Priorities and Strategies

Los Alamos Computer Science Institute

On March 18th and 19th the Los Alamos Computer Science Institute (LACSI) Executive Committee and Principle Investigators met to discuss methods of addressing issues raised in the 2001 LACSI Contract Review. The body was tasked to develop priorities and strategies to meet future programmatic and LANL computer science needs.

A framework was developed to address long term strategic thrust areas. Specific objectives were called out as near term priorities. The objectives were folded into the framework to form a coherent planning view.

The framework outlined five strategic thrust areas:

- Components
- Systems
- Foundations of Computational Science
- Application Performance
- Computer Science Community Interaction

This document presents the research vision and implementation strategy in each of these areas.

Components

Ken Kennedy (ken@rice.edu)
Jack Dongarra (dongarra@cs.utk.edu)
Doug Kothe (dbk@lanl.gov)
Craig Rasmussen (crasmussen@lanl.gov)
Brian VanderHeyden (wbv@lanl.gov)

The goal of the component architectures effort is to make application development easier through the use of modular codes that integrate powerful components at a high level of abstraction. Such integration systems can be viewed as providing a "software backplane" into which various components can be plugged to add new strategies for solving fundamental problems such as hydrodynamics, mesh generation, and transport.

Through modularization and the existence of well-defined component boundaries (specified by programming interfaces), components allow scientists and software developers to focus on a their own areas of expertise. For example, components and modern scripting languages allow physicists to program at a high level of abstraction (by composing off-the-shelf components into an application), leaving the development of components to expert programmers. In addition, because components foster a higher level of code reuse, components provide an increased economy of scale, allowing resources to be shifted to areas such as performance, testing, and platform dependencies, thus improving software quality, portability, and application performance.

A fundamental problem with this vision is that Los Alamos application developers, and most others in science, cannot afford to sacrifice significant amounts of performance for this clearly useful functionality. Therefore, an important part of the effort will be the exploration of integration strategies that perform context-dependent optimizations automatically as a part of the integration process. This theme defines a significant portion of the research content of the work described in the remainder of this section.

Short-term Goal

Component Frameworks Review

Subproject Leads: Craig Rasmussen, LANL, crasmussen@lanl.gov; Ken Kennedy, Rice, 713-348-5186, ken@cs.rice.edu

The focus of this subproject will be to conduct a study of available component frameworks. This will be done to support the goal of integrating high-performance component technology into strategic LANL applications and into the LANL software culture. A team of both LANL staff and academic partners will carry out this survey. This team will establish the requirements for component architectures that fit the needs of the Los Alamos code-development environment. This effort should be viewed as a precursor to a code project that would update a major hydrodynamics application to a more modern and maintainable form.

In conducting the study, the team should draw, to the maximum extent possible, on recent advances in the design and implementation of complex software systems. The approach should attempt to understand the value and impact of incorporating new languages, such as Java and Python, and new compiler strategies for addressing the problem. In addition, the connection to the Common Components Architecture (CCA) effort should be a major consideration. In examining recent advances, the team should consider the ease of adapting existing codes to take advantage of new software technologies. Issues such as language interoperability and the ability to create components out of existing codes should be carefully considered.

Tasks:

- Assemble team and establish LANL-specific requirements for component architectures (Quarter 1).
- Initiate review of existing component architectures (Quarter 2).
- Pick one or two of the most promising frameworks and test them on a range of code types.
- Produce report and recommendations (Quarter 4).

Long-term R&D

Once the problems and current solution strategies are well understood, the Components research effort should focus on long-term research and development projects that will not only address the problem effectively when they mature many years in the future, but also provide important short- and medium-term payouts for ASCI and Los Alamos applications.

A major goal of this research should be to address the trade-off between generality of programming systems and the performance that applications written in them can deliver. Today, many high-level problem-solving systems exist, but the performance penalty for using them is severe. Is this situation an immutable law of nature, or merely an artifact of the implementation approaches we have pursued to date? The proposed research efforts on telescoping languages and high-performance Java for prototyping attempt to address this issue.

A second major question in this area is: Can we build components with the built-in ability to adapt with high performance to new computational platforms? One approach that has proved successful is the Atlas system, which uses substantive amounts of computation to provide versions of a computational linear algebra kernel that are highly tuned in advance to different machines. If this approach can be extended more generally to components of all kinds, it would help avoid the enormous costs involved in retargeting applications to different machines.

In the sections that follow, we will elaborate on some promising research directions addressing these issues.

Generation of Problem-Solving Systems Through Component Integration

Subproject Leads: Ken Kennedy, Rice, 713-348-5186, ken@cs.rice.edu; Craig Rasmussen, LANL, crasmussen@lanl.gov

The goal of this research is to develop compiler technologies and library designs that will make it possible to automatically construct domain-specific development environments for high-performance applications. This effort will develop advanced compiler technology to construct high-level programming systems from domain-specific libraries. Programs would use a high-level scripting language such as Matlab to coordinate invocation of library operations. Scripting languages typically treat library operations as black boxes and thus fail to achieve acceptable performance levels for compute-intensive applications. Previously, researchers have improved performance by translating scripts to a conventional programming language and using whole-program analysis and optimization. Unfortunately, this approach leads to long script compilation times and has no provision to exploit the domain knowledge of library developers.

To address these issues we will pursue a new approach called "telescoping languages," in which libraries that provide component operations accessible from scripts are extensively analyzed and optimized in advance. In this scheme, language implementation would consist of two phases. The offline translator generation phase would digest annotations describing the semantics of library routines and combine them with its own analysis to generate an optimized version of the library, and produce a language translator that understands library entry points as language primitives. The script compilation phase would invoke the generated compiler to produce an optimized base language program. The generated compiler would (1) propagate variable property information throughout the script, (2) use a high-level "peephole" optimizer based on library annotations to replace sequences of calls with faster sequences, and (3) select specialized implementations for each library call based on parameter properties at the point of call.

We will use this strategy to construct a high-level application development environment for an application of interest to LANL based on Matlab. This system could be seen as a flexible replacement for POOMA.

We also plan to extend these programming systems to prepare applications for execution on computational grids. If this effort is to succeed, it must take into account two important realities. First, many components will be constructed using object-oriented languages, so techniques for optimizing such languages are critical. Second, the execution environments for the resulting programs are likely to be distributed, so the implementation must take into account the performance implications of distributed systems, even if the applications are compiled together. For these reasons, basing a significant portion of the work on the Java programming language makes sense. Java is portable and includes distributed computing interfaces. However, we must overcome one major drawback of Java if it is to be used in scientific computation, namely its less-than optimal performance. Although we intend to focus on Java, many of the strategies developed for Java will extend to other object-oriented languages, such as C++.

With these considerations in mind, we plan to pursue research in two fundamental directions:

Toolkits for Building Problem-Solving Systems: The effort will focus on the production of tools for defining and building new domain specific PSEs, including:

- Tools for defining and building scripting languages.
- Translation of scripting languages to standard intermediate code.
- Frameworks for generating optimizers for scripting languages that treat invocations of components from known libraries as primitives in the base language.
- Optimizing translation of intermediate language to distributed and parallel target configurations.
- Tools for integrating existing code.
- Demonstration of these techniques in specific applications of interest to ASCI and LANL.

An important goal of this effort is to make it possible to build highly efficient applications from script-based integration of pre-defined components. Building on the component architecture efforts described in this section, we will pursue the novel strategy of "telescoping languages" to make it possible to extend existing languages through the use of software components.

Component Design for Integration: This effort will focus on the design and specification of components that can be used in a PSE for high-performance computation. Significant issues will be flexibility and adaptability of the components to both the computations in which they are incorporated and the platforms on which they will be executed. In addition, these components must have architectures that permit the effective management of numerical accuracy.

Retargetable High-Performance Applications

Subproject Lead: Ken Kennedy, Rice, 713-348-5186, ken@cs.rice.edu

For many years, retargeting of applications for new architectures has been a major headache for high performance computation. As new architectures have emerged at dizzying speed, we have moved from uniprocessors, to vector machines, symmetric multiprocessors, synchronous parallel arrays, distributed-memory parallel computers, and scalable clusters. Each new architecture, and even each new model of a given architecture, has required retargeting and retuning every application, often at the cost of many person-months or years of effort.

Unfortunately, we have not yet been able to harness the power of high-performance computing itself to assist in this effort. We propose to change that by embarking on a project to use advanced compilation strategies along with extensive amounts of computing to accelerate the process of moving an application to a new high-performance architecture.

To address the problem of application retargeting, we must exploit some emerging ideas and develop several new technologies.

- First, we will exploit the recent work on automatically tuning computations for new machines of a given class. Examples of effective use of this approach include FFTW, Atlas, and UHFFT. The basic idea is to organize the computation so that it is structured to take advantage of a variety of parameterized degrees of freedom, including degree of parallelism and cache block size. Then, an automatically generated set of experiments picks the best parameters for a given new machine. This approach has been extremely successful in producing new versions of the LAPACK BLAS needed to port that linear algebra package to new systems.

- Recent work on cache-oblivious recursive algorithms has shown that it is possible to use recursion to avoid the need of selecting block sizes for memory hierarchy. Furthermore, work by faculty at Rice has established that excellent recursive algorithms can be automatically constructed by advanced compiler technologies, especially transitive dependence and iteration-space slicing. This work has succeeded in blocking LU decomposition without pivoting and simple matrix multiplication, achieving performance at least as good and usually much better than hand blocked versions. In addition, it has generated a credible blocking for LU decomposition *with* pivoting, something that was until recently thought to be impossible in a dependence-based system. This powerful analysis can be used to generate versions of numerical algorithms that are portable across a variety of architectures.

- New work is needed to extend the work on automatic tuning to arbitrary codes, so that programmers are not forced to laboriously convert each application to a format that is amenable to the Atlas-style experimental tuning strategy. Here we propose to develop a conversion tool that uses deep compiler analysis to translate critical loop nests to a form where automatic tuning can be effectively applied. This strategy will involve extensive use of symbolic strip-mining of loops. In addition, it will require exploitation of a new strategy called *overpartitioning* that can be used to provide flexible mapping of grid computations to machines of different configurations. The basic idea of overpartitioning is to tile a computation to a size that represents a minimum computation that makes sense to carry out on a single processor. There will usually be many more tiles than processors as a result, so each processor will be responsible for carrying out the computations associated with several tiles. The strategy for grouping tiles would be selected for each new architecture based on extensive tuning experiments, much in the way Atlas tunes the BLAS.

We propose to conduct research on the topics described in the previous sections and to use the results of this effort to construct at least one retargetable application of interest to DOE and the ASCI program.

Rapid Performance Prototyping of Algorithms

Subproject Leads: W. B. VanderHeyden, LANL, (505) 667-9099, wbv@lanl.gov, Ken Kennedy, Rice, 713-348-5186, ken@cs.rice.edu

The goal of this sub-project is to develop tools for the rapid prototyping of algorithms and models for use in complex large-scale scientific simulation codes. The approach will be to capitalize on two successful efforts—one at Los Alamos and one at Rice University—involving the use of the Java programming language as a basis for rapid prototyping tools. Java has a number of attractive features for the development of rapid prototyping tools, including full support for software objects, parallel and networking operations, relative language simplicity, type-safety, portability, and a robust commercial marketplace presence leading to a wealth of programmer productivity tools.

Rapid Prototyping Environment: This sub-project will build on the CartaBlanca framework to explore new design patterns for advanced component architectures. The CartaBlanca framework has been shown to be a promising rapid prototyping environment for both method and model development. For example, an interface-tracking scheme was developed using CartaBlanca and has been successfully transferred into ASCI performance codes. The success of CartaBlanca is due to its component-like design and its use of object-oriented Java and to the productivity enhancing features of the Java language itself like strong typing, clean design, etc. Both the Java language and the component design of CartaBlanca are serious candidates for future component architecture for Los Alamos integrated software applications. In addition, Rice University offers significant Java and Java-related expertise that can be brought to bear on these issues. Three examples are automatic differentiation for optimization and sensitivity analysis, advanced Java compilers for object inlining, and distributed shared memory systems. The work here will expand the use of the component architecture in CartaBlanca to include the various physics classes such as multiphase flow, heat transfer and phase change, radiation, turbulence, etc. The objective is to design an architecture that easily allows for the interchange of different components of differing levels of sophistication to enable high developer productivity and paths for future method and model improvements.

Compilation of Object-Oriented Languages: To make it possible to move rapidly from prototyping to high-performance application development, compilation strategies must be developed for object-oriented languages such as Java and C++. This should include interprocedural techniques such as inlining driven by global type analysis and analysis of multithreaded applications. This work would also include new programming support tools for high-performance environments. Initially, this work will focus on Java, through the use of the JaMake high-level Java transformation system developed at Rice. This system will include two novel whole-program optimizations, “class specialization” and “object inlining,” which can improve the performance of high-level, object-oriented, scientific Java programs by up to two orders of magnitude. Later we will consider extensions to other object-oriented languages.

Tasks:

- Develop a design for component architecture for the physics classes in CartaBlanca (Quarter 1).
- Implement component architecture design and demonstrate for radiation hydrodynamics and compressible shock flows. (Quarter 2)
- Test an automatic differentiation tool for Java supplied by Rice (Quarter 3).
- Demonstrate Rice JaMake compiler on Los Alamos particle code. (Quarter 4).

Systems

Rod Oldehoeft (rro@lanl.gov)
Rob Fowler (rjf@rice.edu)
Edward Angel (angel@cs.unm.edu)
Thomas Caudell (tpc@eece.unm.edu)
Lennart Johnsson (johnsson@cs.uh.edu)
Barney Maccabe (maccabe@cs.unm.edu)
John Mellor-Crummey (johnmc@rice.edu)

Research Thrusts

1. Reliability

The high-end systems we envision will be even more complex than those we use today, which are even now dangerously close to escaping our ability to operate reliably. Extensive R&D is needed to tame this complexity so that the power of future systems will actually be available for productive work.

Anticipation of failure: We will develop systems that can monitor the status of various components and take action to keep applications running as failing parts are mapped out and replaced. This will require further development of technologies in monitoring, rapid migration of processes in any state, component virtualization and hot-swapability, and rapid resumption. Work in application checkpointing needs to be more automated to require smaller amounts of saved data and checkpoints need to be non-synchronous across an application for increased overall performance.

Response to failure: Messaging systems will be developed for future interconnects to prevent errors (dropped or corrupted messages, loss of a link or a NIC) from terminating an application. Techniques, both manually programmed and automated by compilers, for applications to continue to function when a component computation fails will be developed and deployed.

In either approach, alternate programming models or compilation techniques may be needed.

2. New Methods for Procuring and Developing Systems

Modeling applications and architectures: We have developed models that combine applications of relevance to LANL along with architectural descriptions. They are being successfully used to characterize how these applications will perform on systems available to us. These models will be altered to encapsulate systems under consideration for acquisition to accurately predict their performance and to make meaningful discriminations among the systems. In addition, more relevant applications will be added to the suite to more fully characterize the LANL workload.

Systematically embracing heterogeneity: In the past, the system acquisition process has required massive amounts of funding at one time in order to acquire the next big system in a single contract. In contrast, we propose to regularly (annually) acquire a fraction of a large system to make acquisitions more manageable. But this will introduce heterogeneity into the overall system among processor speeds, network technologies, IO subsystems, and even system software. Mastering this complexity will have great payoff, but will require systematically accounting for heterogeneity in everything that we do.

3. Programming Models Suitable for Human Beings

The current standard methods for programming high-performance applications are still mired in the explicit message-passing style. While currently portable, these methods are also incredibly complex and error-prone, leading to low programmer productivity.

Higher-level explicit constructs: Language constructs that enable programmers to explicitly control data distribution and motion at a much higher and hence simpler level of expression will be developed and deployed. The language constructs will be embedded in standard languages. This effort entails design, compilers and run-time software for implementation.

Domain-specific languages: Notations that enable programmer expression of algorithms in a much more domain-specific way are needed. A general framework will be developed that can capture information from annotated, application-specific libraries and produce high-level scripting languages. These can be used to efficiently write applications that run with high performance.

Portable libraries with high performance: Applications will continue to rely on high-quality libraries of high-performance components for scientific computing. The ATLAS project has shown that one-time extensive and expensive automated optimization of libraries for a specific architecture can result in greatly enhanced performance when the libraries are put into production. Techniques will be developed to perform these optimizations on larger bodies of code written at higher levels.

4. Advanced Architectures

Future systems of interest will be different from the clusters of SMPs that dominate high-performance computing today. Technologies such as multithreading, caches that can be managed by compiler-generated instructions in sophisticated ways, memory modules with active processing components, new interconnect topologies, multiple levels of parallelism, and streaming operations are alternatives available to future system designers. In a partnership that we will forge between DOE and a DARPA HPCS consortium, we will bring several technologies to bear.

Modeling of applications and architectures: Using LANL's model applications, expertise we develop for acquiring systems will be applied to the architectures being designed to expose problems in program performance at the earliest possible stages, when alternatives may be considered.

Programming models and languages: Another aspect of developing programming constructs and new approaches to languages is the ability to access advanced architectures at a lower level for explicit control (mostly for system and runtime software) and at a higher level (for effective programmability). Systems with new kinds of components will require programming approaches and compilers that can accommodate them explicitly or implicitly.

Reliability: These systems must be complex to achieve the performance required, and our work in reliability will be as directly applicable to these long-term architectural designs as to the medium-term work that we will also undertake.

Visualization: Current work at LANL in exploiting COTS technologies for high-performance scientific visualization has a natural place in an advanced architecture project. IO systems in general will exploit COTS to the hilt, and visualization subsystems based on our work will be a major contribution.

5. High-performance Communication Based on Commodity Protocols

The goal of this research is to investigate high-performance implementations of traditional and specialized communication protocols on commodity network fabrics. We define commodity networking in terms of the protocol layer implemented in the network fabric. In particular, we focus on IP networks: networks that route traffic using the Internet Protocol.

Traditionally, communication performance has been defined in terms of bandwidth and latency. While these metrics are still important, CPU overhead, the percentage of the host processor consumed in the implementation of a communication protocol, has recently emerged as another important metric. As network speeds have increased, so has CPU overhead. In many instances, the bandwidth that can be attained is limited by the speed of the processor.

In commodity systems, the interface point between the network fabric and computational node is a network interface card (NIC) on the PCI bus. Many vendors provide programmable NICs. In a preliminary study, we wrote a control program for the Alteon Acenic Gigabit Ethernet NIC. The default control program simply moves IP fragments between the host processor and the network. The altered control program implements IP fragmentation and reassembly on the NIC. We measured UDP delivery rates of 960 Mb/s for both control programs. However, while the CPU utilization for the default control program was about 80%, the CPU utilization for our control program was closer to 60%, a significant improvement. We plan to use the Myrinet interconnection network on the large LANL cluster for additional scaling studies.

We have two specific goals:

- Determine the performance benefits of moving the entire IP layer into a communication co-processor.
- Evaluate alternative implementations of the TCP sliding window and high-performance implementations of alternate communication protocols, including: VIA, Scheduled Transfer, and Portals 3.0.

6. Hierarchical Visualization of Large Data Sets and Visualization on PC clusters with Programmable Pipelines

Much of scientific high performance computing is focused on executing simulations that generate huge data sets whose size makes interpretation difficult. Scientific visualization supports improved interpretation through graphical methods. Specifically, we are working with an object-oriented particle system for both numerical simulations and visualization. We will add intelligence to the system so that it can be used to search out "interesting" areas in large data sets to keep the computational effort within predetermined bounds. These datasets are from the global climate simulations being run at LANL.

We have a commodity cluster of dual-processor Intellistations with Nvidia G Force 3 cards connected by Gigabit Ethernet. In this environment, network performance can be the limiter on visualization performance. We will characterize the separation of network performance and rendering performance. Techniques similar to those described in a previous section will be used to improve the overall speed of large-scale rendering.

Features of the G Force 3 cards, such as pixel shaders and a programmable pipeline, open up many new algorithmic possibilities for visualization. We will exploit these possibilities and benchmark new algorithms.

We have three specific goals:

- Implement Particle System used for feature extraction on ocean circulation data set.
- Characterize overall cluster visualization performance in terms of network performance and rendering performance.
- Benchmark algorithms using G Force 3 cards in a cluster environment.

7. Studies of Human Pixel Utilization Scaling for Scientific Visualization

Large-scale scientific visualization sometimes uses very large format tiled displays (LFTD), such as power walls, made up of a rectangular array of several modules (screen, projector, computer) positioned to create a large viewing area. Total pixel counts in excess of 60 million are achievable with this display technology, with roughly linear cost per pixel at the module level. Several physical considerations must be handled to make such a facility useful, which drives up costs and results in the necessity for them to be treated as shared resources.

It is not well understood when such massive displays are needed. This work will answer several specific questions:

- When do you need how many pixels to see at the least cost?
- How do users use these large displays and are they using the pixels efficiently?
- What are patterns of use as a function of display size and locale?
- Are the larger number of pixels actually required for all visualizations?
- Can any form of panning, zooming, and multi-resolution methods on smaller displays approximate actually walking around an LFTD?
- Can the results of other related human perception / comprehension studies be applied to this domain to optimize the use of visualization resources?

We will design and pilot a set of human-subject experiments to attempt to quantitatively answer some of these questions. We will identify a relevant human visualization task in collaboration with collaborators at LANL. Performance will be measured by a combination of time and accuracy. We will design and refine a set of specific experiments. Supporting software will be developed to drive the range of display technologies available to us: a six-module power wall, a single-module high end projection system, an IBM ultrahigh resolution desktop monitor, a conventional high resolution computer monitor, and a low resolution head tracked display. The experiments will be performed across as many of these display technologies as is possible.

We have three specific goals:

- Modify UNM visualization software to support compound/composite displays.
- Define and codify the experimental processes.
- Execute the experiment, perform preliminary analysis, and report findings.

Foundations of Computational Science

Mike Pernice (pernice@lanl.gov)
Dan Sorensen (sorensen@rice.edu)
Mike Fagan (mfagan@rice.edu)
Lennart Johnsson (johnsson@cs.uh.edu)
Deepak Kapur (kapur@cs.unm.edu)
Doug Kothe (dbk@lanl.gov)

Based on a contemporary assessment of LANL's needs, the *Foundations of Computational Science* effort encompasses two main aspects of computational mathematics:

1. Mathematical Methods and Algorithms
2. Verification (and validation) of Simulation Codes

Generally, the team works on the development, analysis, and implementation of numerical methods that address leading problems of science and engineering. In addition, the team is interested in developing computational metrics and invariants that can confirm important properties of current and future simulation codes. We also have a keen interest in software design and practice suitable for large-scale simulation codes, and in automating the process of tuning those codes for efficiency on specific platforms. These interests are very much in line with the current and future needs of ASCI.

The capabilities of multi physics applications have improved significantly over the last thirty years. Much of this improvement is due to highly visible advances in computing hardware, whose capacity has increased by more than five orders of magnitude over this period of time. However, a similar contribution is due to advances in numerical algorithms and their realization in efficient software implementations. Because the ASCI program places an increased emphasis on predictive capability, there has been an ever-increasing demand for the ability to evaluate the correctness of simulations through the use of techniques such as uncertainty quantification, and verification and validation. Automating this evaluation process and retrofitting it to existing application codes motivates exploration of additional classes of numerical algorithms not traditionally associated with multi physics applications.

Many simulation codes for the ASCI effort employ semi-Lagrangian methods in which the computational grid evolves to track solution features. This technique often leads to strongly distorted meshes, which creates difficulties in designing accurate discretizations. The solution of the resulting systems of linear equations is often the principal bottleneck in many ASCI codes. Consequently, research and development of efficient parallel numerical methods for diffusion equations in highly heterogeneous anisotropic media continues to be critical to the overall success of the stockpile simulation program.

Users of complex, large-scale computer models often need to assess the sensitivity of model output to changes in model input. Such sensitivity calculations lie at the heart of methods for uncertainty quantification. Code-based sensitivity analysis combines compiler-based source code analysis and transformation with modern numerical algorithms to augment model output with sensitivities. These techniques have been successfully used on models written in Fortran 77 and C. This sub-project focuses on the extension of these techniques to models written in Fortran 90.

There is a growing interest in the use of large-scale simulations for design, parameter identification, or uncertainty analyses of multi physics applications. Optimization-based approaches can be used to effectively explore parameter space, but must account for the fact that most complex multi physics codes are originally designed for simulations only, allow little intrusion by the optimizer, and in most cases are only available as black-boxes. Techniques that have been successfully applied to industrial multidisciplinary design optimization can be applied in these circumstances, but effort is needed to extend the capabilities of these methods to handle larger problems on the order of 100-200 design variables.

Scalable methods for eigenvalue problems are necessary ingredients for practical large-scale stability and sensitivity analysis. Eigenanalysis performance can be improved by orders of magnitude through a class of techniques for introducing preconditioning into eigenvalue calculations, known as approximate shift-and-invert schemes. This effort leverages previous research efforts into the implicitly restarted Arnoldi method, and its parallel implementation in P_ARPACK, which has been successfully used in a number of large-scale applications.

Project Descriptions

Parallel Numerical Methods for the Diffusion and Maxwell Equations in Heterogeneous Media on Strongly Distorted Meshes

UH: Y. Kuznetsov, R. Glowinski, L. Johnsson
LANL: M. Shashkov, M. Berndt, D. Moulton

Efficient parallel numerical methods for the diffusion and Maxwell equations in highly heterogeneous anisotropic media is an important topic for scientists and engineers working in computer simulation of complex physical phenomena. This statement is very relevant to several research groups at LANL and UH, for instance, to the T7 group at LANL. The researchers from the LANL part of the project are experienced in accurate and physically consistent approximations to the diffusion and Maxwell equations on strongly distorted meshes as well as in applications of advanced numerical methods to real-life scientific and engineering problems.

The researchers from UH have long-term experience in discretization of partial differential equations by mixed and hybrid finite element methods. They also hold the worldwide leading positions in designing of efficient parallel iterative solvers based on a combination of domain decomposition, fictitious domain, and multilevel techniques.

Short term

The first main objective of the project is to develop and investigate new accurate, physically consistent, and convenient methods for applications approximations to the 3D diffusion and Maxwell equations with heterogeneous anisotropic coefficients on strongly distorted, logically rectangular meshes.

Long term

The second main objective of the project is to design, investigate, and implement on parallel computers new fast iterative solvers for large-scale algebraic systems resulting from mesh discretizations. The expected size of mesh problems is 10**7 - 10**8 degrees of freedom.

Methods and Tools for the Solution of Non-smooth, Multi-scale, Coupled Models

Rice: P. Kloucek, F. Toffoletto

LANL: J. Brackbill

We will concentrate on three areas of research in which we have ongoing collaboration with two groups in LANL. The first area is focused on developing coupling equations that can connect stochastic and deterministic models. Namely, we propose a form of Fokker-Planck equation that can connect any form of Landau-Vlasov theory with field models. We have completed a one-dimensional model, which we will be testing in collaboration with J. Brackbill. Upon successful implementation, we will extend our theory to higher dimensions. Large-scale application of our approach will include materials science as well as magnetospheric multiscale physics applications. The second area is focused on the development of a massively parallel implementation of non-relativistic Landau-Vlasov-Maxwell kinetic equations. These equations represent one of the outstanding challenges for applied mathematics due to their complexity. The third area is focused on the collaboration with the CartaBlanca team. We have developed a theoretical basis for active control of shape memory materials. We will implement the model in the CartaBlanca environment to provide a materials science component of the package. We strive to derive an equivalent of Landau-Vlasov theory for the crystalline and structured materials. Such theory should be useful for computational evaluation and design of complicated composite materials, stochastic foams, and similar nano-to-micro-scale based materials. We also plan to implement these equations in the CartaBlanca environment.

Analysis and Optimization of Linked Subsystems

J. Dennis, M. Heinkenschloss

We are working on surrogate management frameworks and simultaneous analysis and design approaches for design, parameter identification, or uncertainty analyses governed by complex simulations.

One aspect of our research targets problems that involve complex multi physics codes that are originally designed for simulations only, allow little intrusion by the optimizer, and, in most cases, are only available as black boxes. Frequently, traditional optimization approaches cannot be applied to these problems. In meetings with the T7 group at LANL, these were identified as the types of optimization problems that they encounter most often. Our surrogate management framework has been quite successful in solving several such complex design problems. Design parameters can be continuous or categorical. A categorical variable is a variable that must always be a member of a finite set, or else the simulations on which the design is to be based cannot be run. For example, a categorical variable could represent the choice from a set of materials for a given insulation sector. We have even solved problems where the categorical variable determines its dimension. Software is available through the LACSI web site and a version of our surrogate management framework is implemented in the Boeing Design Explorer, used routinely in industrial MDO problems at Boeing.

Users are clamoring for several enhancements, most notably, that we be able to apply our techniques to larger problems on the order of 100-200 variables rather than the 20 or so variables we are handling for them now. The major obstacle here is the difficulty in building such high dimensional surrogates. Another extremely important question is how to handle mixtures of continuous and categorical design variables in black-box optimization problems. Traditional branch and bound techniques cannot even be applied to these problems. We have a successful algorithm developed with LACSI support, but we need to study how to handle larger numbers of variables.

Another aspect of research is the development of simultaneous analysis and design (SAND) tools for optimization problems with a large number of variables. These tools complement our surrogate management frameworks. They have the potential to solve the optimization problem in a time that is only a small multiple of the time needed for a single simulation, and they can handle very large numbers of continuous variables. However, they are tightly integrated with the simulation. Such methods have been used, e.g., for optimal design and control application in fluid flow. We have approached the T3 group to explore possible collaborations. Despite the success of SAND methods, several important questions remain. These include the integration of parallelism into the optimization, extension of our SAND approaches to better handle inexact problem information, improvements to handling a very large number of inequality constraints, and use of model hierarchies to enhance the convergence properties.

Short term

- Evaluate domain decomposition approaches to integrate parallelism into SAND optimization as well as using it for subproblem solves.
- Improve use of derivative information to direct search methods.

Long term

- Extend surrogate management approaches to black-box MDO to handle larger numbers of variables, especially mixtures of continuous and categorical variables.

The major obstacle here is the difficulty in building such high dimensional surrogates. We plan to investigate decompositional and compositional approaches to overcome this obstacle. An approach that seems promising for the effective handling of larger numbers of categorical variables is to use directional derivative information.

- Extend SAND optimization methods to better handle inexact problem information and allow the use of model hierarchies to enhance the convergence properties.

Methods and Software for Inverse and Control Problems

W.W. Symes, L. Borcea

Component frameworks for simulation driven optimization (W. W. Symes)

This is a special case of the general component framework design problem, and does not have nearly the complexity implications of the applications contemplated by CCA, for example. However it has considerable scope and I believe that our solution will be useful to many other groups. The framework must couple a control process, typically an optimization algorithm, with a simulation process, typically a parallelized PDE solver, running in two or more different software environments. The construction of such a framework is a good deal easier if the interfaces defined by the control and simulation processes are limited in type. The latest version of the Hilbert Class Library provides two class families, DataContainers and FunctionObjects, which completely encompass all interacting with the communications layer in the component framework. With several graduate students, I am working on making as transparent as possible the interaction of these two types with a simple communications layer built out of standard Unix sockets. The aim is to render the transition between serial and client-server applications as simple as possible.

Simulation driven optimization (W. W. Symes)

Simulation driven optimization poses a notorious problem: discretization and linearization do not commute when adaptive gridding is part of the simulation package. This leads to the "optimize then discretize" vs. "discretize then optimize" debate. We propose that neither approach is correct, and that another approach, based on locally defined smooth problems, is actually the only mathematically consistent way to combine adaptivity and Newton-type optimization. Eric Dussaud, my current LACSI-supported PhD student, has taken this as his thesis problem. The implications for code design interact with the component considerations just mentioned.

We hope to foster interaction with other activities at LANL involved with similar work (especially on component framework applications). Ultimately we hope to accommodate other applications of client-server and peer-peer component architecture (other than simulation-driven optimization).

Imaging and time reversal in random media (L. Borcea)

We are interested in the inverse scattering problem of target identification in a random medium, in a regime with significant multipathing. Specifically, we seek to develop

imaging algorithms that are statistically stable, such that the resulting images are reliable and independent of the realizations of the random medium. We have already been successful in imaging small scatterers buried in infinite random media, via active arrays of transducers (antennas). Our algorithms combine state of the art signal processing techniques with a careful statistical analysis of time reversal in random media and they are proven theoretically and numerically to be statistically stable. We wish to extend our work to finite size targets and to media with interfaces (such as the sea surface or the earth surface). This work will consist of both analysis and extensive numerical calculations, especially for the case of data gathered with very large, synthetic aperture arrays of antennas.

Numerical Linear Algebra for Large Systems
(Eigenvalue Methods and Software for ASCI-MPP Systems)

Rice: D.C. Sorensen, M. Embree, H. Thornquist

LANL: B. Nadiga

We are working on Krylov and Newton-Krylov techniques for time integration and linear stability analysis of an ocean circulation model (OCM) developed by Nadiga. One goal of the project is to develop improved Matrix-free Newton-Krylov methods for solving these large-scale systems of nonlinear equations arising in the time integration of the dynamical system. We are also concerned with steady state linear stability analysis on an idealized reduced-gravity quasigeostrophic ocean model. Part of this success has been the detection of Rossby waves, which can be indicators for the onset of chaotic behavior in the dynamical system.

Short term

- Stability and sensitivity analysis of certain steady states of the OCM.
- Determination of Rossby waves, which are fundamental to the understanding of very long term cycles in ocean circulation.

Considerable algorithmic work is needed to accomplish this. We expect to improve eigenanalysis performance by orders of magnitude with an approximate shift invert scheme, a technique for introducing pre-conditioning in eigenvalue calculations. This scheme has been implemented and tested on the OCM and has successfully computed the desired Rossby waves and corresponding eigenvalues much faster than any competing schemes. We observe linear scaling with respect to the problem size (i.e. mesh independence for the required number of iterations).

To accelerate convergence for these problems, Embree and Sorensen are studying adaptive pre-conditioners based upon partial eigensystem information and effective restarting and deflation methods.

Long term

The efficiency of a Newton-Krylov method is generally dependent upon the quality of the linear system preconditioner. The best preconditioners incorporate problem-specific information. We believe the information obtained from the linear stability analysis can be used to build a better Newton-Krylov preconditioner.

We hope to use eigenanalysis to design improved linear system preconditioners that are adaptive. These preconditioners are to be integrated into the design of a Newton-Krylov solver for dynamics calculations in the OCM.

We plan to utilize a component framework for software integration within this project as a prototype for future projects that might require steady state calculations, stability, and bifurcation analysis. We intend to use the Python scripting language to implement our component framework. However, prior to a final decision, we shall also consider other options. Many of the components that we need are already available. We expect that integration of these components with Python will result in a framework that can be easily modified to accommodate improved components and future advances in the OCM.

Code-Based Sensitivity Analysis

Rice: M. Fagan

LANL: R. Henninger

Short term

The short-term goals for the code-based sensitivity effort are directly related to the ongoing collaboration with the Telluride Project directed by Doug Kothe (technical point-of-contact is Rudy Henninger). There are two major short-term goals:

1. Assist in the application of Adifor to current Fortran 77 codes.
2. Extend the Adifor technology to Fortran 90, so that accurate sensitivity calculations can be easily generated for the Truchas code.

Long term

Longer-term goals for the code-based sensitivity project go in four different directions:

1. Applying automatic differentiation for verification of ODE-based and PDE-based computer models.
2. Applying automatic differentiation in the construction of Newton-Krylov solvers.
3. Extending Adifor-style automatic differentiation to additional programming languages, notably Fortran 90, Java, and C/C++. The CartaBlanca project, in particular, has expressed interest in Java.
4. Extending the augmentation paradigm to include other sensitivity measures such as intervals or probability distributions.

Large-Scale Nonlinear Optimization Algorithms and Software

R. Tapia, Y. Zhang

The objective of this project is the development of algorithms and software for certain large-scale nonlinear optimization problems. One primary research area will be in linear and nonlinear semidefinite programming, i.e., optimization problems with matrix-valued variables and/or constraints. Another emphasis will be on the development of novel algorithms for large-scale graph-partitioning problems based on nonlinear optimization.

This investigation will support computational infrastructure building for emerging applications. It will provide reliable and enabling tools for important computational tasks that need to be solved in LANL's applications and predictability studies. In particular, effective graph-partitioning algorithms will have important applications and immediate impact in parallel computations.

Application Performance

Adolfy Hoisie (hoisie@lanl.gov)
Dan Reed (reed@cs.uiuc.edu)
John Mellor-Crummey (johnmc@cs.rice.edu)
Rob Fowler (rjf@cs.rice.edu)

Building scientific applications that can effectively exploit extreme-scale parallel systems has proven to be incredibly difficult. The sheer level of parallelism in such systems poses a formidable challenge to achieving scalable performance. In addition, the architectural complexity of extreme-scale systems makes it hard to write programs that can fully exploit the capabilities of these systems. In today's extreme-scale systems, complex processors, deep memory hierarchies and heterogeneous interconnects require careful scheduling of an application's operations, data accesses and communication to enable the application to achieve a significant fraction of a system's potential performance. Furthermore, the large number of components in extreme-scale parallel systems makes component failure inevitable; therefore, long-running applications must be resilient to hardware faults or risk being unable to run to completion.

The principal goals of the application performance research thrust are

- understanding application performance on present-day extreme-scale architectures through the development and application of technologies for measurement and modeling of program behavior,
- devising software strategies to ameliorate application performance bottlenecks on today's architectures, modeling the behavior of applications to understand factors affecting their scalability on future generations of extreme-scale systems, and
- investigating software technology that will enable higher performance on next-generation, extreme-scale parallel systems.

A broad spectrum of issues affects application performance. A multitude of challenging problems must be solved to understand how to best implement scientific applications so that they can achieve scalable high performance on extreme-scale parallel systems. As part of this research thrust, the project team will explore the topic of application performance on many fronts and undertake a program of research that aims to develop technologies to support measuring, modeling, understanding, tuning, and steering application performance on current and future generations of extreme-scale parallel architectures. This work will address all aspects of performance and reliability spanning system architecture, network, and applications. Our investigation will include work on both scalability and node performance.

The findings from this research, as well as tools and software infrastructure developed as products of this effort, are expected to benefit all ASCI application teams by providing them with more efficient programming models, technology for compiler-assisted tuning of applications, better performance instrumentation and diagnostic capabilities, improved algorithm-architecture mapping, and better performing extreme-scale parallel architectures.

Short- term Goals

A principal short-term goal of this effort is to understand the key performance issues for scientific applications characteristic of ASCI workloads. The application performance team will initiate a study of application performance that focuses on software infrastructure and architectures of interest to ASCI. A team, including LANL application experts, LANL computer scientists, and academic partners, should collaborate in this study. The study should include, but not be limited to, codes characteristic of the Crestone and Telluride projects. For the set of applications chosen, the study should aim to quantify the key factors limiting scalable performance. In particular, the study should try to understand the cost to applications from load imbalance, serialization, underutilization of processor functional units, data copying, poor temporal and spatial locality of data accesses, exposed communication latency, high communication frequency, and large communication bandwidth requirements. This quantitative assessment of factors limiting application performance on current-generation architectures will help to focus long-term research on software and hardware technologies that hold the most promise for improving application performance and scalability on future systems.

A second goal is to assess the impact of component failure on ASCI applications. As the scale of ASCI systems increases, hardware reliability is becoming increasingly problematic. To assess the impact of component failure on ASCI applications, the academic partners hope to study the system-wide performance and reliability of extreme-scale systems by examining traces from Blue Mountain and other large-scale systems. A precondition for this work is for system traces from Blue Mountain to be made available to academic partners. Acquiring an understanding of what component failure modes are common will be useful for designing strategies for coping with them.

To foster greater interaction among LANL application teams, LANL computer science teams, and academic computer scientists, we plan to organize a tools workshop at LANL. This workshop will have two academic purposes. First, it will introduce tools that have been developed by the performance team to the application teams with the aim of engaging application scientists in using these tools to understand how their choices of data structures and algorithms affect performance. Second, it will provide an opportunity for cross-disciplinary working groups that are undertaking detailed analysis of ASCI workload exemplars to meet and exchange ideas and results.

The Rice team will continue with refinement of the HPCView performance analysis toolkit to facilitate analysis of node performance on principal ASCI platforms. Ongoing work includes (1) enhancement and hardening of a Java-based performance browser that supports detailed analysis of large programs by dynamic generation of views and (2) extending binary analysis capabilities to support current ASCI platforms including Alpha, Power and Pentium, along with IA-64, which is a likely building block for future ASCI systems. The Illinois team will (1) explore failure modes of large-scale systems from workload traces, (2) develop statistical characterization techniques to quantify system behavior more inexpensively and (3) extend the notion of performance contracts for performance specification and validation.

The LANL team will continue with on-going analysis and modeling of applications in the ASCI workload. To date, they have developed models that characterize the communication and parallel scaling of Sn transport on a structured grid and hydrodynamics; the team is now tackling Sn transport on unstructured grids. The Rice team is working to develop software tools to synthesize parameterizable performance models of node programs. A goal is to combine these modeling efforts to yield models reflecting both communication and computation performance.

Long-term Goals

Better tools for measurement and analysis of application performance: Current tools for measurement and analysis of application performance on extreme-scale systems suffer from numerous shortcomings. Typically, they provide a myopic view of performance; they provide only descriptive rather than prescriptive information; and they fail to present data effectively for extreme-scale systems. The scale of these systems implies that a large volume of performance data must be collected and digested. Improved strategies are needed for data collection, both to reduce the volume of information that must be collected and to collect more useful information. To help users cope with the overwhelming volume of information about application behavior on extreme-scale systems, more sophisticated analysis strategies are needed for automatically identifying and isolating key phenomena of interest, distilling and presenting application performance data in ways that provide insight into performance bottlenecks, and providing application developers with guidance about where and how their programs can be improved.

Scaling the Memory Wall: As Moore's law has predicted, processor speeds have increased by roughly 60% per year over the last several decades. Over that time, memory speeds have increased by about only 7% per year. As a result, there is a large mismatch between processor and memory speeds in today's computer systems: memory accesses commonly take 100 processor cycles. Most modern computer systems use layers of caches to bridge this gap, but typically bandwidth between memory and the lowest layer of cache in the memory hierarchy is insufficient to deliver enough data to computations that do not achieve substantial temporal reuse in cache and registers. Unfortunately, most scientific computations that manipulate large data sets often realize little temporal reuse. Consequently, typical scientific computations are starved for memory bandwidth in systems based on modern microprocessors. A multi-pronged approach is needed to address this problem.

Radical changes in hardware and software design offer the most potential for overcoming the memory bottleneck. Architectural alternatives being actively explored include processor-in-memory (PIM) designs and streaming architectures. To influence the development of these systems, design alternatives must be evaluated with ASCI workloads and feedback must be provided to computer architects about the findings. Efficient software for such radically different, next-generation computer systems will have very different organizing principles at the machine level. Research is needed to improve programming models so that applications can be expressed more naturally, yet

mapped onto current and next-generation architectures more efficiently. For high performance on present day architectures and future systems, better compiler technology is needed for transforming applications from an organization appropriate for humans to one that is well matched to exploiting an architecture's capabilities. In particular, typical applications fail to adequately exploit temporal reuse of data at all levels of the memory hierarchy. Compiler technology for reorganizing programs to exploit latent opportunities for temporal reuse can substantially improve performance. However, compiler technology is not omniscient. Choice of application data structures can aid or thwart compiler transformation capabilities. Further investigation is needed to understand the impact that algorithms and data structures have on the compiler's ability to transform programs for high performance and to determine what kinds of data structure representations are appropriate for organizing ASCI applications so that they can be mapped efficiently to present-day and future computer systems.

Automatic application tuning: The architectural complexity of modern computer systems makes it very difficult to manually tune codes to achieve top performance. Furthermore, the rapidly changing landscape of computer platforms means that any investment in manual tuning for a particular platform is likely to soon be obsolete. A promising approach that has emerged for automatically tuning library codes is embodied by the ATLAS and UHFFT projects. At library generation time, these systems generate a multitude of variants of a pre-determined set of library primitives, run a collection of experiments to empirically evaluate the performance of each variant on the target platform, and then select the variants with the best performance for inclusion in the run-time library. This search-based tuning strategy virtually eliminates the manual effort of tuning for particular target platforms and generates highly efficient code. The tuning strategy can be re-executed when a version is needed for a new platform. Research is needed to devise search algorithms that make it practical to apply this style of automatic empirically driven tuning to whole applications. Previous work on self-tuning libraries required library developers to write a code generator to enumerate each interesting variant of a library procedure for empirical evaluation. Generalizing this automatic tuning approach to whole programs requires developing algorithms for identifying critical loop nests that could potentially benefit from tuning. Exhaustive search through all combinations of variants for each loop nest in a program is impractical. The challenge is to develop intelligent search algorithms that use program semantic information to guide the selection of promising implementation variants that merit empirical evaluation. Moreover, these techniques must include alternatives and solutions that can respond to hardware and software faults present in extreme scale systems.

Compiler support for scalable parallel programming: Today, MPI is the model of choice for writing scalable parallel programs. However, as a programming model, MPI has several shortcomings. Explicitly coding communication using MPI primitives is tedious and error prone. Also, writing MPI programs to achieve top performance increases programming complexity: application developers must choreograph asynchronous communication and overlap it with computation. Furthermore, because of the explicit nature of MPI communication, significant compiler optimization of communication is impractical. Programming abstractions in which communication is not expressed in such

a low-level form are better suited to having compiler optimization play a significant role in improving parallel performance. SPMD programming models such as Co-array Fortran (CAF) and Unified Parallel C (UPC) offer promising near-term alternatives to MPI. Programming in these languages is simpler: one simply reads and writes shared variables. With communication and synchronization as part of the language, these languages are more amenable to compiler-directed communication optimization. This offers the potential for having compilers assist effectively in the development of high performance programs. Research into compiler optimizations for SPMD programming languages offers the potential of not only simplifying parallel programming, but also yielding superior performance because compilers are suited for performing pervasive optimizations that application programmers would not consider employing manually because of their complexity. Also, because CAF and UPC are based on a shared-memory programming paradigm, they naturally lead to implementations that avoid copies where possible; this is important because on modern computer systems, copies are costly. Beyond explicitly parallel SPMD programming models, implicitly parallel programming models such as High Performance Fortran offer an even simpler programming paradigm, but require more sophisticated compilation techniques to yield high performance. Research into compiler technology to increase the performance and scalability of data-parallel programming languages as well as broaden their applicability is important if parallel programs are to be significantly simpler to write in the future.

Compiler technology for exploiting modern processors: Keeping pace with the Moore's law curve and delivering 60% annual increases in processor performance has come at the expense of increasing complexity of processor architectures. Exploiting modern processor architectures to achieve a significant fraction of peak performance with code compiled from conventional programming languages such as Fortran, C, and C++ requires compiler innovation for each new architectural feature. For example, the IA-64 introduces a set of new architectural features that pose a considerable set of challenges for compilers. First, the functional units must be kept busy. This requires the compiler to transform the input program so that it has enough instruction-level parallelism (ILP) to sustain the computation rate. It requires instruction-scheduling techniques that can convert available ILP into dense schedules - for simple loops, for loops with control flow, and for straight-line code. Second, operands must be ready for each instruction. This will involve transforming programs to match their locality to the memory hierarchy of the target system, including real applications of blocking, prefetching, and (perhaps) streaming. Once data is on-chip, a compiler may need to manage instruction and data placement with respect to the clustered register file, along with the classic problems of allocation and scheduling. Third, predication must be handled with a holistic approach. If-conversion is not the whole answer. Open issues include understanding the tradeoff between underutilization of instruction issue slots with predication versus branching to out-of-band denser (unpredicated) code, predicate register management, the interaction between predicate lifetimes and instruction placement in the scheduler, and minimizing the impact of predicate evaluation on overall application performance.

Compiler and tool support for fault-tolerant programming: Component failure is already a problem in current extreme-scale computing platforms. Today, programmers typically write programs to survive component failure by explicitly augmenting them with code to write periodic checkpoints and restarting from the latest checkpoint following component failure. Although fault-tolerance techniques have been studied for decades, their use in programs is neither easy nor common. For data-intensive programs, one promising approach is to store persistent data on disks and use dynamic assignment of large-grain tasks to collections of processors, coupled with transaction-like recording of computation results. Compiler support to simplify the data choreography in such a programming model would help ease the transition from conventional programming styles employed today to fault-tolerant models for high performance programming. Alternative techniques based on enhanced hardware support and smoothly degradable algorithms are also potentially applicable.

Technology for model-driven performance steering: Today, most applications simply run to completion on the resources that they acquire at program launch. Performance steering offers an opportunity to adjust a running program for more efficient execution and to adapt to changing resource availability (e.g., due to component failures or resource sharing) Research into strategies for automatic performance steering based on performance and fault models incorporated in application programs offers the potential to enable long-running programs to repeatedly adjust themselves to changes in the execution environment – perhaps to opportunistically acquire more resources as they become available, to rebalance load or adapt to component failures. Validated performance "contracts" among applications, systems, and users that combine temporal and behavioral reasoning from performance predictions, previous executions, and compile-time analyses are one promising approach.

Computer Science Community Interaction

John Thorp (thorp@lanl.gov)
Rob Fowler (rjf@rice.edu)
Ken Kennedy (ken@cs.lanl.gov)
Linda Torczon (linda@rice.edu)

LACSI is a collaborative research effort between Los Alamos National Laboratory, Rice University, the University of Houston, the University of Illinois at Urbana-Champaign, the University of New Mexico, and the University of Tennessee at Knoxville. Effective means of supporting collaborations are important to the success of LACSI. To support collaboration, LACSI will provide a variety of opportunities for researchers from Los Alamos and the academic partner sites to visit each other, to share ideas, and to actively collaborate on technical projects.

In addition, we will organize, host, and otherwise support a series of technical workshops on topics related to the LACSI technical vision. This will include a series of workshops at LANL targeted at exposing application researchers to emerging technologies.

LACSI will also host an annual symposium to showcase LACSI results and to provide a forum for presenting outstanding research results from the national community in areas overlapping the LACSI technical vision. This will be a traditional conference-style meeting with participation by both LACSI members and scientists from the community at large.

We will also coordinate a technical infrastructure between Los Alamos and the academic partners, enabling web broadcasting of local technical talks, workshops, and the Annual Symposium to an off-site audience.