CTWatch QUARTERLY
November 2007

Creating Software Tools and Libraries for Leadership Computing

John Mellor-Crummey, Rice University
Peter Beckman, Argonne National Laboratory
Keith Cooper, Rice University
Jack Dongarra, University of Tennessee, Knoxville
William Gropp, Argonne National Laboratory
Ewing Lusk, Argonne National Laboratory
Barton Miller, University of Wisconsin, Madison
Katherine Yelick, University of California, Berkeley

1. Center for Scalable Application Development Software

The Department of Energy's (DOE) Office of Science is deploying leadership computing facilities, including a Blue Gene/P system at Argonne National Laboratory and a Cray XT system at Oak Ridge National Laboratory, with the aim of catalyzing scientific discovery. These emerging systems composed of tens of thousands of processor cores are beginning to provide immense computational power for scientific simulation and modeling. However, harnessing the capabilities of such large-scale microprocessor-based, parallel systems is daunting for application developers. A grand challenge for computer science is to develop software technology that simplifies using such systems.

To help address this challenge, in January 2007 the Center for Scalable Application Development Software (CScADS)[1] was established as a partnership between Rice University, Argonne National Laboratory, University of California – Berkeley, University of Tennessee – Knoxville, and University of Wisconsin – Madison. As part of the DOE's Scientific Discovery through Advanced Computing (SciDAC) program, CScADS is pursuing an integrated set of activities that aim to increase the productivity of DOE computational scientists by catalyzing the development of software tools and libraries for leadership computing platforms. These activities include workshops to engage the research community in the challenges of leadership-class computing, research and development of open-source software, and work with computational scientists to help them develop codes for leadership computing platforms.

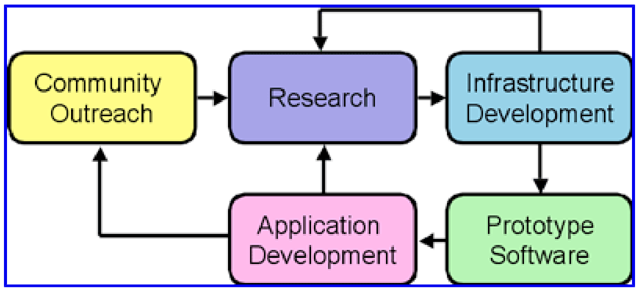

Figure 1. Relationship between CScADS activities.

Figure 1 illustrates the relationships between the Center's activities. The flow of ideas originates from two sources: workshops for community outreach and vision-building, and direct involvement with application development. These activities focus research efforts on important problems. In turn, research drives the infrastructure development by identifying capabilities that are needed to support the long-range vision. Infrastructure feeds back into the research program, but also supports prototyping of software tools that support further application development. Finally, experiences by developers using prototype compilers, tools and libraries will spur the next cycle of research and development.

First, we briefly describe each of the Center's activities in a bit more detail. Then, we describe the themes of CScADS research. Finally, we conclude with a brief discussion of ongoing work.

1.1 Community Outreach and Vision-Building

Achieving petascale performance with applications will require a close collaboration between scientists developing computational models and computer science teams developing enabling technologies. To engage the community in the challenges and to foster interdisciplinary collaborations, we have established the *CScADS Summer Workshops* – an annual series of workshops that will focus on topics related to scalable software for the DOE's leadership-class systems. In July 2007, we held our first series of four workshops in Snowbird, Utah.

- *Automatic Tuning for Petascale Systems*. This workshop brought together researchers to discuss some of the code generation challenges for multicore processors that are the building blocks for emerging petascale systems, to identify some of the opportunities afforded by the use of automatic tuning techniques, and to explore opportunities for collaboration among tuning researchers.
- *Performance Tools for Petascale Computing*. This workshop brought together researchers to discuss the challenges of measurement, attribution, analysis, and presentation of application performance for leadership computing platforms. An important workshop goal was to explore opportunities for research teams to break their software into components to foster community collaboration and accelerate development of effective performance tools for petascale systems.

- *Petascale Architectures and Performance*. This workshop brought together computer scientists and SciDAC application scientists who aim to develop codes for the leadership computing platforms. The goal of this workshop was to introduce the computer scientists to the SciDAC applications and to familiarize the application scientists with the emerging leadership computing platforms, software libraries and tools for parallel computing, and effective strategies for parallel programming.
- *Libraries and Algorithms for Petascale Applications*. This workshop brought together computer scientists working on algorithms and libraries with members of the SciDAC application teams. The principal workshop goal was to identify challenges for library and algorithm developers from the needs of the SciDAC applications and to foster collaboration between the communities. Workshop topics included the use of multicore processors and the use of automatic tuning in libraries.

The latter two workshops included “hands-on” sessions in which computer scientists and application scientists collaboratively explored application challenges on leadership computing platforms.

1.2 Research and Development

Several national reports, such as the 2000 report Information Technology Research: Investing in our Future by the President’s Information Technology Advisory Committee, have pointed out that open-source software represents an opportunity to address the shortage of software support for programming high-end systems. The power of this approach has been amply demonstrated by the success of Linux in fostering the development of operating systems for high-performance clusters.

The CScADS research program focuses on strategies for improving the productivity of application developers for developing high-performance codes for leadership-class machines. Rather than attack a narrow range of problems within this space, CScADS will explore a broad spectrum of issues because we believe that there is a high degree of synergy to be exploited.

Research on software support for high-end systems cannot be performed in a vacuum. Direct interaction between application developers and enabling technologies teams can clarify the problems that need to be addressed, yield insight into strategies for overcoming performance bottlenecks, identify how those strategies might be automated, and produce a vision for new tools and programming systems.

1.3 Open-Source Software Infrastructure

To facilitate the research, both within CScADS and in the community at large, we are developing the *CScADS Open Software Suite*. This suite will include an open-source software infrastructure to support compiler/programming-language research, development, and evaluation based on the Open64 compiler as well as Rice’s D System compiler infrastructure. Other components will include software infrastructure for performance tools, including support for binary analysis, instrumentation, data collection, and measurement interpretation that will draw from Rice’s HPCToolkit[2] and Wisconsin’s Paradyn[3]
and Dyninst tools, and a range of libraries that help harness the power of leadership-class platforms composed of multicore processors.

2. CScADS Research Themes

In CScADS, we have begun a broad program of research on software to support scalability in three dimensions: productivity, homogeneous scalability, and platform heterogeneity. We briefly outline the themes of this work in each of these areas.

2.1 Rapid Construction of High-Performance Applications

An application specification is high level if (1) it is written in a programming system that supports rapid prototyping; (2) aside from algorithm choice, it does not include any hardware-specific programming strategies (e.g., loop tiling); and (3) it is possible to generate code for the entire spectrum of different computing platforms from a single source version. The goal of CScADS productivity research is to explore how we can transform such high-level specifications into high-performance implementations for leadership-class systems.

For higher productivity, we believe that developers should construct high-performance applications by using scripting languages to integrate domain-specific component libraries. At Rice we have been exploring a strategy, called telescoping languages, to generate high-performance compilers for scientific scripting languages. The fundamental idea is to preprocess a library of components to produce a compiler that understands and optimizes component invocations as if they were language primitives. As part of this effort, we have been exploring analysis and optimization based on inference about generalized types. A goal of CScADS research is to explore how we can adapt these ideas to optimize programs based on the Common Component Architecture (CCA).

2.2 Scaling to Homogeneous Parallel Systems

Achieving high performance on a modern microprocessor, though challenging, is not by itself enough for SciDAC applications; in addition, applications must be able to scale to the thousands or even hundreds of thousands of processors that make up a petascale computing platform. Two general classes of software systems are needed to make this feasible: (1) tools that analyze scalable performance and help the developer overcome bottlenecks, and (2) compiler support that can take higher-level languages and map them efficiently to large numbers of processors.

2.2.1 Tools for Scalable Parallel Performance Analysis and Improvement

Effectively harnessing leadership-class systems for capability computing is a grand challenge for computer science. Running codes that are poorly tuned on such systems would waste these precious resources. To help users tune codes for leadership-class systems, we are conducting research on performance tools that addresses the following challenges:

Analyzing integrated measurements. Understanding application performance requires capturing detailed information about parallel application behavior, including the interplay of computation, data movement, synchronization, and I/O. We are focusing on analysis techniques that help understand the interplay of these activities.

Taming the complexity of scale. Analysis and presentation techniques must support top-down analysis to cope with the complexity of large codes running on thousands of processors. To understand executions on thousands of processors, it is not practical to inspect them individually. We are exploring statistical techniques for classifying behaviors into equivalence classes and differential performance analysis techniques for identifying scalability bottlenecks.

Coping with dynamic parallelism. The arrival of multicore processors will give rise to more dynamic threading models on processor nodes. Strategies to analyze the effectiveness of dynamic parallelism will be important in understanding performance on emerging processors.

This work on performance tools extends and complements activities in the Performance Engineering Research Institute (PERI). The CScADS tools research and development will build upon work at Rice on HPCToolkit and work at Wisconsin on Dyninst as well as other tools for analysis and instrumentation of application binaries. An outcome of this effort will be shared interoperable components that will accelerate development of better tools for analyzing the performance of applications running on leadership class systems.

2.2.2 Compiler Technology for Parallel Languages

The principal stumbling block to using parallel computers productively is that parallel programming models in wide use today place most of the burden of managing parallelism and optimizing parallel performance on application developers. We face a looming productivity crisis if we continue programming parallel systems at such a low level of abstraction, as these parallel systems increase in scale and architectural complexity. As a component of CScADS research, we are exploring a range of compiler technologies for parallel systems ranging from technologies with near-term impact to technologies for higher level programming models that we expect to pay off further in the future. This work is being done in conjunction with the DOE-funded Center for Programming Models for Scalable Parallel Computing. Technologies that we are exploring include:

Partitioned global address space (PGAS) languages. Communication optimization will be critical to the performance of PGAS languages on large-scale systems. As part of CScADS, we are enhancing the Open64 compiler infrastructure to support compile-time communication analysis and optimization of Co-Array Fortran and UPC.

Global array languages. High-level languages that support data-parallel programming using a global view offer a dramatically simpler alternative for programming parallel systems. Programming in such languages is simpler; one simply reads and writes shared variables without worrying about synchronization and data movement. An application programmer merely specifies how to partition the data and leaves the details of partitioning the computation and choreographing communication to a parallelizing compiler. Having an HPF program achieve over 10 TFLOPS on Japan's Earth Simulator has rekindled interest in high-level programming models within the US. Research challenges include improving the expressiveness, performance, and portability of high-level programming models.

Parallel scripting languages. Matlab and other scripting languages boost developer productivity both by providing a rich set of library primitives and by abstracting away mundane details of programming. Ongoing work at Rice is exploring compiler technology for Matlab. Work at Tennessee involves parallel implementations of scripting languages such as Matlab, Python, and Mathematica. As a part of this project, we are exploring compiler and run-time techniques that will enable such high-level programming systems to scale to much larger computation configurations while retaining support for most languages features.

2.2.3 Support for Multicore Platforms

Multicore chips will force at least two dimensions of parallelism into scalable architectures: (1) on-chip, shared-memory parallelism and (2) cross-chip distributed-memory parallelism. Many architects predict that with processor speed improvements slowing, the number of cores per chip is likely to double every two years. In addition, many of the contemplated architectures will incorporate multi-threading on each of the cores, adding a third dimension of parallelism. Based on this increased complexity, we see three principal challenges in dealing with scalable parallel systems constructed from multicore chips.

- *Decomposing available parallelism and mapping it well to available resources*. For a given loop nest, we will need to find instruction-level parallelism to exploit short-vector operations, multi-threaded parallelism to map across multiple cores, and outer-loop parallelism to exploit an entire scalable system.
- *Keeping multiple cores busy requires that more data be transferred from off-chip memory*. In the near term, given the limitations on sockets, the aggregate off-chip bandwidth will not scale linearly with the number of cores. For this reason, it will be critical to transform applications to achieve high levels of cache reuse.
- *Choreographing parallelism and data movement*. Rather than having cores compute independently, coordinating their computation with synchronization can improve reuse.

We are pursuing three approaches to cope with the challenges of multicore computing. First, Tennessee is exploring the design of algorithms and component libraries for systems employing multicore chips. This work seeks to achieve the highest possible performance, produce useful libraries, and drive the research on compilation strategies and automatic tuning for multicore chips. Second, Rice is exploring compiler transformations to exploit multicore processors effectively by carefully partitioning and scheduling computations to enhance inter-core data reuse. Third, Argonne is exploring the interaction of multi-threaded application programs with systems software such as node operating systems and communication libraries such as MPI.

2.3. Portability and Support for Heterogeneous Platforms

The third dimension of scalability is mapping an application to different sequential and parallel computing platforms. Over the lifetime of an application, the effort spent in porting and retuning for new platforms can often exceed the original implementation effort. In support of portability, we are initially focusing on obtaining the highest possible performance on leadership-class machines. In addition, we will explore compilation and optimization of applications to permit them to run efficiently on computer systems that incorporate different kinds of computational elements, such as vector/SIMD and scalar processors.

2.3.1 Automatic Tuning to New Platforms

The success of libraries such as ATLAS[4] and FFTW[5]
has increased interest in automatic tuning of components and applications. The goal of research in this area is to develop compiler and run-time technology to identify which loop nests in a program are critical for high performance and then restructure them appropriately to achieve the highest performance on a target platform.

The search space for alternative implementations of loop nests is too big to explore exhaustively. We have been exploring several strategies to reduce the cost of searching for the best loop structure. By leveraging capabilities of Rice's HPCToolkit, we can pinpoint sources of inefficiency at the loop level, which can guide exploration of transformation parameters. Also, we have been employing model guidance along with search to dramatically reduce the size of the search needed for good performance.[6]

As part of CScADS, the Rice and Tennessee groups are continuing their efforts based on LoopTool, HPCToolkit, and ATLAS 2, with a focus on pre-tuning component libraries for various platforms. This work will provide variants of arbitrary component libraries optimized for different platforms and different application contexts, much as Atlas does today. A second group at Rice is extending adaptive code optimization strategies to tune components. This work will explore adaptive transformations and aggressive interprocedural optimization.

2.3.2 Compiling to Heterogeneous Computing Platforms

Emerging high-end computing architectures are beginning to have heterogeneous computational components within a single system. Exploiting these features (or even coping with them) will be a challenge. We believe that new techniques must be incorporated into compilers and tools to support portable high-performance programming. To date, our work has explored compilation for chips with attached vector units (SSE on Intel chips, Altivec on the IBM G5) and code generation for Cell.[7] We are building upon this work to develop compiler techniques for partitioning and mapping computations onto the resources to which they are best suited. These techniques will be critical for effective use of systems that incorporate both vector and scalar elements in the same machine, such as those outlined in Cray's strategy for "adaptive supercomputing."

3. Recent and Ongoing Work

To date, work in CScADS has included both research and development of a range of technologies necessary to support leadership computing, as well as direct involvement with SciDAC application teams. We briefly summarize a few of these efforts.

3.1 Research and Development of Software for Leadership Computing

Rice and Wisconsin have begun collaborative development of a series of performance-tool components that can serve as community infrastructure for performance tools for leadership computing platforms. Initial efforts in this area have been focused on development of multi-platform components for stack unwinding within and across process address spaces that has uses for both debugging and performance analysis, and a library that provides a foundation for sampling-based performance measurement of both statically-linked and dynamically-linked executables.

Berkeley and Tennessee have been collaborating on re-engineering numerical libraries for parallel systems. Initially, this work has been exploring parallel matrix factorization using multi-threading in combination with intelligent scheduling. The new execution model relies on dynamic, dataflow-driven execution and avoids both global synchronization and implicit point-to-point synchronization due to send/receive-style message passing. Experimental results indicate that this strategy can significantly outperform traditional codes by hiding both algorithmic and communication latencies. Future plans call for exploring this programming paradigm for both two-sided linear algebra algorithms and sparse matrix algorithms.

Argonne has been exploring the implementation and performance evaluation of MPI support for multi-threading and remote memory access. Experiments with Argonne's own MPI implementation (MPICH) and various vendor implementations have demonstrated the potential contribution these still little-used parts of MPI can make to parallel program performance and have revealed widely varying attention to efficient implementations.

3.2 Application Engagement

As part of the Center's application engagement efforts, Rice has been working closely with several of the SciDAC S3D and GTC application teams to diagnose application performance bottlenecks on leadership-class platforms using a combination of measurement, analysis, and modeling. S3D is a massively parallel solver for turbulent reacting flows.[8] GTC (Gyrokinetic Toroidal Code) is a three-dimensional particle-in-cell (PIC) code used for studying the impact of fine-scale plasma turbulence on energy and particle confinement in the core of tokamak fusion reactors.[9]

Our early experiences with both S3D and GTC demonstrate the value of the CScADS approach of tightly coupling computer science research with application development and tuning. Work with these applications has influenced development of software tools for performance measurement and performance modeling, as well as motivated a study of run-time libraries for adaptive data reordering.

Work with S3D uncovered opportunities for using source-to-source tools to tailor code to improve memory hierarchy utilization. This led to refinement of Rice's LoopTool program transformation tool. Applying LoopTool to S3D yielded improved performance of S3D's most memory intensive loop by nearly a factor of three.[10]
Additionally, analysis of experiments with S3D on the hybrid Cray XT3/XT4 system showed that the lower memory bandwidth on the XT3 nodes hurts the weak scaling performance of S3D on the hybrid system. Performance on the hybrid system could be improved by proportionally adjusting the partitioning of computation to account for the higher efficiency of the XT4 nodes.

Work with GTC has focused on exploring opportunities for improving memory hierarchy utilization. One component of this effort has been studying the impact of data structure layout and code organization on the spatial and temporal locality present in data access patterns. A detailed study of GTC using a performance modeling toolkit developed at Rice[11] identified several opportunities for improving application performance. These included reorganizing the particle data structures to improve spatial reuse in the charge deposition and particle pushing phases of the application, using loop fusion to increase temporal reuse of particle data, and transforming the code to increase instruction-level parallelism and reduce translation look-aside buffer misses. A study of the transformed code on an Itanium2 system showed that our code transformations improved performance by 33%. Code modifications have been provided back to the application team. Ongoing work is exploring on-line adaptive reordering of particle data to improve temporal locality for the cell data structures during the charge deposition and particle pushing phases. Preliminary experiments indicate that this approach offers the potential for substantially improving performance.

An outcome of the CScADS summer workshop *Libraries and Algorithms for Petascale Applications* was a substantial improvement in I/O scaling and performance of the Omega3P simulation tool under development at the Stanford Linear Accelerator Center. Discussions at the workshop led to the use of collective communication patterns to avoid scaling bottlenecks associated with reading input data. Additionally, adjusting the application to use parallel netCDF and MPI-IO reduced the time for writing output data by a factor of 100 when Omega3P was run on thousands of processors on the Cray XT system at Oak Ridge. These improvements dramatically enhanced the scalability of Omega3P.

Acknowledgement
The Center for Scalable Application Development Software is supported by cooperative agreement number DE-FC02-07ER25800 from the Department of Energy's Office of Science.

[1] CScADS - http://cscads.rice.edu/

[2] HPCToolkit - http://www.hipersoft.rice.edu/hpctoolkit/

[3] Paradyn - http://www.paradyn.org/

[4] Whaley, C., Petitet, A., Dongarra, J. "Automated empirical optimizations of software and the ATLAS project," *Parallel Computing*, Vol. 27 (2001), no. 1, pg. 3-25.

[5] Frigo, M. "A fast Fourier transform compiler," *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, Atlanta, Georgia, May 1999.

[6] Qasem, A., Kennedy, K. "A cache-conscious profitability model for empirical tuning of loop fusion," *Proceedings of the 2005 International Workshop on Languages and Compilers for Parallel Computing*, Hawthorne, NY, October 20-22, 2005.

[7] Zhao, Y., Kennedy, K. "Dependence-based code generation for a CELL processor," *Proceedings of the 19th International Workshop on Languages and Compilers for Parallel Computing (LCPC)*, New Orleans, Louisiana, November 2 - 4, 2006.

[8] Monroe, D. "Energy science with digital combustors," *SciDAC Review*. Fall 2006. http://www.scidacreview.org/0602/html/combustion.html

[9] Krieger, K. "Simulating star power on earth," *SciDAC Review*, Spring 2006. http://www.scidacreview.org/0601/html/fusion.html

[10] Mellor-Crummey, J. "Harnessing the power of emerging petascale platforms. SciDAC 2007," *Journal of Physics*: Conference Series 78 (2007) 012048.

[11] Marin, G., Mellor-Crummey, J. "Understanding unfulfilled memory reuse potential in scientific applications," *Technical Report TR07-6*, Department of Computer Science, Rice University, October 2007.

URL to article: http://www.ctwatch.org/quarterly/articles/2007/11/creating-software-tools-and-libraries-for-leadership-computing/