulting in an 18% improvement in point-to-point communication performance.

References

[1] SORREL: A Soft-Error Detection Framework. http://www.cs.utah.edu/formal_verification/fmr/#sorrel.

[2] K. Ahmad, A. Venkat, and M. Hall. Optimizing LOBPCG: Sparse matrix loop and data transformations in action. In *Languages and Cmopilers for Parallel Computing: 29th International Workshop*, Editors: Chen Ding, John Criswell and Peng Wu, pages 218–231. Springer-Verlag, 2016.

[3] H. M. Aktulga, A. Buluc, S. Williams, and C. Yang. Optimizing sparse matrix-multiple vector multiplication for nuclear configuration interaction calculations. In *International Parallel and Distributed Processing Symposium (IPDPS)*, 2014.

[4] H. M. Aktulga, A. Buluc, S. Williams, and C. h. Yang. Optimizing sparse matrix-multiple vectors multiplication for nuclear co nfiguration interaction calculations. In *Parallel and Distributed Processing Symposium, 2014 IEEE 28th Inter national*, pages 1213–1222. IEEE, 2014.

[5] H. M. Aktulga, M. Afibuzzaman, S. Williams, A. Buluc, M. Shao, C. Yang, E. G. Ng, P. Maris, and J. P. Vary. A high performance block eigensolver for nuclear configuration interaction calculations. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 2016.

[6] G. Aupy, A. Benoit, T. Héfault, Y. Robert, and J. Dongarra. Optimal checkpointing period: Time vs. energy. In *Proceedings of the 4th International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS13)*, November 2013.

[7] P. Balaprakash, S. M. Wild, and P. D. Hovland. An experimental study of global and local search algorithms in empirical performance tuning. In *Proceedings of the 10th International Meeting on High-Performance Computing for Computational Science (VECPAR 2012)*, July 2012. Available at http://www.mcs.anl.gov/~wild/papers/2012/PBSWPH12.pdf.

[8] P. Balaprakash, R. Gramacy, and S. M. Wild. Active-learning-based surrogate models for empirical performance tuning. In *Proceedings of IEEE International Conference on Cluster Computing (CLUSTER 2013)*, pages 1–8, September 2013. doi: 10.1109/CLUSTER.2013.6702683.

[9] P. Balaprakash, A. Tiwari, and S. M. Wild. Framework for optimizing power, energy, and performance (poster). In *International Conference for High Performance Computing, Networking, Storage, and Analysis (SC13)*, November 2013. URL http://sc13.supercomputing.org/sites/default/files/PostersArchive/post154.html.

[10] 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.

[11] P. Balaprakash, A. Tiwari, and S. M. Wild. Multi-objective optimization of HPC kernels for performance, power, and energy. In S. A. Jarvis, S. A. Wright, and S. D. Hammond, editors, *High Performance Computing Systems. Performance Modeling, Benchmarking and Simulation*, volume 8551 of *Lecture Notes in Computer Science*, pages 239–260. Springer International Publishing, 2014. doi: 10.1007/978-3-319-10214-6_12.

[12] P. Balaprakash, L. A. B. Gomez, M. S. Bouguerra, S. M. Wild, F. Cappello, and P. D. Hovland. Analysis of the tradeoffs between energy and run time for multilevel checkpointing. In S. A. Jarvis, S. A. Wright, and S. D. Hammond, editors, *High Performance Computing Systems. Performance Modeling, Benchmarking, and Simulation – PMBS 2014*, volume 8966 of *Lecture Notes in Computer Science*, pages 249–263. Springer International Publishing, 2015. doi: 10.1007/978-3-319-17248-4_13. Acceptance rate 26%.

[13] P. Balaprakash, A. Tiwari, S. M. Wild, L. Carrington, and P. D. Hovland. AutoMOMML: Automatic Multi-objective Modeling with Machine Learning. In M. J. Kunkel, P. Balaji, and J. Dongarra, editors, *High Performance Computing: 31st International Conference, ISC High Performance 2016, Frankfurt, Germany, June 19-23, 2016, Proceedings*, pages 219–239. Springer International Publishing, 2016. ISBN 978-3-319-41321-1. doi: 10.1007/978-3-319-41321-1_12.

[14] T. Barnes, B. Cook, J. Deslippe, D. Doerfler, B. Friesen, Y. H. He, T. Kurth, T. Koskela, M. Lobet, T. Malas, L. Oliker, A. Ovsyannikov, A. Sarje, J.-L. Vay, H. Vincenti, S. Williams, P. Carrier, N. Wichmann, M. Wagner, P. Kent, C. Kerr, and J. Dennis. Evaluating and optimizing the nersc workload on knights landing. In *Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*, 2016.

[15] B. Behzad, S. Byna, S. M. Wild, Prabhat, and M. Snir. Improving parallel I/O autotuning with performance modeling. In *Proceedings of the 23rd International Symposium on High-Performance Parallel and Distributed Computing (HPDC14)*, 2014. doi: 10.1145/2600212.2600708.

[16] B. Behzad, S. Byna, S. M. Wild, Prabhat, and M. Snir. Dynamic model-driven parallel I/O performance tuning. In *Cluster Computing (CLUSTER), 2015 IEEE International Conference on*, pages 184–193, September 2015. doi: 10.1109/CLUSTER.2015.37.

[17] M. Casas, B. R. de Supinski, G. Bronevetsky, and M. Schulz. Fault resilience of the algebraic multi-grid solver. In *Proceedings of the 26th ACM International Conference on Supercomputing (IcS)*, pages 91–100. ACM, 2012.

[18] N. Chaimov, S. Biersdorff, and A. D. Malony. Tools for machine-learning-based empirical autotuning and specialization. *International Journal of High Performance Computing Applications*, 2013. doi: 10.1177/1094342013493124. URL http://hpc.sagepub.com/content/early/2013/07/12/1094342013493124.abstract.

[19] N. Chaimov, B. Norris, and A. Malony. Multi-target autotuning for accelerators. In *Proceedings of the 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, 2014.

[20] N. Chaimov, B. Norris, and A. Malony. Toward Multi-Target Autotuning for Accelerators. In *20th IEEE International Conference on Parallel and Distributed Systems (ICPADS 2014)*, pages 534–541, Dec. 2014.

[21] C. Chen. Polyhedra scanning revisited. In *Proceedings of the 33rd ACM SIGPLAN conference on Programming Language Design and Implementation*, PLDI '12, pages 499–508, June 2012. ISBN 978-1-4503-1205-9. doi: 10.1145/2254064.2254123. URL http://doi.acm.org/10.1145/2254064.2254123.

[22] R. S. Chen and J. K. Hollingsworth. ANGEL: A Hierarchical Approach to Multi-Objective Online Auto-Tuning. In *Proceedings of the 5th International Workshop on Runtime and Operating Systems for Supercomputers*, ROSS '15, pages 4:1–4:8, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3606-2. doi: 10.1145/2768405.2768409. URL http://doi.acm.org/10.1145/2768405.2768409.

[23] C. Choudary, J. Godwin, J. Holewinski, D. Karthik, D. Lowell, A. Mametjanov, 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.

[24] D. Doerfer, J. Deslippe, S. Williams, L. Oliker, B. Cook, T. Kurth, M. Lobet, T. Malas, J.-L. Vay, , and H. Vincenti. Applying the roofline performance model to the intel xeon phi knights landing processor. In *Intel Xeon Phi User Group Workshop (IXPUG)*, 2016.

[25] A. Druinsky, P. Ghysels, X. S. Li, O. Marques, S. Williams, A. Barker, D. Kalchev, and P. Vassilevski. Comparative performance analysis of an algebraic multigrid solver on leading multicore architectures. In *International Conference on Parallel Processing and Applied Mathematics (PPAM)*, 2015.

[26] A. Eichenberger, J. Mellor-Crummey, M. Schulz, N. Copty, 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.

[27] M. Gamell, D. S. Katz, H. Kolla, J. Chen, S. Klasky, and M. Parashar. Exploring automatic, online failure recovery for scientific applications at extreme scales. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '14, pages 895–906, Piscataway, NJ, USA, 2014. IEEE Press. ISBN 978-1-4799-5500-8. doi: 10.1109/SC.2014.78. URL http://dx.doi.org/10.1109/SC.2014.78.

[28] P. Ghysels, X. S. Li, F.-H. Rouet, S. Williams, and A. Napov. An efficient multicore implementation of a novel hss-structured multifrontal solver using randomized sampling. *SIAM Journal of Scientific Computing (SISC)*, 2016.

[29] L. A. B. Gomez, P. Balaprakash, M.-S. Bouguerra, S. M. Wild, F. Cappello, and P. D. Hovland. Energy-performance tradeoffs in multilevel checkpoint strategies. In *2014 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 278–279, 2014. doi: 10.1109/CLUSTER.2014.6968749. Poster Extended Abstract.

[30] R. B. Gramacy, M. A. Taddy, and S. M. Wild. Variable selection and sensitivity analysis via dynamic trees with an application to computer code performance tuning. *Annals of Applied Statistics*, 7:51–80, 2013. doi: 10.1214/12-AOAS590.

[31] T. Herault, A. Bouteiller, G. Bosilca, M. Gamell, K. Teranishi, M. Parashar, and J. Dongarra. Practical Scalable Consensus for Pseudo-Synchronous Distributed Systems. November 2015.

[32] K. Huck, A. Malony, S. Shende, and D. Jacobsen. Integrated measurement for cross-platform OpenMP performance analysis. In L. DeRose, B. R. de Supinski, S. L. Olivier, B. M. Chapman, and M. S. Müller, editors, *Using and Improving OpenMP for Devices, Tasks, and More*, volume 8766 of *Lecture Notes in Computer Science*, pages 146–160. Springer International Publishing, 2014. ISBN 978-3-319-11453-8. doi: 10.1007/978-3-319-11454-5_11. URL http://dx.doi.org/10.1007/978-3-319-11454-5_11.

[33] K. A. Huck, K. Potter, D. W. Jacobsen, H. Childs, and A. D. Malony. Linking performance data into scientific visualization tools. In *Proceedings of the First Workshop on Visual Performance Analysis*, pages 50–57. IEEE Press, 2014.

[34] S. Hukerikar, K. Teranishi, P. Diniz, and R. Lucas. Opportunistic Application-level Fault Detection through Adaptive Redundant Multithreading. In *Proc. of the IEEE Intl. Conf. on High Performance Computing and Simulation (HPCS 2014)*, July 2014.

[35] S. Hukerikar, P. Diniz, and R. Lucas. Enabling Application Resilience through Programming Model based Fault Amelioration. Sept. 2015.

[36] K. Z. Ibrahim, E. Epifanovsky, S. Williams, and A. I. Krylov. Cross-scale efficient tensor contractions for coupled cluster computations through multiple programming model backends. *Journal of Parallel and Distributed Computing (JPDC)*, 2017.

[37] F. Isaila, P. Balaprakash, S. M. Wild, D. Kimpe, R. Latham, R. Ross, and P. D. Hovland. Collective I/O tuning using analytical and machine learning models. In *2015 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 128–137. IEEE, 2015. doi: 10.1109/CLUSTER.2015.29. Acceptance rate 24%.

[38] M. Khan, P. Basu, G. Rudy, M. Hall, C. Chen, and J. Chame. A script-based autotuning compiler system to generate high-performance cuda code. *ACM Transactions on Architecture and Code Optimization*, Jan. 2013.

[39] T. Koskela. Roofline hackathon. URL http://www.nersc.gov/users/training/events/roofline-training-2162017-2172017/.

[40] M. Laurenzano, M. Tikir, L. Carrington, and A. Snavely. 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.

[41] Lawrence Livermore National Laboratory. ROSE compiler infrastructure. http://rosecompiler.org, April 2013.

[42] T. Ligocki. Roofline toolkit. URL https://bitbucket.org/berkeleylab/cs-roofline-toolkit.

[43] Y. J. Lo, S. Williams, B. V. Straalen, T. J. Ligocki, M. J. Cordery, L. Oliker, and M. W. Hall. Roofline model toolkit: A practical tool for architectural and program analysis. In *Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*, 2014.

[44] A. Malony and K. Huck. General hybrid parallel profiling. In *Parallel, Distributed and Network-Based Processing (PDP), 2014 22nd Euromicro International Conference on*, pages 204–212, Feb 2014. doi: 10.1109/PDP.2014.38.

[45] A. Malony, S. Biersdorff, S. Shende, H. Jagode, S. Tomov, G. Juckeland, R. Dietrich, D. Poole, and C. Lamb. Parallel Performance Measurement of Heterogeneous Parallel Systems with GPUs. In *International Conference on Parallel Processing (ICPP 2011)*, pages 176–185. IEEE Computer Society, Sept. 2011. doi: 10.1109/ICPP.2011.71.

[46] A. Mametjanov and B. Norris. Autotuning of Vectorization in Stencil Computations. Workshop on Optimizing Stencil Computations (WOSC'13), Indianapolis, Indiana, October 2013.

[47] 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.

[48] A. Mametjanov, P. Balaprakash, C. Choudary, P. D. Hovland, S. M. Wild, and G. Sabin. Autotuning FPGA design parameters for performance and power. In *2015 IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 84–91, 2015. doi: 10.1109/FCCM.2015.54. Acceptance rate 22.10%.

[49] A. Mametjanov, P. Balaprakash, C. Choudary, P. D. Hovland, S. M. Wild, G. Sabin, and G. Wolfe. Improving FPGA design parameter exploration: Timing, power, and area. Preprint ANL/MCS-P7000-0117, Argonne National Laboratory, Mathematics and Computer Science Division, Jan 2017.

[50] Z. Meng, A. Koniges, Y. H. He, S. Williams, T. Kurth, B. Cook, J. Deslippe, and A. L. Bertozzi. Openmp parallelization and optimization of graph-based machine learning algorithms. In *12th International Workshop on OpenMP (iWOMP)*, 2016.

[51] T. Nelson, A. Rivera, P. Balaprakash, M. Hall, P. D. Hovland, E. Jessup, and B. Norris. Generating efficient tensor contractions for GPUs. In *2015 44th International Conference on Parallel Processing (ICPP)*, pages 969–978, 2015. doi: 10.1109/ICPP.2015.106. Acceptance rate 32.5%.

[52] B. Norris, W. Spear, and A. Malony. Performance analysis of applications in the context of architectural rooflines. In *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering*, ICPE '17, pages 345–348, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4404-3. doi: 10.1145/3030207.3030232. URL http://doi.acm.org/10.1145/3030207.3030232.

[53] T. Patki, D. K. Lowenthal, B. Rountree, M. Schulz, and B. R. de Supinski. Exploring hardware overprovisioning in power-constrained, high performance computing. In *Proceedings of the 27th International ACM Conference on International Conference on Supercomputing*, ICS '13, pages 173–182, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2130-3. doi: 10.1145/2464996.2465009. URL http://doi.acm.org/10.1145/2464996.2465009.

[54] 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*.

[55] A. Roy, P. Balaprakash, P. D. Hovland, and S. M. Wild. Exploiting performance portability in search algorithms for autotuning. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 1535–1544, 2016. doi: 10.1109/IPDPSW.2016.85.

[56] O. Sarood, A. Langer, L. Kale, B. Rountree, and B. de Supinski. Optimizing power allocation to cpu and memory subsystems in overprovisioned hpc systems. In *Cluster Computing (CLUSTER), 2013 IEEE International Conference on*, pages 1–8, Sept 2013. doi: 10.1109/CLUSTER.2013.6702684.

[57] L. Savoie, D. K. Lowenthal, B. R. de Supinski, T. Islam, K. Mohror, B. Rountree, and M. Schulz. I/O Aware Power Shifting. In *Parallel and Distributed Processing Symposium, 2016 IEEE International*, pages 740–749. IEEE, 2016.

[58] H. Shan, K. McElvain, C. W. Johnson, S. Williams, and W. E. Ormand. Parallel implementation and performance optimization of the configuration-interaction method. In *Supercomputing (SC)*, 2015.

[59] H. Shan, S. Williams, W. de Jong, and L. Oliker. Thread-level parallelization and optimization of nwchem for the intel mic architecture. In *Proceedings of the Sixth International Workshop on Programming Models and Applications for Multicores and Manycores*, PMAM '15, pages 58–67, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3404-4. doi: 10.1145/2712386.2712391. URL http://doi.acm.org/10.1145/2712386.2712391.

[60] H. Shan, S. Williams, C. Johnson, and K. McElvain. A locality-based threading algorithm for the configuration-interaction method. In *Parallel and Distributed Scientific and Engineering Computing (PDSEC)*, June 2017.

[61] V. C. Sharma, Z. Rakamaric, and G. Gopalakrishnan. FUSED: A Low-cost online Soft-Error Detector. In *10th Workshop on Silicon Errors in Logic - System Effects (SELSE)*, 2014.

[62] V. C. Sharma, G. Bronevetsky, and G. Gopalakrishnan. Detecting soft errors in stencil based computations. In *11th IEEE Workshop on Silicon Errors in Logic - System Effects (SELSE)*, 2015.

[63] 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.

[64] E. Solomonik, D. Matthews, J. Hammond, and J. Demmel. Cyclops Tensor Framework: Reducing communication and eliminating load imbalance in massively parallel contractions. In *The IEEE 27th International Symposium on Parallel Distributed Processing (IPDPS)*, pages 813–824, 2013. doi: 10.1109/IPDPS.2013.112.

[65] S. Sreepathi, E. DAzevedo, B. Philip, and P. Worley. Communication characterization and optimization of applications using topology-aware task mapping on large supercomputers. In *ACM/SPEC International Conference on Performance Engineering (ICPE)*, 2016.

[66] O. Subasi, S. Di, L. Bautista-Gomez, P. Balaprakash, O. Unsal, J. Labarta, A. Cristal, and F. Cappello. Spatial support vector regression to detect silent errors in the exascale era. In *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 413–424, 2016. doi: 10.1109/CCGrid.2016.33. Acceptance rate 20.0%.

[67] C. Ţăpuş, I.-H. Chung, J. K. Hollingsworth, et al. Active harmony: Towards automated performance tuning. In *Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, pages 1–11. IEEE Computer Society Press, 2002.

[68] K. Teranishi and M. A. Heroux. Toward Local Failure Local Recovery Resilience Model Using MPI-ULFM. In *Proceedings of the 21st European MPI Users' Group Meeting*, EuroMPI/ASIA '14, pages 51:51–51:56, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2875-3. doi: 10.1145/2642769.2642774. URL http://doi.acm.org/10.1145/2642769.2642774.

[69] A. Tiwari, C. Chen, C. Liao, J. Chame, J. Hollingsworth, M. Hall, and D. Quinlan. Auto-tuning full applications: A case study. *International Journal of High Performance Computing Applications*, pages 286–294, Aug. 2011.

[70] A. Tiwari, J. Peraza, M. Laurenzano, L. Carrington, and A. Snavely. Green queue: Customized large-scale clock frequency scaling. In *Proceedings of the Second International Conference on Cloud and Green Computing*, CGC '12, 2012.

[71] A. Tiwari, A. Gamst, M. Laurenzano, M. Schulz, and L. Carrington. Modeling the impact of reduced memory bandwidth on hpc applications. In F. Silva, I. Dutra, and V. Santos Costa, editors, *Euro-Par 2014 Parallel Processing*, volume 8632 of *Lecture Notes in Computer Science*, pages 63–74. Springer International Publishing, 2014. ISBN 978-3-319-09872-2. doi: 10.1007/978-3-319-09873-9_6. URL http://dx.doi.org/10.1007/978-3-319-09873-9_6.

[72] University of Maryland and University of Wisconsin. Paradyn/dyninst - putting the performance in high performance computing. http://www.dyninst.org, April 2013.

[73] A. Venkat, M. Hall, and M. Strout. Loop and data transformations for sparse matrix codes. In *Proceedings of the 36th ACM SIGPLAN conference on Programming Language Design and Implementation*, PLDI '15, June 2015.

[74] A. Venkat, M. S. Mohammadi, J. Park, H. Rong, R. Barik, M. M. Strout, and M. Hall. Automating wavefront parallelization for sparse matrix computations. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '16, pages 41:1–41:12, Piscataway, NJ, USA, 2016. IEEE Press. ISBN 978-1-4673-8815-3. URL http://dl.acm.org/citation.cfm?id=3014904.3014959.

[75] R. Venkatasubramanian, J. P. Hayes, and B. T. Murray. Low-cost on-line fault detection using control flow assertions. In *IEEE On-Line Testing Symposium (IOLTS)*, pages 137–143, 2003.