LACSI Impact on ASCI Projects at LANL

Draft Report

November 7, 2004

Most of the LACSI projects are driven by long-term goals. However, these projects are structured to deliver intermediate results that have impact in both the short and medium term. Many of these intermediate payouts have had significant direct impact on the ASCI weapons programs; others will bear fruit in the near future. It is a goal of this report to elaborate these impacts.

The material in this report is structured according to a series of projects, with the impact of each project detailed separately. In some cases, particularly with the more mature projects, these impacts have already been realized. In other cases, the research projects have been redirected to deliver future impacts, based on our collaborations with LANL staff members.

Performance Analysis Tools

Rice computer scientists have designed and implemented HPCToolkit, a suite of novel performance analysis tools that are dramatically simpler to use and provide a superior user interface for analyzing multiple factors affecting the node performance of scientific programs.

These tools support effective analysis of the node performance of large-scale scientific applications consisting of thousands of procedures, hundreds of thousands of lines of code, and external (possibly binary-only) libraries. The toolkit is designed to work directly with optimized application binaries. By doing so, the toolkit is language independent and it avoids the need for manual instrumentation, changes to the build process, and recompilation. A feature that distinguishes HPCToolkit from any other performance analysis tool is that it analyzes application binaries, recovers information about the loop nesting structure in the program, and maps performance back to the source code at the loop level—the level that matters for optimized scientific programs. The toolkit is multiplatform (including Alpha+Tru64, MIPS+IRIX, Pentium+Linux, Opteron+Linux, and Itanium+Linux)—a range of architectures of past, current, and emerging interest to ASCI in general and LANL in particular.

HPCToolkit supports scalable data collection using hardware performance monitors for analysis of both serial and parallel codes. To facilitate interpretation of performance data, toolkit components combine measured metrics with program structure information from the application binary and correlate it with the application source code. The toolkit includes an intuitive browser for interactive exploration of application performance data. A novel and important feature of the performance browser is its support for top-down analysis, which is essential for effective analysis of large codes.

Recent work at UNM is designed to complement the performance monitoring work being done at Rice University. While Rice's HPCToolkit focuses primarily on CPU performance on a single machine, new work begun at UNM is focused on fine-grained, lightweight measurement of operating system and networking software performance. Recent work at CCS-3 has shown that operating system performance is critical to exposing the full capabilities of large-scale machines to ASCI applications such as SAGE; Prof. Patrick Bridges at UNM has recently begun work on a new performance monitoring system designed to measure OS and message passing performance at runtime with minimal overhead. This system is designed to enable the deployment of operating system and runtime components that can adapt their behavior to the varying needs of ASCI applications.

Training. In 2003, we held two performance analysis workshops at LANL to train LANL personnel on how to use LACSI-supported tools for performance analysis of their codes. LACSI researchers provided hands-on training to members of the Telluride, Shavano, Crestone, Eolus code teams, as well as members of the transport group from X division. In the first workshop, researchers from Rice, UH, Illinois and Tennessee worked with LANL scientists on codes including Truchas, SAGE, and FLAG. The Eolus team was represented in the sessions, but declined to study their code at the workshop because of export control concerns. In addition to the workshop attendees, the Rice team also assisted the Partisn team in using HPCToolkit to study their code's performance in an extended teleconference.

In the second performance analysis workshop, teams from Rice (HPCToolkit) and UNC/Illinois (SvPablo and DynaProf) returned to LANL to work with broader groups from the Crestone and Shavano teams. Each tool team was teamed with a code team for a full day and we switched for the second day. SvPablo complements HPCToolkit by enabling source code instrumentation of application codes and correlation of hardware and software metrics, together with support for dynamic application adaptation. Based on experiences in the workshops and on our interactions with LANL application researchers, the Illinois/UNC team enhanced SvPablo to simplify instrumentation of multi-file codes with integrated build procedures.

The Tennessee team worked with Harvey Wasserman, Olaf Lubeck, and Michael Lang on the PAPI installation on the Opteron system as well as answered questions about performance monitoring hardware.

Scalability. Large-scale systems posed many scalability challenges, ranging from simple measurement of system and application behavior to predicting application scalability with varying numbers of processors. To enable efficient measurement of large systems, the Illinois/UNC team developed a new statistical sampling methodology that uses measurements from a statistically valid subset of components to derive aggregate performance estimates. Statistical sampling provides a formal basis for quantifying the resulting accuracy of the estimation, and guides the selection of a subset that meets accuracy specifications.

Based on these experiences and insights, the Illinois/UNC team also enhanced the SvPablo toolkit with new scalability analysis capabilities. One capability of this software is to estimate bottleneck migration in large codes. The scalability analysis module enables the aggregation of performance data from multiple code executions, with varying numbers of processors, and the visualization of resulting scalability plots. SvPablo creates plots on demand as the user clicks on a given construct in the application's source code.

Deployment for Production. The Rice team has worked with members of CCN-8 (including Brett Kettering, Susan Post, and Chip Kent) to deploy HPCToolkit on all major platforms at LANL. To date, it has been deployed on open and classified SGI Origin, AlphaServer SC, and Opteron systems. The Rice team is working with CCN-8 to enhance the capabilities of HPCToolkit to provide superior support for performance diagnosis on Clustermatic Opteron clusters. In particular, the Rice team is integrating HPCToolkit with kernel support for system-wide performance data collection.

HPCToolkit is being used by LANL for performance evaluation of stockpile stewardship applications. In particular, we were informed that it was used for analysis of FLAG for the ASCI burn code review in August 2003. (See “Analysis and Tuning of ASCI Applications”, below.)

The Illinois/UNC team deployed the new SvPablo version on the external Alpha cluster at LANL, and will soon install it on LANL’s Opteron cluster. In addition, the Illinois/UNC team has been working with LANL staff (Susan Post and Chip Kent) to deploy the new SvPablo tools on LANL’s secure systems.

Adaptability, Scalability and Fault-Tolerance

Work at Illinois (and now at UNC) and University of Tennessee has focused not just on development of traditional performance tools, but also on the next generation of performance scalability and fault tolerance problems to be encountered by LANL and ASCI applications. With emerging systems containing thousands (and soon tens of thousands) of processors, new approaches are needed that can estimate performance efficiently, enable adaptability in the face of component failures, and ensure high application performance.

Reliability and Fault Tolerance. Today, clusters built from commodity PCs dominate high-performance computing, with systems containing thousands of processors now being deployed. As node counts for multi-teraflop systems grow to thousands and with proposed petaflop system likely to contain even more nodes, the standard assumption that system hardware and software are fully reliable becomes much less credible. Concomitantly, understanding application sensitivity to system failures is critical to establishing confidence in the outputs of large-scale applications and in specifying systems and architectures than can continue operation given component failures.

To assess current system reliability and to extrapolate to coming systems, we have collected and analyzed system failure logs from ASCI and ASCI-class systems. This data has shown that at smaller scales software and operator errors are the most common causes of system failures. At larger scales, hardware component failures (e.g., power supplies, disks and networks) are the most common. This suggests that next-generation procurements must include higher mean time between failure (MTBF) specifications for components and that software development efforts must include detection and recovery mechanisms for transient and hard component failures. This work was highlighted at a recent LANL Director's colloquium.

In addition, we have experimentally assessed the susceptibility of applications to transient memory and communication failures by injecting faults into executing applications. We found that these failures can be manifest in many ways, from application failure to, more perniciously, incorrect output. Application assertions and internal consistency checks can detect some of these errors, albeit at the expense of additional execution time. Because the MPI standard has minimal support for error detection and recovery, new approaches are needed to augment existing capabilities.

Hence, we have investigated schemes to minimize the effects of such failures on applications. We extended the Los Alamos MPI (LA-MPI) package, based on a discussion with Rich Graham, to support in-memory checkpointing, using spare nodes, rather than having the application move the data onto disks as in traditional checkpointing. Under this diskless checkpointing scheme, the parity checksum over the checkpoint data is calculated and stored on spare nodes. This checkpointing scheme complements disk-based checkpointing by interleaving memory-based checkpoints between disk-based checkpoints, reducing the overhead for disk I/O and allowing semi-transparent application recovery from component failures.

Reflecting the need for greater focus on scalability and fault tolerance, we have organized, with LANL researches Adolfy Hoisie and Darren Kerbyson, a special issue of the journal *Future Generation Computer Systems*. This issue will contain eight refereed articles related to performance issues in large systems. In addition, we are jointly organizing a workshop at the LASCI symposium on large-scale performance measurement and fault tolerance.

The Tennessee team has been working extensively with Rich Graham and others at LANL to develop an OPEN-MPI implementation.

Fault-Tolerant Algorithms. The Tennessee team has been working on fault-tolerant methods for iterative solutions of large sparse systems of equations. The goal of the FT-LA project is to discover and evaluate algorithms and techniques for fault-tolerant linear algebra operations on massively parallel computing environments, and encode the results in software tools that facilitate the research and help disseminate its benefits. Accordingly, FT-LA's research program divides roughly into three main parts. The FT-LA project aims to

1. *Discover and develop algorithmic innovations*, focusing on novel "lossy" algorithms that avoid checkpointing and its costs altogether but still reach the solution with reasonable performance, and on coding algorithms for algorithm-based diskless checkpointing scenarios that can succeed in massively parallel situations, where they have previously been unusable.
2. *Analyze parameter-driven performance and error propagation factors* of our new algorithms, enabling accuracy and performance through prediction and parameter selection, and leading to simulation models that support the automatic choice of optimum runtime parameters.
3. *Develop and disseminate software for creating fault tolerant numerical libraries*, facilitating both FT-LA research and enabling the community to utilize its positive results to implement resilient versions of key modules and to integrate them with libraries of choice.

Adaptability. Working with the LANL/Radiant research group (Wu Feng), the Illinois/UNC team integrated their Autopilot adaptive control toolkit with the LANL MAGNET package. The integrated toolkit contains sensors that can export critical data from a Linux kernel and actuators that enable external control of the kernel operation. We demonstrated the initial prototype of the integrated toolkit during SC03 at the LANL research booth. Based on these preliminary results, we have submitted a joint proposal with Feng to expand this prototype for network performance assessment.

We have also conducted studies on the trade-offs between performance and power consumption. These studies were motivated by the observation that power could become a critical factor for petascale systems, comprising thousands of processing elements. An Illinois student spent a summer working with Feng to explore power-aware tradeoffs for large-scale systems. This work and related work was extended as an M.S. thesis on power-aware parallel I/O for high performance storage systems. During his internship, the student developed an integrated hardware/software solution to conserve power in parallel applications running on clusters.

Analysis and Tuning of ASCI Applications

Rice computer scientists have used ASCI applications to drive research in compiler technology and run-time libraries for large-scale scientific codes. This work has two goals: (1) to identify long-term research challenges in algorithms, data structures, and compiler technology that are motivated by problems faced by ASCI applications and other large-scale scientific codes, and (2) to identify important code improvements that can be performed manually in the near term to improve performance of the ASCI workload. In the course of this work, we studied several unclassified ASCI applications, identified opportunities for improving application performance, and reported our results back to code teams at LANL. Several of these improvements have been particularly noteworthy. We mention a few of our direct impacts on ASCI applications below.

SAGE. Rice computer scientists identified a subtle but serious performance degradation in SAGE that resulted from unnecessary copying of on-node data. The copying was a result of the strategy employed by the code for managing off-processor data. We performed experiments that demonstrated substantial performance improvement by removing this bottleneck. We advised members of SAGE team (Mike Gittings and Tom Betlach) about a strategy for restructuring SAGE's handling of off-processor data and its benefits. Improvements they implemented roughly doubled solver performance on Blue Mountain, and boost it by roughly 50% on ASCI Q.

Our further study of SAGE also identified that sparse-matrix-vector product computation, a key component of many ASCI codes, tends to be inefficient on modern microprocessor-based systems. Working with a benchmark derived from SAGE timing tests, Rice computer scientists identified an interaction between sparse matrix representation, sparse matrix data, and compilers that was responsible for low sparse-matrix-vector product performance in SAGE. We devised a new sparse-matrix representation that makes it possible to reorganize sparse matrix computation for higher efficiency. Working with Harvey Wasserman at LANL, we demonstrated that this new sparse-matrix-vector multiplication improves sparse-matrix vector product performance in SAGE (with on-node data) by a factor of 2 on Itanium2 and by 47-71% on Power3-II. Computation on off-node data could benefit by applying the same technique.

Sweep3D. As a transport application, sweep3D represents an important part of the ASCI workload. At Rice, it has served as a driving application motivating the development of performance tools including a memory hierarchy simulation package and the HPCToolkit performance tools. Although the memory hierarchy simulator was principally built for internal use for studying applications to gain insight into opportunities for compiler-based improvement, it has also seen external use at Sandia. Sweep3D has been the subject of several investigations into program transformations for improving memory hierarchy performance. As a result of this work, several higher performance variants of Sweep3D were provided back to LANL with a report showing 20% single-processor speedup. Further investigation uncovered additional opportunities for applying program reordering transformations (in particular, loop fusion). The transformed version of this application delivers 44% higher single-processor performance on Alpha systems on a 150^3 problem size. The same version delivers 90% improvement on single processors of the SGI Origin.

Blanca. The Blanca project was encountering a severe performance bottleneck in AMR vertex and zone setup. Jay Mosso and Richard Barrett produced a representative standalone unclassified code that showcased the problem. Analysis by the Rice team showed that the algorithm and data structure for maintaining an ordered collection of vertices was a severe bottleneck for large problem sizes. We implemented two alternative strategies – one for maintaining an ordered collection on the fly and a second for sorting and eliminating duplicates. The algorithm and data structure changes improved the performance asymptotically from $O(n^2)$ to $O(n \log n)$, which had the effect of improving performance by a factor of 26 for the 5-level refinement test case.

CHAD. CHAD uses an irregular mesh data structure at the heart of its computation. Rice computer scientists analyzed the impact of data ordering on cache performance in CHAD. We showed that intelligent data orderings based on space-filling curves can double performance over naïve orderings for irregular and adaptive applications. Experiments with the volume module in CHAD showed that inadequate loop fusion and ineffective scalarization of Fortran 90 array notation increased the memory traffic by 40%. This experience was the motivation for compiler research by Rice computer scientists into better algorithms for scalarization of Fortran 90 array operations, loop fusion, and array contraction to reduce memory traffic.

Performance Modeling

The Performance and Architecture Laboratory at LANL has developed powerful techniques for modeling the scalability of parallel codes on large-scale systems. One limitation of their approach is that they lack the ability to accurately predict node performance of whole complex codes on systems that are unavailable – this includes systems such as the Earth Simulator as well as future systems such as processor-in-memory systems. Rice computer scientists have developed a complementary strategy for semi-automatically synthesizing cross-architecture performance predictions for whole programs. Prediction tools developed at Rice have been used to make accurate cross-architecture performance predictions (e.g. accurately predict performance on MIPS and Itanium systems given only SPARC binaries) and to study the impact on performance of changing memory hierarchy characteristics.

A key feature of our strategy for developing models for predicting node performance is that the predictions are scalable. Experiments with modest problem sizes can be used to accurately predict node performance characteristics for significantly larger problem sizes that may not be feasible to simulate (or even to run) on today's systems. A Rice student interned with the PAL group during the summer of 2003. Our aim is to integrate the complementary technologies developed at Rice and LANL to better support scalable cross-architecture predictions. A key goal of this work is to influence the design of emerging petascale machines by projecting how well their proposed architectures will support ASCI workloads.

Open-Source Compiler Technology for ASCI Applications

Economic forces in the marketplace mean that relatively little attention is paid to the performance of scientific applications on commodity microprocessors; the emphasis of ongoing development is targeted primarily at supporting business applications. As a result, there is a little focus on research and development of compiler technology to boost the performance of scientific codes. In part, this also hinders technology transfer because vendors have little interest in incorporating new algorithmic techniques in their compilers unless these techniques will boost performance of commercial workloads in addition to scientific ones. For this reason, it is necessary for universities to directly transfer compiler technology for improving the performance scientific codes to end users – government laboratories in particular – through open source compilers.

There has been considerable interest from the DOE in general and LANL in particular in supporting development of open source compilers. The NNSA OSSODA RFI is a recent byproduct of that interest. Rice was involved in a response to that RFI and has been active in subsequent meetings with the DOE steering committee to formulate a plan that will best meet the needs of the NNSA and make it possible to deploy results of computer science compiler research for use on production codes at the national laboratories.

Building source-to-source program translators is an effective way to transfer compiler technology that is developed in the course of university research on high-level high-impact optimizations for scientific programs. For this reason, Rice computer scientists have been working to create an open-source software platform that can transform production Fortran 90, C++ and C code based on the Open64 compiler infrastructure. Rice University is leading a collaborative effort that now has active developers at Rice Argonne National Laboratory, and Lawrence Livermore National Laboratory, in addition to international collaborators at the University of Vienna and the Universitat Politècnica de Catalunya. As this system has matured, Craig Rasmussen (LANL ACL) has expressed interest in using it for program transformation. Also, the Fortran 90 source-to-source transformation infrastructure we have developed in this project is supporting development of Fortran 90 support for automatic differentiation (described in the next section).

With supplementary funding from the DOE Office of Science, the Open64 compiler infrastructure is being used as the infrastructure supporting research and development of emerging programming languages including Co-array Fortran, which is under development at Rice, and Unified Parallel C (UPC), which is being developed by our collaborators at Berkeley. Ultimately, petascale machines will need to be programmed differently than today's parallel systems. Our ongoing research efforts to build compilers for global address space languages are intended as steps towards having compilers manage data movement and synchronization implicitly and simplify parallel programming for application scientists. There has been interest in Co-array Fortran at LANL, in particular from the Crestone code team. Experiments by Cray using SAGE on the Cray X1 required converting parts of it to Co-array Fortran for best performance.

Automatic Differentiation

LACSI has supported ongoing development of the ADIFOR automatic differentiation tool. This tool has been employed by a number of projects at LANL. Primary, it has been used for verification/validation and parameter identification/estimation. Rudy Henninger's group (currently CCS-2) has used ADIFOR on MESA 1D & MESA 2D (Armor/Antiarmor codes), "Caramana's Lagrangian Test Code" (used to study hydro methods that are implemented in the Shavanno project's FLAG code), and Truchas-1D (Telluride project). Ralph Nelson (Shavano Project) has used ADIFOR on a light-water-reactor safety code called TRAC. In addition to Henniger and Nelson, there are six additional LANL scientists who have registered for copies of the ADIFOR code. Three of them indicated that they obtained a copy of ADIFOR after seeing a talk at LANL by a Rice computer scientist. Current work is focused on applying the latest ADIFOR to Truchas 3D and to FLAG. To support automatic differentiation of Fortran 90 codes, ADIFOR is relying on compiler infrastructure for source-to-source transformation developed with LACSI support, as described in the previous section.

Compiler Technology for High Productivity Parallel Programming

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 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, while decreasing in whole system reliability because of the proliferation of components. Growing awareness of the looming productivity crisis has led to the creation of DARPA's High Productivity Computing Systems program. Recently, LANL's Jeff Brown has been advocating that NNSA support a high productivity computing research thrust.

Since the inception of LACSI, the Rice team has been exploring compiler technology to support high productivity parallel programming languages. High-level data parallel languages based on a global view of data 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.

For high-level models for data-parallel programming to be widely used, they must satisfy four criteria:

1. they must be be expressive enough to support a broad spectrum of sophisticated parallel algorithms,
2. they must deliver performance competitive with that of hand-coded parallelizations,
3. they must be ubiquitously available everywhere from emerging tightly-coupled architectures to desktop workstations, and
4. they must deliver appropriate levels of efficiency on each class of systems.

The Rice team has been investigating locality-aware compiler technology under LACSI support since the founding of the institute. The principal products of this research have been language-independent analysis and code generation techniques for managing single-threaded parallel programming languages. These techniques are applicable to a broad class of languages ranging from MATLAB to Cray's Chapel. This research has garnered two best paper awards to date and has been selected for publication in three journal special issues (among other publications). To date, this research has made significant progress in developing compiler technology that helps address the productivity problem by transforming complex programs written in a high-level single-threaded form into sophisticated scalable codes that run with hand-coded efficiency on large-scale parallel systems. Ongoing work in this area is focused on enhancing compiler capabilities to more effectively support adaptive and multi-level algorithms. A study in progress aims to evaluate the effectiveness of compiler technology developed by the Rice team for transforming a single-threaded HPF program developed for the Earth Simulator into one that can execute efficiently on parallel microprocessor-based clusters that have been the focus of ASCI investments.

Optimization of Object-Oriented Programming Languages

Several of the key LANL weapons codes include substantive code written in object-oriented languages, especially C++. It is our understanding that the Marmot project is considering using Python as well as C++ for coding. Object-oriented technologies present unique challenges for achieving the highest possible node performance. With these issues in mind, Rice has undertaken a study of optimizations for such languages. Although this study was conducted in the context of Java, the techniques are equally applicable to other languages, including C++. The principal focus of this work has been two different kinds of optimizations. First, a technique called “object inlining” can be used to eliminate the overheads, especially memory access costs, associated with the use of arrays of object data structures. Essentially, it replaces arrays of such objects with arrays of the instance variables for the given objects. Second, through the use of global type analysis, the overheads associated with dynamic dispatch, an important technique in object-oriented programming, can be reduced or eliminated. We have developed a global type analysis algorithm that can determine—in many cases precisely—the actual method being called at a dynamic method invocation site. In codes with a significant amount of virtual function use, this can have a huge payoff.

We have demonstrated the effectiveness of our compiler technologies on a particle-in-cell code, Parsek, written in object-oriented style in Java at LANL. With our collaborators from Los Alamos, have published these results in the Journal of Concurrency and Computation: Practice and Experience 2004. Our analysis and optimization infrastructure, JaMake, optimized the object-oriented version of Parsek to achieve a performance improvement of approximately 80% over the original code. The automatically produced code is very competitive with the hand-optimized JavaParsek.

Through our LACSI collaborators at LANL, we are now exploring ways to apply this technology to new codes developed in the Marmot project.

Component Integration Systems for HPC

Through the LACSI collaboration, the Rice team has come to understand the importance of applying sound software engineering strategies to ASCI codes. One such strategy is the extensive use of pre-developed, reusable software components for structuring the code. Such components can be independently tested, providing additional reliability to applications that employ them.

Currently, most component-based systems suffer significant overheads for crossing component boundaries. This tends to drive the developers of HPC codes toward heavyweight components. Although this approach has proven successful in several HPC codes at the DOE laboratories, it precludes some important structuring strategies, such as the mixing of data structure components (e.g., sparse matrices or adaptive meshes) with functional components (e.g., linear algebra) because such integration would incur high overheads for cross-component method invocation. Instead, developers often bundle the desired functionality into the data structure component (or vice versa) making it difficult to change the data structure or to use the same data structure with different functions.

To address these problems, the Rice component integration group has redirected its research effort to focus on a new component integration strategy that preprocesses collections of components with the aim of dramatically reducing component-crossing penalties through compiler optimization. Although our initial prototypes have used Matlab as the language for application and component implementation (this is because many library developers use Matlab, or another scripting language, for prototyping), the techniques are applicable to any implementation language. In fact, components written in Matlab are automatically translated to C or Fortran before being used in an application. Preliminary performance results show that application run times can be improved significantly, in some cases by integer factors.

We are now driving this research effort by working with the developers of applications within the Marmot project to understand how the technology can be effectively applied to future weapons codes. Although the effort is still in its preliminary stages, the goal is to have significant impact on future generations of the Marmot applications, increasing the ease of implementation without incurring undue performance costs.

To date, this effort has convened one workshop on components, and has had one full-day collaboration and design workshop with members of the Marmot team. Other workshops are being planned for the future.

Automatic Tuning of Components and Applications

Because of the substantive efforts involved in retuning applications when they are moved from one platform to another, LACSI researchers are involved in several projects to explore strategies for automatically tuning applications. These efforts are using experimental strategies that explore various ways to structure computationally intensive loop nests to achieve near-optimal performance on new platforms. This line of work is extremely relevant to the weapons program because it can lead to significant savings in tuning effort by application developers.

One model for the automatic tuning strategies is the ATLAS system developed at the University of Tennessee, a LACSI partner. ATLAS is a system for pre-tuning the LAPACK BLAS by extensive preliminary execution of the critical loop nest with different parameters for blocking and unrolling. Experiments have shown that ATLAS achieves performance that is very close to the best hand-tuned versions of the BLAS on a variety of platforms. This work has established that automatic tuning systems can be an effective substitute for human tuning, thus achieving a significant boost in programming productivity.

While ATLAS is focused on specific loop nests, several projects at Rice, Tennessee, and Houston are pursuing the goal of automatically tuning general applications and components. Rice researchers are pursuing the strategy of "adaptive compilation" which can be used to try, in a heuristic search, different orders of compilation passes, different compiler optimization settings, or different strategies for unrolling and inlining. This effort has already conducted the most extensive experiments on automatic tuning that have ever been attempted. The work has shown that large improvements over standard compilation can be achieved by finding the compilation order, parameter settings, and inlining strategy best suited to a particular platform.

A second group at Rice has been using LoopTool, which applies program transformations such as loop blocking and unroll-and-jam based on directives manually inserted in the program text, to explore automatic selection of transformation parameters. This work has shown that direct search strategies can reduce the search and execute time needed to pre-tune a general application to reasonable levels without significant loss of performance. Both Rice projects were reported on in papers selected for presentation at the 2003 and 2004 LACSI Symposia.

The Tennessee group is extending the ATLAS work to more general component tuning while the UH effort has focused on automatic tuning of components of modest complexity that are decomposable in different ways, such as typical linear algebra library components. The technique has so far been applied to FFTs. FFTs on most data set sizes can be decomposed in several different ways that not only result in different number of arithmetic operations but more importantly in a range of opportunities to schedule sets of operations for maximum exploitation of the memory system as well as of functional units. The approach taken in exploiting the degrees of freedom in decomposition and scheduling has two phases: one carried out at library build time, one at run-time. At build time highly optimized “kernels” or subcomponents are created and their performance measured and entered into a data base. At run time when actual problem sizes and data allocation are known alternative decompositions down to the kernel level are evaluated for performance and an execution plan generated for a particular decomposition with a schedule that seeks to minimize execution time. Various strategies that trade off the number of options evaluated against the expected performance gain from evaluating additional options are currently being explored. Though the actual decompositions depend on the function to be performed, such as the FFT, the methodology used should apply to common mathematical operations that are decomposable. In carrying out this development we have created a toolbox which we call CodeLab that should enable us to apply the technique to other important functions in ASC codes with a more modest effort than the one we have devoted to this first application of the methodology. Our next intended target is support for codes using multigrid techniques.

All of the automatic tuning work is directly relevant to ASC goals, because it will lead to tools that facilitate better and faster migration of existing weapons codes to new HPC platforms.

Operating Systems and Networking

Systems researchers at UNM and Rice are working with the LA-MPI (OpenMPI) team on performance issues. These will be critical for ensuring that LANL’s new generation of Linux clusters will meet performance and usability goals.

The Rice team has integrated an event-driven progress engine into LA-MPI. This improves overall responsiveness and performance of the LA-MPI TCP path. This is recent work, so it is not yet integrated back into the main source tree. We have shown that with this new event-driven progress engine, the performance of MPI over TCP using 1Gbps Ethernet is similar to 1.2Gbps Myrinet on the NAS benchmarks. With a few additional enhancements, we believe that we can beat Myrinet.

Much of the recent work at UNM has focused on protocol optimization for message-passing runtimes for large-scale machines in conjunction with Ron Minnich and Rich Graham of CCS-1. Patricia Crowley is researching TCP scaling issues on large-scale systems and is attempting to reduce resource usage in TCP. In the past, TCP has occasionally been used for message-passing in large-scale machines, though it is more frequently used for performance-critical tasks such as communication with NFS servers to read and write initialization data, checkpoints, and results. With processors containing thousands of nodes, limiting the resource usage of stateful protocols such as TCP while at the same time increasing their performance is essential to providing the full capabilities of the large-scale machines to ASCI applications. Galen Shipman, a new UNM graduate student, is researching custom protocols for the Los Alamos MPI implementation, LA-MPI. Galen's work is aiming to reduce protocol processing overheads for the performance-critical message passing path of LA-MPI, particularly when LA-MPI is used on commodity hardware such as Gigabit Ethernet.

Common Hardware Infrastructure: To streamline the sharing and deployment of research and software artifacts among the LACSI academic partners and LANL, we have begun the procurement by the partners of development systems very similar to the Blue Steel system in CCS-1. These are clusters of dual-processor Opteron nodes with both Gigabit Ethernet and Infiniband interconnects. Because these systems will use the Clustermatic Linux software stack, the products of the academic research will be much more easily deployed at LANL and a wider range of academic personnel will be able to work directly with the systems of interest. Furthermore, these configurations will ensure that academic partners will be addressing the unique issues of using Infiniband as a communication fabric for high performance computing. Rice, UNM, and UNC have placed orders for these systems.

New Polyhedral-Mesh Diffusion Discretizations

Yuri Kuznetzov of UH, working in collaboration with researchers in LANL group T-7, has developed a new arbitrary polyhedral-mesh diffusion discretization scheme that offers several significant advantages over previous methods. The support-operators methodology was used to formulate this discretization. A breakthrough resulted from a key idea contributed by Kuznetsov for forming vector inner products on a general polyhedron/polygon by using a decomposition of the polyhedron/polygon into a set of tetrahedra/triangles in conjunction with an interior-cell vector interpolation scheme and a vector inner product expression for a single tetrahedron/triangle. The resulting diffusion discretization scheme is second-order accurate, symmetric positive-definite, and exactly treats material discontinuities.

LANL researchers Shashkov and Lipnikov are now working with an X-3 team to implement the 3D geometry version of this scheme to model diffusion processes in the Project B. The scheme currently existing is non-symmetric. Thus it is fully expected that the equations for the new scheme will cost far less to solve than the existing equations because the symmetric positive-definite property of the new equations enables them to be solved with a conjugate-gradient method that is far more efficient than the restarted GMRES method currently used to solve the existing equations. Finally, the new scheme is known to be more accurate than the existing one (same order accuracy, but a significantly smaller coefficient for the leading error term). The new discretization method can be used for various types of meshes, including nonmatching and sliding meshes, as well as meshes with local refinements.

Future plans call for the testing of the new methods for 3D problems relevant to ASCI applications versus the existing discretizations schemes and development, investigation, and evaluation of robust and efficient parallel preconditioner for new discretization method on arbitrary distorted polyhedral meshes.