Contract No: TBD

Provide Medium to Long-term Computer Science Research Relevant to the Goals of the Advanced Simulation and Computing (ASC) Program

LACSI Task TBD

Los Alamos Computer Science Institute External Computer Science Research

FY05 Statement of Work

September 24, 2004

- **Components**

This effort will incrementally develop two long-term goals.

Develop *frameworks for integrating existing components* rapidly and conveniently into complete applications. These frameworks must be able to produce efficient applications from scripts within reasonable compile times. In addition, they must be able to integrate components written in different languages, particularly Fortran and object-oriented languages like C++ and Java. Finally, the frameworks must support the generation of applications that execute with reasonable and reliable efficiency in a distributed computing environment.

Develop a *collection of components for use in science and engineering applications*. The algorithms should be general, portable, and usable in a variety of situations.

Specific tasks and deliverables are described in Appendix A, Section 1.1.

- **Systems**

The systems activity focuses on research and advanced development of computer subsystems, both hardware and software, of strategic interest to present and future ASC architectures. The specific tasks and deliverables are described in Appendix A, Section 1.2.

- **Computational Science**

Provide research and development in the areas of numerical methods for partial differential equations, linear and nonlinear solvers, and verification and validation methodologies to address the following goals:

Develop, analyze and implement numerical methods for coupled multi-physics. Particularly important is the discretization of partial differential equations by mixed and hybrid finite element type methods.

Develop linear and nonlinear solvers algorithms.

Develop software tools suitable for large-scale simulation codes and automation of the process of tuning those codes for efficiency on specific platforms.

The specific tasks and deliverables are described in Appendix A, Section 1.3.

- **Application and System Performance**

Develop compiler and run-time technology that will help application developers achieve a high fraction of peak performance on large-scale parallel computing systems. The specific tasks and deliverables are described in Appendix A, Section 1.4.

- **Computer Science Community Interaction**

Foster collaborative relationships between LACSI participants at LANL and at the LACSI academic sites using activities including:

- Host visitors and speakers at academic sites to encourage collaborative relationships between researchers at LANL and at the participating academic institutions.
- Support scientific collaboration visits to Los Alamos National Lab to integrate research products into operations.
- In conjunction with LANL, organize, host, and otherwise support a series of technical workshops on topics related to the LACSI technical vision.
- 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.

Specific tasks and deliverables are described in Appendix A, Section 1.5.

Project Management

The implementation strategy and tasks associated with the academic LACSI projects are detailed in Appendix A. Documentation regarding the objectives of the Institute may be found in the latest LACSI "Priorities and Strategies" document.

Deliverables

- Quarterly status reports on milestones/deliverables

Other reports and briefings as required.

Selected Milestones

Rice University

- Deliver a report on an experimental study that demonstrates the potential of automatic code and data reorganization strategies, including object inlining, on Marmot prototype code.
- Release a version of the Rice HPCToolkit performance software that is integrated with Clustermatic for Opteron clusters. This will include options to use oprofile and PAPI/perfctr for node-wide and application-specific data sources, respectively.
- Augment a serial (i.e. non-parallel) subset of the Truchas code by the Adifor90 automatic differentiation tool to perform forward mode sensitivity computations for two selected scalar inputs.
- Deliver a report exploring the impact of adaptive compilation and autotuning on selected ASC codes.
- Produce implementation of Co-array Fortran for Clustermatic clusters. Demonstrate functionality with CAF versions of LANL's Parallel Ocean Program and ASCI Sweep3D applications.

University of Houston

- In collaboration with LANL staff, review a few multigrid codes chosen by LANL staff, design an adaptive software package for the multigrid components of at least one of the selected codes, and develop a first implementation for assessment of the merit of the adaptive approach. Deliver report on preliminary findings.
- Develop, investigate, and evaluate on test problems relevant to ASC applications efficient parallel algebraic preconditioners/solvers for polyhedral-mesh discretizations of 3D diffusion equations.

University of New Mexico

- Using the Open MPI framework, provide an empirical study of the performance implications associated with handling errors at different levels in the protocol stack.

University of North Carolina

- Develop tools and techniques for failure indicator monitoring and adaptation that support the Clustermatic cluster infrastructure.

University of Tennessee, Knoxville

- Design a heterogeneity interface for Open MPI.

Appendix A

Los Alamos Computer Science Institute

Statement of Work for Academic Participants

The *Los Alamos Computer Science Institute (LACSI)* was created to foster internationally recognized computer science and computational science research efforts relevant to the goals of Los Alamos National Laboratory (LANL). LACSI is a collaborative effort between LANL and the Rice University Center for Research on High Performance Software (HiPerSoft), along with its partner institutions: The University of Houston (UH), the University of New Mexico (UNM), the University of North Carolina at Chapel Hill (UNC), and the University of Tennessee at Knoxville (UTK).

LACSI was founded with the following goals:

- To build a presence in computer science research at LANL commensurate with the strength of the physics community at LANL,
- To achieve a level of prestige in the computer science community on a par with the best computer science departments in the nation,
- To pursue computer science research relevant to the goals of High Performance Computing (HPC) programs at LANL, and
- To ensure that there remains a strong focus on high-performance computing in the academic computer science community.

In keeping with these goals, LACSI researchers engage in joint high-performance scalable computing research and in collaborative activities that foster a strong relationship between LANL and the participating academic institutions.

In the following section, we present the vision, implementation strategy, and tasks associated with LACSI projects at Rice University and its partner academic institutions. In the final section, we describe the management and administrative plan for the LACSI academic partners.

1. Strategic Thrusts

In March 2002, the LACSI Executive Committee (EC) met with LACSI researchers at LANL to discuss methods of addressing issues raised in the 2001 LACSI contract review. The body was tasked to develop priorities and strategies to meet LANL's future programmatic and computer science needs. The group developed a framework 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. A description of the long-term vision, framework, and objectives developed at the meeting is available in a document (LAUR #02-6613) titled *Priorities and Strategies.*

In both April 2003, the LACSI EC met with senior LANL personnel to revise the framework, priorities, and strategies established at the planning meeting in 2002 and *Priorities and Strategies* was revised to incorporate the results of the April 2003 planning meeting (LAUR # 03-7355). In February 2004, the EC again met with senior LANL personnel to revise the framework, priorities, and strategies established in previous planning meetings. *Priorities and Strategies* is being revised to reflect the results of the February 2004 planning meeting. The current framework outlines five strategic thrust areas:

- Components
- Systems
- Computational Science
- Application and System Performance
- Computer Science Community Interaction

The following five subsections describe the vision, implementation strategy, and tasks associated with the academic LACSI projects that fall under the five strategic thrust areas.

1.1. Components: Component Architectures for Rapid Application Development and Composition in a Networked Environment

Investigators: Ken Kennedy, Zoran Budimlic, Keith Cooper, Jack Dongarra, Rob Fowler, Guohua Jin, Lennart Johnsson, Charles Koelbel, John Mellor-Crummey, Dan Reed

LANL Collaborator: Craig Rasmussen

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.

Through modularization and the existence of well-defined component boundaries (specified by programming interfaces), components allow scientists and software developers to focus on their own areas of expertise. For example, components and modern scripting languages enable 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, making it possible for 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 scientific application developers, particularly those at Los Alamos National Laboratory, cannot afford to sacrifice significant amounts of performance for this clearly useful functionality. Therefore, a central goal of the effort is to explore 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.

The overarching goal of this activity is to develop component architectures and component libraries that can be used to support rapid prototyping of portable parallel and distributed applications and rapid reconfiguration of existing applications. These architectures would be the basis for frameworks for applying advanced compilation techniques, run-time system elements, and programming tools to prepare applications for execution on scalable parallel computer systems and distributed heterogeneous grids.

To succeed, this effort will need to accomplish two long-term goals.

1. It must explore frameworks for integrating existing components rapidly and conveniently into complete applications. These frameworks must be able to produce efficient applications from scripts within reasonable compile times. In addition, they must be able to integrate components written in different languages, particularly Fortran and object-oriented languages like C++ and Java. Finally, the frameworks must support the generation of applications that execute with reasonable and reliable efficiency in a distributed computing environment.
2. It must explore the design and implementation component libraries for use in science and engineering applications. Such libraries should be ideally suited for use in the rapid prototyping frameworks developed as part of this activity. The algorithms must be general, portable, and usable in a variety of situations. In addition, components themselves should be auto-tunable for high performance on new computing platforms.

1.1.1. LACSI Component Integration Challenge

Investigators: Ken Kennedy, Zoran Budimlic, Keith Cooper, Jack Dongarra, Rob Fowler, Guohua Jin, Lennart Johnsson, John Mellor-Crummey, Dan Reed

One of the most difficult challenges for component integration is the problem of integrating data structure components (e.g., sparse matrices) with functional components (e.g., linear algebra). This problem is hard because the frequency of invocation of data access methods places a premium on high performance of the component interfaces. The long term-research section of the proposal has taken this as a major focus for the next several years.

To drive this research in directions that are most useful to LANL, we will collaborate with developers on the Marmot code teams to understand how component integration strategies can make their efforts more effective overall. In particular, we will work to define a challenge problem by specifying the interfaces and functionality of components within Marmot that implement abstract meshes on which computations are carried out. These specifications will be developed through a joint study between code developers and computer and computational scientists within LACSI. A goal of this effort is to leverage the telescoping languages strategy for

efficient component integration that is the subject of LACSI research. The ultimate goal is to make it possible for the designer to specify data structures and functionality at a high level of abstraction without sacrificing the efficiency required by production weapons codes.

FY05 Tasks:

- Organize and convene a series of meetings to explore research directions for components in high-end computing with a special emphasis on the Marmot code. (Quarters 1-3)
- Produce a report defining componentization strategies for support of future generations of ASC codes. (Quarter 4)

1.1.2. Supporting Technologies for Component Integration

Investigators: Ken Kennedy, Keith Cooper, Jack Dongarra, Guohua Jin, Lennart Johnsson, John Mellor-Crummey, Dan Reed

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 from collections of components. This effort will develop advanced compiler technology to integrate collections of components into a high-performance application without sacrificing the performance of hand-integrated codes.

In the strategy we envision, programs would use a high-level scripting language such as Matlab or Python to coordinate invocation of library operations, although traditional languages such as Fortran and C++ could also serve this purpose. 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 are pursuing 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 consists of two phases. The offline translator generation phase digests annotations describing the semantics of library routines, combines them with its own analysis to generate an optimized version of the library, and produces a language translator that understands library entry points as language primitives. The script compilation phase invokes the generated compiler to produce an optimized base language program. The generated compiler must (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 attack the problem of making component integration efficient enough to be practical for high-performance scientific codes. Of particular importance in this context is the problem of efficiently integrating data structure components (e.g., sparse matrices) with functional components (e.g., linear algebra). This work will begin with a simple prototype of Matlab (or Python) that includes arrays with data distribution. Specific array distributions for sparse matrices will be explored as a way of understanding the crucial performance issues. In the

long term, this may lead to a new strategy for introducing parallelism into Matlab and other scripting languages—by distributing the arrays across multiple processors and performing computations close to the data. (The parallel Matlab effort is leveraged through funding from the NSF ST-HEC effort. In this project we hope to apply this work to ASC codes.)

Once the Matlab array prototype has been explored, we will focus on the Marmot mesh data structures with the goal of demonstrating a prototype with adequate efficiency for use in production codes based on these components. The ultimate goal is to make it possible to quickly substitute different mesh data structures in a code without rewriting the functional components and vice versa.

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 may be distributed, so the implementation must consider the performance implications of distributed systems, even if the applications are compiled together.

With these considerations in mind, we plan to pursue research in five 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 based on well-known platforms, such as Matlab and Python.
- Strategies for scalable parallelization of scripting languages such as Matlab and Python.
- Translation of scripting languages to standard intermediate code, especially languages like C.
- 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.
- Assessment of performance/fault tolerance and relation to user code
- Tools for integrating existing code.
- Demonstration of these techniques in specific applications of interest to ASC and LANL, with a special emphasis on codes in the Marmot effort.

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.

Advanced Component Integration Systems: This effort will explore the application of telescoping languages technology to the component integration problem, with a particular emphasis on integrating components that support data structures with those that implement functionality. The effort will also consider technologies for optimizing accesses to the component interfaces emerging from the Marmot code development efforts. The long-term goal of this research is to produce a component integration framework that is efficient enough to be accepted by high-performance application developers, such as those in the LANL ASC program.

Design for Efficient Component 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. A specific issue of importance is design strategies for efficient data structure components.

Component Systems for Heterogeneous Computing Systems: The key challenge in this area is to construct applications that can be flexibly mapped to heterogeneous computing components and adapt to changes in the execution environment, detecting and correcting performance problems automatically. In this activity, we will explore the meaning of network-aware adaptive component frameworks and what the implementation and optimization challenges are for applications constructed from them. In addition, we will pursue research on middleware to support optimal resource selection in heterogeneous environments. A major byproduct of this

work will be performance estimators (described in Section 1.4.1, "Modeling of Application and System Performance") and mappers that can be used to map applications efficiently to heterogeneous computing systems, such as distributed networks and single-box systems containing different computing components (e.g., vector processors and scalar processors). The latter is a characteristic of several planned HPC computing systems.

Compilation of Object-Oriented Languages: Object-oriented languages like C++, Java, and Python have 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. However, these languages have significant performance problems when used for production applications. In this effort we are studying strategies for the elimination of impediments to performance in object-oriented systems.

To achieve this goal, we must develop new compilation strategies for object-oriented languages such as C++, Java, and Python. 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 has focused on Java, through the use of the JaMake high-level Java transformation system developed at Rice in collaboration with the LANL CartaBlanca project. This system includes 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.

In the next phase of research, we will consider how to adapt these strategies to develop tools and compilation strategies that would directly support the code development methodologies to be used in the Marmot effort. Examples include not only the application of object inlining and class specialization, but also the use of type analysis to support the elimination of dynamic dispatch of methods, a major problem for high performance codes written in C++. We will also consider ways to apply these compilation strategies to Python used as a high-level application prototyping system.

FY05 Tasks:

- Produce a simple component integration system based on Matlab as a scripting language, which would include the Matlab-to-C compiler developed under earlier LACSI support. (Quarter 3)
- Design the component integration strategy for supporting the Marmot application development and produce a report on the design. (Quarter 2)
- Develop a preliminary implementation for distributed matrices in Matlab and possibly Python. (Quarter 4)
- Deliver prototype performance modeler for heterogeneous components. (Quarter 1)
- Design and develop preliminary tools to support object-oriented programming in high performance applications, delivering a report describing them and experiments on their effectiveness. (Quarter 4)

1.1.3. Retargetable High-Performance Components and Libraries

Investigators: Jack Dongarra, Lennart Johnsson, Ken Kennedy

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.

Automatically Tuned Library Kernels: 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. We will extend this work to systems that can automatically generate the tuning search space for new libraries using automatic application tuning methodologies described in Section 1.4, "Application and System Performance".

Self-Adapting Numerical Software: We will explore new approaches to building adaptive numerical software that overcomes many of the deficiencies of current libraries. An adaptive software architecture has roughly three layers. First, there is a layer of algorithmic decision making; the top level of an adaptive system concerns itself with the user data, and based on inspection of it, picks the most suitable algorithm, or parameterization of such algorithms. The component responsible for this decision process is an "Intelligent Agent" that probes the user data and, based on heuristics, chooses among available algorithms. Second, there is the system layer; software on this level queries the state of the parallel resources and decides on a parallel layout based on the information returned. There can be some amount of dialog between this level and the algorithmic level, since the amount of available parallelism can influence algorithm details. Finally, there is the optimized libraries level; here we have kernels that provide optimal realization of computational and communication operations. Details pertaining to the nature of the user data are unlikely to make it to this level. Implicit in this approach is a distinction among several kinds of adaptivity. First of all, there is static adaptivity, where adaptation happens during a one-time installation phase. Contrasting with this type of adaptivity is dynamic adaptivity, where at run-time the nature of the problem and environment are taken into account. Orthogonal to this dichotomy is the distinction of adapting to the user data or the computational platform (e.g., memory hierarchy, communication latency/bandwidth or failure modes). We stress the obvious point that, in order to adapt to user data, a software system needs software that

engages in discovery of properties of the input. Oftentimes, such discovery can only be done approximately and based on heuristics, rather than on an exact determination of numerical properties.

Using the above framework we will investigate the use of Matlab as a front-end for computing on a cluster.

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

FY05 Tasks:

- Feature Detector. This component collects timing data and examines it for special features. It may interpolate to fill in gaps or request additional timings to enhance complicated parts of timing curves. (Quarter 1)
- Investigate search space optimization and automatic search space generation; expand to heterogeneous clusters. (Quarter 2)
- Develop and implement UHFFT-style code generation and optimization for a limited set of multi-grid methods to be chosen in collaboration with LANL staff for maximum benefit to the ASC program within given resource constraints. (Quarter 2)
- Implement ATLAS-style tuning to sparse linear algebra and cluster numerical library. (Quarter 3)
- Incorporate optimizations into targeted applications; cultivate second round of applications for optimization. (Quarter 4)

1.2. Systems

Investigators: Rob Fowler, Patrick Bridges, Alan Cox, Jack Dongarra, Kevin Gamiel, Arthur Maccabe, John Mellor-Crummey, Dan Reed, Scott Rixner

The Systems sub-area encompasses research in operating systems and closely allied areas as applied to high performance computing at LANL, specifically within the ASC program. We focus on research problems that will be critical to the program in a multi-year window beginning in FY05. In addition to the needs of ASC, the scope of this discussion is further constrained by the interests and abilities of the researchers, research and development programs funded by other sources at the participating institutions, and the LACSI funding level for the work.

Research issues in Systems are organized into two main areas. First, "networking/messaging" refers to problems specifically related to communication research, spanning low-level network architecture to high-level messaging and parallel I/O. Second, "clustering" encompasses research in software for effective integration of nodes, communication, storage and tools into scalable, high-performance systems.

By the end of FY05, we expect to see dramatic improvements in the raw capabilities of networking hardware and these improvements will become available in commodity products in the succeeding years. Initially, the commercial and industrial emphasis will be on the use of this hardware in network infrastructures (backbones) and in commercial servers. Our challenge is to integrate these technologies into system area networks in new generations of clusters for scientific computing. Software layers must evolve to leverage new hardware to realize better network performance, with lower system overheads, to maintain and enhance the reliability of message passing, and to implement new standards in communication to make systems more useful.

Cluster technology, whether vendor-integrated, user-built Beowulf's, or *ad hoc* aggregations of workstations, have had a huge impact on parallel computing. Because they are effective on many (not all) high-end applications, they have become the backbone that provides capacity computing to LANL, DOE, and the nation.

In recent years, however, it has become apparent that we need a new generation of clusters to improve productivity. Conventional clusters are labor intensive to set up, administer, maintain, and upgrade; in many organizations much of the expense of these activities is invisible because they are spread across staff other than designated system administrators. Better approaches to system integration and system software are needed. Efficiency and manageability will improve the economics of small to moderate scale systems for capacity computing, but they are absolutely necessary in order to build and run scalable capability systems.

A promising approach to dealing with this issue is the single system image (SSI) model of clusters. Initially under LACSI support, later from DOE Office of Science, the Cluster Research Lab in CCS-1 pioneered the Clustermatic SSI software package. While Clustermatic has evolved enough to be useful in production systems, there is still a considerable amount of work to do. This work on the next generation of Clustermatic spans a spectrum from speculative research to "nuts-and-bolts" development work.

Because of the breadth and scale of "next generation Clustermatic", the academic partners are committed to being engaged in this effort. It is therefore important that Clustermatic testbed systems be placed at each of the academic institutions to expose the academic community to the issues (research, development, and operational) of building and using SSI systems. In addition, placing systems at each of the academic institutions will ensure that software efforts are consistent with mainstream Clustermatic development. Rice, the University of New Mexico, and the University of North Carolina acquired such testbeds during FY04.

1.2.1. Efficient, Portable, and Scalable Support for MPI Messaging

Investigators: Scott Rixner, Alan Cox

LANL Contacts: Ron Minnich, Rich Graham

The goals of this research are to investigate the performance tradeoffs of using TCP over Ethernet in cluster computing and to deploy the results of this work on clusters compatible with those in use at LANL. Specialized networks, such as Quadrics and Myrinet, are typically used in cluster computing because they offer higher bandwidth and lower latency than traditional commodity networks. However, raw Gigabit Ethernet is competitive in terms of bandwidth and latency, and it is especially attractive when cost is considered. The drawbacks of Ethernet typically arise because of the way that it is used both by the operating system and the MPI library. With specialized networks, protocol processing is usually handled directly in the MPI library. By doing so, the transport protocol can be tailored to the cluster computing domain by reducing latency, and copying using such techniques as remote DMA. However, these specialized protocols are difficult to develop and improve, and make it difficult to take advantage of many of the features provided by modern operating systems for networking and event management.

In TCP implementations, protocol processing is handled by the operating system, which can be much more efficient than a user-level library. The techniques used in the network stack are mature and are highly optimized for all networking applications. Being in the operating system, all applications benefit from the performance enhancements. Furthermore, Ethernet is clearly less expensive than specialized networks and TCP provides reliability and easy portability across systems. Network servers have been able to achieve extremely high performance levels with TCP, using scalable event notification systems, such as /dev/epoll in Linux, zero-copy I/O, and asynchronous I/O. We have shown that implementing the LA-MPI library with an event-driven messaging thread, which is a well-known technique in the network server domain, can make TCP over Gigabit Ethernet competitive with Myrinet networks with similar raw bandwidth.

We will build on this work and show that other general optimizations to TCP, including zero-copy I/O and TCP segmentation offload, will further improve the performance of our event-driven OpenMPI (previously LA-MPI) library. Memory management within the operating system's network stack can also be a significant bottleneck. We intend to study and improve the memory management within the stack to streamline networking performance. These changes are a combination of improvements to the operating system's network stack and the implementation of the MPI library itself, but are mostly applicable to all network communication, not just MPI messaging, making them valuable beyond the supercomputing domain.

FY05 Tasks:

- Publish a comparison of OpenMPI using our event-driven messaging techniques on TCP over Gigabit Ethernet with OpenMPI using Myrinet. (Quarter 1)
- Demonstrate deployable improvements to the event-driven OpenMPI implementation using techniques to accelerate TCP, mainly including TCP segmentation offload and improved memory management. (Quarter 4)

Long Term Task:

- Eliminate bottlenecks in the TCP over Ethernet MPI implementation using a combination of advanced OS support and programmable network interfaces. Specifically, streamline the use of TCP over Ethernet for cluster computing and eliminate copying on both message sends and receives.

1.2.2. Operating System Issues Related to Scalability

Investigators: Arthur B. Maccabe, Patrick G. Bridges

LANL Collaborators: Ron Minnich, Rich Graham

Work at UNM is focused on low-level performance issues associated with communication and host operating systems. These activities support work at LANL on Clustermatic and OpenMPI (LA-MPI). Clustermatic and OpenMPI are the base-level infrastructure for LANL's ASC codes. Recent work at CCS-3 has shown that system software performance can have a large impact on ASC application performance. Our work is attempting to quantify these impacts and explore the design space of possible solutions to these performance problems in collaboration with researchers in CCS-1.

1.2.2.1. *Scalability of TCP*

Our primary focus in this work is to identify and address limits to scalability in these protocols. The need to manage connection state is perhaps the single biggest factor limiting the scalability of TCP. This is especially true when protocol processing is offloaded to a network interface with limited resources. We have developed and refined methods to determine the actual amount of system memory used per open TCP connection. Using these methods, we have demonstrated that memory usage will become a bottleneck to TCP performance in the future.

We are in the process of defining and implementing a connection-less TCP that allows the user (and eventually the system itself) to deactivate a socket when it is not being used. Deactivation removes the heavyweight socket and replaces it with a timewait structure that is nearly eight times smaller. This should enable us to support tens to hundreds of thousands of TCP connections, most of which are inactive; while a smaller, working set of active connections are fully instantiated. We will measure the costs associated with dynamic activation and deactivation of sockets. This involves defining the appropriate metrics (e.g., round-trip time, congestion window size, slow-start threshold) related to reactivation, cached metrics during deactivation, static metrics and shared metrics across connections (bundling). We are also exploring methods for automatically deactivating sockets when they are not part of the "working set'" for a process and automatically re-activating them as they are needed.

FY05 Tasks:

- Demonstration of automatic deactivation and re-activation of TCP sockets, including performance metrics. (Quarter 1)
- Perform ns-2 simulation of "connection-less" TCP. (Quarter 2)
- Identification of methods for automatic deactivation and re-activation of sockets. (Quarter 3)
- Evaluation of automatic deactivation and re-activation methods. (Quarter 4)

1.2.2.2. *Application Impact of Fault-handling Placement*

Message-passing systems such as MPI have to handle possible hardware faults in message passing, including (primarily) network packet losses, but also possibly including packet data corruption by either the network itself or the host hardware. Message-passing systems can handle such network faults either at a low-level, such as in kernel network protocols, or in user libraries and applications. Both approaches have their advantages, with low-level fault handling potentially offering lower overhead and high-level fault handling providing complete end-to-end reliability.

Preliminary studies at UNM have shown that the actual cost/benefit tradeoffs in these decisions are complex; the costs of allowing for user-level end-to-end reliability in LA-MPI/OpenMPI, for example, may be higher than initially expected even when this functionality is not needed. In addition, the performance benefits and reliability costs provided by fault handling in low-level networking protocols still need to be accurately quantified.

In FY2005, UNM will continue to work with Rich Graham in CCS-1 to study the impact of kernel-level and library-level fault handling in OpenMPI and LA-MPI. By quantifying the performance impact and potential reliability risks of fault-handling placement, we hope to aid LANL in improving both the reliability and performance of OpenMPI and LA-MPI in supporting ASC applications, as well as to help direct future networking and operating systems research at UNM.

FY05 Tasks:

- Port native Ethernet path to OpenMPI. (Quarter 1)
- Quantification of the costs of the reliability portion of LA-MPI/OpenMPI when the reliability protocol elements are disabled. (Quarter 2)
- Evaluation of the user-level visible bit-error rates on an Infiniband cluster. (Quarter 3)
- Design and initial prototyping of a framework for moving protocol \ fault-handling services between user-level, kernel-level, and potentially programmable NICs. (Quarter 4)

1.2.2.3. *Infiniband Testbed*

Infiniband will be an important, if not the preferred, interconnect for future systems based on commodity components. As such, the UNM group is starting to look at the implications of Infiniband as an interconnect. The potential for very high bandwidth, 40 GB/second, in the near future is of particular interest. In the past, the introduction of 1 Gb/second Ethernet led to major changes in operating system structure, including: zero-copy structures and OS-bypass. We

anticipate that the increase to 40 GB/second will lead to similar changes in OS structures. In this context, we are working with Ron Minnich from CCS-1 to design a networking testbed that will allow us to experiment with a network interface that has a very powerful processor.

FY05 Tasks:

- Design of network testbed with powerful NIC processors. (Quarter 1)
- Identification of OS issues to explore using the testbed. (Quarter 2)
- Implementation of network testbed. (Quarter 3)
- Initial performance study using the testbed. (Quarter 4)

1.2.3. OpenMPI

Investigator: Jack Dongarra

OpenMPI is a community version of MPI. Each of the core contributors is the developer of an existing production-quality implementation of the Message Passing Interface (MPI) standard—FT-MPI (UTK), LA-MPI (LANL) and LAM/MPI (IU)—which offer various approaches to data and process fault tolerance in addition to high-performance communication. The OpenMPI project is developing a highly configurable and extensible runtime environment— or middleware—to support robust parallel computation on systems ranging from small mission-critical and embedded systems to future petascale supercomputers. OpenMPI has a light-weight component architecture that allows for on the fly loading of component modules and run-time selection of features (including network device, OS, and resource management support), enabling the middleware to be highly adaptable, both statically to accommodate a wide variety of system types, and dynamically in response to rapidly changing heterogeneous environments. The architecture also provides an ideal framework for adding support for experimental or innovative devices. Project OpenMPI's initial goal is to provide a framework for a new high-quality implementation of MPI Version 2 with high levels of communication performance, scalability to hundreds of thousands of processes, and data and process fault tolerance. The first release of open-MPI is scheduled for November 2004. OpenMPI was designed, however, to be the foundation of more complete runtime environment than a simple message-passing library. A central goal of OpenMPI is to enable effective fault management (an essential requirement for scalable computers). Middleware such as OpenMPI is uniquely positioned to coordinate and broker the tasks of fault prediction, detection, recovery and reconfiguration. We do not propose to provide a fully automatic or "canned" solution to fault management, but rather to provide a consistent and common APIs so that applications can discover, characterize, and respond appropriately to faults.

The low-level communication layer of OpenMPI is designed with high-performance in mind, providing low latency, and scalable high bandwidth through the striping of message fragments across multiple network devices, with optional end-to-end data integrity through a lightweight checksum/retransmission protocol. The design is structured in such a way that all or part of the communication protocol may be offloaded to network-device processors on architectures where this is beneficial. Finally, OpenMPI is highly portable, conforming to ISO C and POSIX standards throughout. This enables us to target a variety of operating systems, including novel choices such as Plan 9 and realtime operating systems (RTOSs).

FY05 Tasks:

- Complete the Beta release Open MPI Release at SC, Pittsburgh USA November 2004. (Quarter 1)
- Produce a report on OPEN-MPI in Heterogeneous Network Environment. Implement collective communication devices based on various flavors/algorithms layered on point-to-point. (Quarter 2)
- Generate a first implementation of Data Fault Tolerance (expand on work LA-MPI). (Quarter 3)
- Investigate and produce a report on various collective communication optimizations. (Quarter 4)
- Implement an initial Process Fault Tolerance (expand on work FT-MPI). (Quartet 4)
- Implement collective communication devices based on various flavors/algorithms via shared and distributed memory. (Quarter 4)

1.2.4. Highly Scalable Fault Tolerance

Investigators: Dan Reed, Kevin Gamiel

As supercomputers scale to tens of thousands nodes, reliability and availability become increasingly critical. Both experimentation and theory have shown that the large component counts in very large-scale systems mean hardware faults are more likely to occur, especially for long-running jobs. The most popular parallel programming paradigm, MPI, has little support for reliability (i.e., when a node fails, all MPI processes are killed, and the user loses all computation since the last checkpoint). In addition, disk-based checkpointing requires high bandwidth I/O systems to record checkpoints. The collaborative OpenMPI effort promises one possible solution to application-mediated recovery.

To complement this effort, we will build on and expand development of tools for real-time monitoring of system failure indicators (e.g., temperature, soft memory errors and disk retries), tied to Clustermatic and other scalable cluster infrastructures. These tools will include mechanisms to estimate node failure probabilities, as a basis for fault tolerance techniques. We will develop and expand "performability" models that combine both fault-tolerance and performance for systems containing thousands of nodes. These models will include total time to solution as a function of failure modes and probabilities.

The modeling will be complemented by an experimental harness in which developers of scalable fault-tolerant applications will be able to test their codes by selecting special batch queues with controls for likely failure probabilities and modes. The latter will rely on a set of fault injection tools that can assess the susceptibility of large-scale applications to transient memory, network interface card (NIC) or storage errors.

Our goal is to estimate node failure probabilities and introduce enough redundancy to enable recovery. This approach complements disk-based checkpointing schemes to recover from failures between disk checkpoints. We envision it is a low overhead checkpoint alternative that can be performed much more often than disk checkpointing, triggered either periodically or via system measurements. Finally, we expect this approach to include intelligent learning and adaptation. By monitoring and analyzing failure modes, the system can estimate the requirements

adaptation to achieve a specified reliability. This will enable smoothly balancing performance and reliability.

In addition to these issues, fault tolerance data collection must be scalable and integrated with low overhead performance measurement systems. We will investigate integrated, sample-based measurement schemes that can collect failure and performance data from systems containing thousands or tens of thousands of nodes.

We will also work with the OpenMPI effort to integrate these predictive capabilities with newly developed MPI fault tolerance mechanisms, with extensions to adaptively choose checkpoint frequencies (either disk or memory) based on predictive failure probabilities. Our goal is an adaptive MPI system that can estimate and configure the degree of redundancy and disk or memory checkpointing needed to ensure reliable computation.

FY05 Tasks:

- Failure mode instrumentation and analysis toolkit development and extension. (Quarter 2)
- Integrated performability measurement tools suitable for very large systems. (Quarter 3)
- Adaptive checkpoint frequency selection. (Quarter 3)
- Semi-automated batch queue selection based on failure modes. (Quarter 4)

1.2.5. Clustermatic Performance Instrumentation

Investigators: Rob Fowler, Patrick Bridges, John Mellor-Crummey

Application performance engineering requires a performance instrumentation and analysis infrastructure that is robust and scalable while providing the analysis capabilities needed by developers of both system and application code. There is a demand for instrumentation of processor performance within an application process and across all code running on a node. While this may be sufficient for measuring compute-bound applications, operating system operations play a vital role in the performance of applications that communicate with the outside world, including messaging in parallel applications and any interaction with I/O subsystems. Measuring these costs will require adding instrumentation to the operating system kernel and to the specific device drivers that contribute to the costs.

It will be crucial that the performance instrumentation infrastructure be scalable and impose low enough overheads that it can be used to measure production runs on large systems.

1.2.5.1. *Performance Counter Profiling*

HPCToolkit from Rice runs on current Clustermatic systems by layering itself on top of PAPI from Tennessee. One problem with this approach is that it allows one to look at internal performance of an application, but it does not provide a system-wide view that captures all phenomena relevant to performance. We will investigate adding such pervasive performance monitoring and analysis infrastructure into Clustermatic systems. The approach taken will be to begin with the *oprofil* software and extend and modify it to work on Clustermatic systems. This work is coupled to the activities described in Section 4.1, "Application and System Performance."

FY05 Tasks (Rice):

- Integrate a pervasive performance instrumentation system such as *oprofil* with Clustermatic and layer HPCToolkit on top. (Quarter 2).
- Deploy instrumentation and analysis at LANL. Hold a workshop on the use of the extended tool set. (Quarter 4)

1.2.5.2. *Fine-grained monitoring of System Software Costs*

General operating system costs are becoming increasing impediments to ASC application performance on large-scale machines. Recent studies at CCS-3, for example, have shown that operating system effects in the SAGE ASC code can cause up to a 50% performance penalty on large-scale systems such as ASCI-Q. The current solution to this problem is to dedicate approximately 12.5% of ASCI-Q to operating system services to OS interference issues. While this approach does mitigate the problem, it comes at making a non-trivial portion of the ASCI-Q system unavailable to applications.

To address these issues, UNM is working on novel approaches to measuring operating system and message-passing costs in large-scale systems that are central to ASC's mission. The first part of this research consists of modifying the Linux kernel to monitor and report the operating system costs associated with each network transmission and reception on a per-request basis. By augmenting the Linux kernel with per-request monitoring facilities, we aim to quantify the exact operating system costs that cause operating system interference performance effects similar to those measured at CCS-3. These per-request monitoring facilities then can be used to guide later modifications of Linux to increase message-passing performance, and integrated with more comprehensive system monitoring facilities.

One such comprehensive system monitoring approach is the focus of UNM's other LACSI research on monitoring. This approach, which we term message-centric monitoring, seeks to extend this approach to the entire system and to measure the complete hardware, operating system, and communication costs associated with ASC message-passing codes. Instead of examining the performance of individual requests on a host-by-host basis, our message-centric profiling approach associates performance data with the data in MPI messages as this data propagates across the system, is received by one host, processed by an application, and sent to another host. The overall goal is to have the performance data associated with a message encompass the entire diameter of the computation required to generate it, with the emphasis on profiling message-passing and operating system costs.

FY05 Tasks (UNM):

- General framework for collecting per-request performance data in Linux on both network and disk requests. (Quarter 1)
- Evaluation of message-centric monitoring in LAMPI on medium and large-scale applications. (Quarter 2)
- Integration of per-request performance monitoring with message-centric monitoring. (Quarter 3)

- Evaluation of the feasibility of integrating per-request and message-centric monitoring information into HPCToolkit. (Quarter 4)

1.3. Computational Science

Investigators: Yuri Kuznetsov, Mike Fagan

The *Computational Science* effort focuses on the development, analysis, and verification and validation (V&V) of numerical solution techniques for physical models embodied within large-scale multi-physics simulation tools designed to address today's leading problems in science and engineering. Key applications currently include the predictive simulation of weapons manufacturing and performance as supported by the DOE Advanced Simulation and Computing (ASC) Program and global climate modeling as supported by the DOE Scientific Discovery Through Advanced Computing (SciDAC) Program. The computational science effort can be divided into three principal research thrust areas: algorithms and models for specific physical phenomena of interest, numerical methods for the algorithmic coupling of these physical phenomena, and metrics for correctness and robustness of these models and algorithms. The thrust areas are:

1. Numerical Solution of Partial Differential Equations for Continuum Dynamics, Energy Transport, and Materials Science;
2. Linear and Nonlinear Solvers; and
3. Methodologies for V&V, Sensitivity, and Uncertainty Quantification.

A key product of this effort, both in the long and short term, is verified and validated software components constructed with defensible (demonstrable) software quality engineering practices. These components must instantiate robust and accurate solution techniques for the physical models required by the multi-physics simulation tools. The computational science effort devoted to "multi-physics coupling" algorithm research is necessary for the faithful simulation of multiple, simultaneously-occurring physical phenomena.

Long-term goals. Ensuring computational science follows the fundamental principles of the scientific method requires long term investigation of numerical methods and algorithms and careful software development. For example, a physicist or engineering analyst using these simulation tools should be able to generate high fidelity three-dimensional simulations, attain similar answers with two different numerical techniques, and be assured that each technique has been verified and validated. Because the transformation of physical principles into software can take many different paths, long-term research focuses on the investigation of new, possibly high-risk, methods along with new ideas for the improvement of classical methods that are parallel and scalable.

Experience shows investigation of new methods must be built upon the foundation of good software quality engineering. Unit-testing and component-based designs for even one-dimensional tests are necessary to assess the impact of this long-term research on next-generation simulation tools.

Long-term goals of the computational sciences effort include:

- Understanding the physics and mathematics of the phenomena to be simulated so that improved numerical methods can be devised that are both robust and accurate;

- Developing new algorithms for the resulting physical models that possess good single processor performance as well as being parallel and scalable;
- Instantiating these algorithms into component-based software as guided by sound software quality engineering practices. Unit-testing is of primary importance compared to reusability;
- Developing improved and automated methodologies for the verification of the algorithms and the software and the validation of the models; and
- Devising strategies for successful team software development of large-scale simulation tools.

Short Term Plans

With the LACSI funds currently projected to be available in FY05, we propose to fund the projects described in the following sections. The first project, "Code-based Sensitivity Analysis," is in the "Methodologies for V&V, Sensitivity, and Uncertainty Quantification" thrust area. The second project, "Adaptive Numerical Methods for Diffusion and Transport Equations in Heterogeneous Media on Distorted Polyhedral Meshes," is in the "Numerical Solution of Partial Differential Equations for Continuum Dynamics, Energy Transport, and Materials Science" thrust area.

1.3.1. Code-Based Sensitivity Analysis

Investigator: Mike Fagan

LANL Collaborators: Ken Hanson (CCS-2), Jim Sicilian (CCS-2), John Turner (CCS-2), Ralph Nelson (X4)

Predictive computational models used for stockpile stewardship studies require sophisticated models simulated on the world's largest computers. These models are complex; hence advanced verification ("solving the equations right") and validation ("solving the right equations") methodologies are needed to assess their accuracy and predictive capability. In every major ASC simulation code, complex subsystems interact in complex ways to form cohesive computer programs that predict important physical processes. Sophisticated, component-based software enables analysts to unit test and verify the codes even if there are major improvements and changes to the subsystems of the code. Finally, even with verified numerical algorithms (in the physics and software sense) and validated physical models, the uncertainty of the model/algorithm and its sensitivity to change must be better understood.

A priority of LANL's mission is to validate and verify ("V & V") the complex computer programs used to model equally complex physical processes. One of the major techniques employed in the verification and validation process is sensitivity calculation. Consequently, the aim of the code-based sensitivity analysis project is to develop methods for accurately and efficiently computing sensitivities of complex scientific simulation programs. Three projects at LANL are the principal targets for code-based sensitivity analysis over the short term: the Telluride Project, the Shavano Project, and the Marmot Project.

FY05 Tasks:

- Continue to develop Adifor 90 algorithms and general AD infrastructure. (Ongoing)
- Continue to assist LANL in the application of Adifor 77 codes of interest. (Ongoing)
- Develop software tools for validating derivative code. (Ongoing)

- Write a technical report about the optimal step size for verifying derivatives. (Quarter 2)
- Demonstrate prototype tool for automatic differentiation of object code. (Quarter 3)
- Demonstrate ability to apply Adifor 90 to differentiate a substantial part of the Truchas code. (Quarter 4)
- Initiate development of Infrastructure for augmentation of C/C++/Java codes. (Quarter 4)

1.3.2. Adaptive Numerical Methods for Diffusion and Transport Equations in Heterogeneous Media on Distorted Polyhedral Meshes

Investigator: Yuri Kuznetsov

LANL Collaborators: J. Morel (CCS-2), G. Olson (CCS-4), M. Shashkov (T-7)

Efficient numerical methods for the diffusion and radiation transport equations in highly heterogeneous media on general distorted polyhedralmeshes 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, for instance, to the T-7 and CCS-4 groups, and at UH.

The project is based on the results of very successful cooperation between researchers at LANL and at UH. In 2002-2003, Yu. Kuznetsov conceived of a fundamentally new approach for solving the diffusion equations on general polygonal and polyhedral meshes by the mixed finite element method. In 2003-2004, the idea of this method was applied by researchers from LANL (M. Shashkov, J. Morel, and K. Lipnikov) and UH (Yu. Kuznetsov) to design new accurate and physically consistent mimetic discretizations based on the support operator method for the diffusion equations on polygonal meshes. The resulting method represents a genuine breakthrough in the numerical solution of the diffusion equations on arbitrary polygonal meshes including locally refined (AMR) and nonmatching ones. The method is slated for implementation in certain ASC projects at LANL. Extension of the method to polyhedral meshes with application to 3D diffusion equations has been done recently (FY 2004).

In this project (FY 2005), we plan to continue joint research on development and investigation of the proposed methods as well as on implementation aspects of the method and LANL relevant applications.

Long term goal: The major long term goal of the project is to develop, investigate and evaluate on the test problems relevant to LANL applications new adaptive mimetic compatible discretizations and efficient parallel multilevel preconditioners/solver for the diffusion and radiation transport equations in heterogeneous media on strongly distorted polyhedral meshes.

FY05 Tasks:

- To implement the proposed polyhedral discretization method and to evaluate its accuracy and efficiency on 3D test problems relevant to ASC applications. To deliver a technical report with description of test problems and results of numerical experiments.
- To investigate convergence properties of the proposed methods and to derive a posteriori error estimation for polygonal discretizations of the diffusion equations.

- To develop an AMR methodology based on a posteriori error estimator for the polygonal discretizations of the diffusion equations and to evaluate it on test problems relevant to ASC applications. To deliver a report with results of numerical experiments.
- To develop, investigate and evaluate on selected test problems the new multilevel preconditioner based on macro-element coarsening in the space of the Lagrange multipliers. To deliver a report with description of preconditioners to be developed and results of numerical experiments.

1.4. Application and System Performance

Investigators: John Mellor-Crummey, Keith Cooper, Jack Dongarra, Robert Fowler, Guohua Jin, Dan Reed, Linda Torczon

Building scientific applications that can effectively exploit extreme-scale parallel systems has proven 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 their capabilities. 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 and system performance research thrust are

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

A broad spectrum of issues affects application performance, including operating system activity, 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. A quantitative assessment of factors limiting application performance on current-generation architectures will help focus long-term research on software and hardware technologies that hold the most promise for improving application performance and scalability on future systems. 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 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 ASC application teams by providing them with more efficient programming models, technology for compiler-assisted tuning of applications, better performance instrumentation and diagnostic capabilities, insight into the performance and

scaling of applications and systems through modeling, improved algorithm-architecture mapping, and better performing extreme-scale parallel architectures.

1.4.1. Modeling of Application and System Performance

Investigators: John Mellor-Crummey, Robert Fowler

LANL Collaborator: Adolphy Hoisie

Performance models are an important tool for gaining insight into applications and systems. They can be used for scalability analysis on both existing and proposed future architectures, in procurement to compare proposed alternatives, in software development to ascertain the performance impact of code re-configuration prior to implementation, and in real-time to steer the processing of code to increase processing efficiency. Accurate models are useful for guiding architecture design. The modeling of high performance software and hardware systems is highly complex, requiring the encapsulation of key processing structures and characteristics. This is a direct result of the performance space being multi-dimensional and highly non-linear in any of its dimensions.

Research in this area at LANL and Rice will span a wide range of topics. At Rice, the focus of research will be on designing, building and evaluating semi-automatic tools for synthesizing models and model components, as well as exploring how to integrate model components synthesized automatically into hand-crafted model frameworks. We will use the tools that we build to synthesize prototype models for target applications and then evaluate how effectively our semi-automatically generated models predict application behavior on a set of target systems.

FY05 Tasks:

- Refine capabilities for modeling and prediction of an application's memory hierarchy performance for a range of applications, architectures, and problem sizes. (Quarter 1)
- Refine capabilities for combining memory hierarchy and computation predictions to improve prediction accuracy; evaluate relative accuracy of cross-architecture predictions of node performance for multiple architectures. (Quarter 2)
- Explore how to extend capabilities for predicting node performance of sequential programs to predict performance of parallel programs. Explore strategies for integrating Rice's semi-automatically generated node performance predictions with LANL's whole system models. (Quarters 3-4)

1.4.2. Better Tools for Measurement and Analysis of Application Performance

Investigators: Robert Fowler, John Mellor-Crummey, Dan Reed

On terascale systems, performance problems are varied and complex. Hence, a wide range of performance evaluation methods must be supported. The appropriate data collection strategy depends on the aspect of program performance under study. Key strategies for gathering performance data include statistical sampling of program events, inserting instrumentation into the program via source code transformations, link time rewriting of object code, binary modification before or during execution, or program state modification during execution.

Capturing traces of program events such as message communication helps characterize the temporal dynamics of application performance; however, the scale of these systems implies that a large volume of performance data must be collected and digested. Improved data collection strategies are needed for collecting more useful information and reducing the volume of information that must be collected. Statistical sampling provides a formal basis to achieve desired estimation accuracy under a certain measurement cost. We will investigate the feasibility of using statistical sampling and population dynamics techniques to characterize performance on large systems. This approach will enable tunable control of measurement accuracy and instrumentation overhead. Concurrently, we will explore application of these techniques to the temporal domain, with a goal of bounding temporal performance trajectories.

Research problems to be addressed include determining the appropriate level for implementing different instrumentation and measurement strategies, how to support a modular and extensible framework for performance evaluation, as well as the appropriate compromise between instrumentation cost, the level of detail of measurements, and the volume of data to be gathered.

Current tools for analysis of application performance on extreme-scale systems suffer from numerous shortcomings. Typically, they provide a myopic view of performance emphasizing descriptive rather than prescriptive data (i.e., what happened rather than guides to improvement), and they do not support effective analysis and presentation of data for extreme-scale systems. 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.

Comparing profiles based on different events, computing derived metrics (e.g., event ratios), and correlating profile data with routines, loops and statements in application code can provide application developers with insight into performance problems. However, better statistical techniques are needed for analyzing performance data and for understanding the causes and effects of differences among process performance. Instead of modeling each system component, these techniques select a statistically valid subset of the components, and model the members of that subset in detail. Properties of the subset are used as a basis in estimates for the entire system. Our research in this area, so far, has focused on system availability. We plan to expand that scope and apply these techniques to study application performance. The main goal is to evaluate how well application performance can be characterized and understood, based on a more efficient data collection scheme.

FY05 Tasks:

- Explore integrating information about dynamic execution context (i.e., call paths) into HPCToolkit and explore strategies for analyzing and presenting such profiles for large applications. (Quarters 1-2)
- Develop and demonstrate an analysis package for combining and analyzing HPCToolkit performance profiles for a large collection of node computations. (Quarters 2-3)
- Explore extensions for improving the effectiveness of the HPCToolkit user interface for exploring the performance of parallel applications. (Quarter 4)

- Interact with LANL application researchers to (i) characterize the behavior of their applications executed on large-scale systems, and (ii) explore opportunities for performance improvements based on the findings produced by this characterization. (Ongoing)
- Continue refinement of the PAPI interface for accessing hardware performance counters. The goal of this effort is to provide a robust implementation of PAPI including features such as thread safety, counter multiplexing, and counter-driven user callbacks on important computing platforms. (Ongoing)
- The academic performance analysis team will continue to hold performance tools workshops at LANL if the applications teams or LANL management believe additional such workshops would be productive.

1.4.3. Automatic Application Tuning

Investigators: Keith Cooper, Ken Kennedy, John Mellor-Crummey, Linda Torczon

Increased complexity in both applications and architectures has created an environment in which producing effective code is difficult. The classic software production cycle, in which an application is compiled once at a high-level of optimization, is no longer sufficient to produce high-quality executable code. Systems that use run-time adaptation, such as ATLAS, or that generate code tailored for specific problem instances, such as UHFFT, demonstrate that adaptive strategies can produce consistently good results. Building tools that incorporate and automate such adaptation is a major challenge. The goal of this project is to achieve results comparable to those of ATLAS or UHFFT using automatic techniques—thereby making the benefits of such adaptation available over a wider range of applications. (This goal stands in contrast to the work in Section 1.1.3, which aims to generate additional software libraries that implement their own adaptive behavior. Success in this project will complement success in that project.)

Adaptive Optimization Strategies: Modern compilers use a handful of strategies to improve each optimization. Typically, they apply the same strategies to all programs. For example, GCC supports three levels of optimization, –O1, -O2, and –O3, each representing a fixed strategy. Recent work has shown that program-specific strategies can produce consistently better code; several studies suggest that the improvements from program-specific strategies range up to 25% over any fixed strategy.

The problem with program-specific strategies lies in the cost of discovering them. One goal of this project is to develop cost-effective techniques to discover and apply program-specific optimization strategies. The work includes strategies for compiler configuration (e.g., both the set of optimizations to run and an order in which to apply them), for determining command-line parameter settings (e.g., GCC offers roughly fifty individual flags that can control different aspects of the individual passes), and for controlling the application of specific optimizations (e.g., loop blocking or inline substitution).

Performance-based Optimization Strategies: Recent work in performance analysis and modeling has enabled tools to identify performance bottlenecks in an application. Tools such as the HPCToolkit can use hardware performance counters to pinpoint both a region and a problem, as in "this inner loop has excessive L2 cache misses." Classic optimizing compilers have no way

to target such problems; in particular, techniques to ameliorate one region's problem may exacerbate the problems of another region.

Performance-based optimizations will combine a regional approach to applying a particular transformation (as opposed to uniform application across an entire procedure) with a feedback-based steering mechanism that selects transformations and regions based on actual or predicted performance problems.

Research is needed into a spectrum of technologies to support effective whole program tuning. This research will include techniques to identify regions of inefficiency and to pinpoint symptoms of inefficiency (e.g. excessive TLB misses in a particular loop), strategies for coordinated application of integrated code transformations to ameliorate program bottlenecks, and search techniques for determining what the next step should be to tune the program based on the results of tuning attempts thus far.

FY05 Tasks:

- Deliver a prototype tool for automatically tuning whole applications for the x86 architecture based on feedback from empirical performance measurements. The tool will use the HPCToolkit package for collecting performance measurements and will use a search strategy to tune transformation parameters such as tile sizes and unroll factors to direct LoopTool. (Quarter 1)
- Demonstrate applicability of adaptive compilation sequences in compilers other than our research prototype. We will work within the LLVM system to show the improvements from optimization choice. (Quarter 2). We will experiment with adaptive compilation sequences in Microsoft's new Phoenix compiler infrastructure. (Quarters 3—4, pending licensing and availability issues)
- Explore adaptive techniques for control of an individual optimization. We will complete our study of adaptive control of source-to-source inline substitution. (Quarters 2—3). We will release our prototype adaptive inliner as a standalone tool. (Quarter 4)
- Expand our experiments on compilation-order decisions to include source-level application properties. We will develop source-level metrics and try to correlate them with effective compilation sequences. (Quarters 3—4)

1.4.4. Compiler Technology for Exploiting Modern Processors

Investigators: Keith Cooper, Ken Kennedy, John Mellor-Crummey, Linda Torczon

To keep pace with the Moore's law curve and deliver 60% annual increases in processor performance, architects have increased the complexity of commodity processors and the memory systems that surround them. To produce code that achieves a significant fraction of peak performance on a modern commodity processor (e.g., Pentium, IA-64, Opteron, SPARC, or MIPS), a compiler must apply a complex series of transformations to the code (optimization) and then translate the result into the appropriate assembly code (code generation). To create code that executes efficiently, the compiler must address a number of challenging problems.

1. The code must keep the functional units busy. The optimizer must transform the input program so that it has enough instruction-level parallelism to sustain the computation rate as well as an appropriate instruction mix. The code generator must discover a dense

instruction schedule for the final code—it may need to use different scheduling algorithms for different points in the code, making the choice on a loop-by-loop or block-by-block basis.

2. The optimizer must transform the code so that its pattern of memory accesses matches those of the processor and memory system—adjusting locality with blocking, prefetching, and (perhaps) streaming. After the optimizer has rewritten the code so that it can move sufficient data onto the chip in a timely fashion, the code generator must manage instruction and data placement so that operands are kept in appropriate registers and, for clustered register-file machines, in the cluster where the operand is consumed.
3. The optimizer and the code generator must work together to make effective use of processor features such as predicated execution, register windows, register stacks, auto-increment options, branch-delay slots, and hints to the hardware about locality and branch targets.

Research on this project is aimed at developing new techniques to address these problems—techniques suitable for implementation in either open source or commercial compilers, *and* at improving the quality of optimization and code generation available in both open source and commercial compilers for commodity processors used in high-performance computing.

FY05 Tasks:

- Continue our experiment with automatic choice of command-line parameters for IA-64 compilers (both ORC and the Intel compiler). In Quarters 1—2, we will quantify the potential improvement from parameter setting. In Quarters 3—4, we will package those results in a tool that automatically finds appropriate, application-specific parameter settings.
- Investigate the use of dynamic reoptimization to improve performance on scientific codes. We will use the LLVM framework (support for Pentium, PowerPC, and Sparc, with Opteron in process) in this work. Release new register allocators for LLVM. (Quarter 2). Begin development of an advanced scheduler for LLVM. (Quarters 3—4)
- Investigate the use of algebraic reassociation in conjunction with strength reduction to reduce the number of integer instructions created in critical blocks. We will work with partners at LANL to identify critical loops that schedule poorly due to instruction mix (Quarter 1) and develop reassociation strategies to reduce the operation count. (Quarters 2—3)

1.4.5. Application Mapping, Dynamic Adaptation and Steering

Investigators: Dan Reed, Ken Kennedy

As computer systems grow in size and complexity, tool support is needed to facilitate the efficient mapping of large-scale applications onto these systems. Today, most applications are mapped to a set of resources at program launch and then run to completion using these resources. However, large-scale systems built from commodity components are prone to failure and long-running applications for such systems must sense and respond to component failure.

Intelligent mapping and performance steering offer 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). A challenge is to develop strategies that enable applications running on ASC-scale systems to monitor their own behavior and reactively adjust their behavior to optimize performance according to one or more metrics. For this purpose, performance analysis tools must provide robust performance observation capabilities at all levels of the system and the ability to map low-level behavior to high-level program constructs. Our goal is to develop tools and approaches that can help applications achieve high performance even when system components fail or applications are subject to other system constraints – managing the challenge of large scale and integration with multiple subsystems. *This work will explicitly target the LANL Clustermatic infrastructure*.

Our goal is to develop tools and approaches that can help applications achieve high performance even when system components fail or applications are subject to other system constraints. Strategies for automatic performance steering based on performance and fault models offer 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. Moreover, measurement of environmental conditions on nodes promises to allow users and schedulers to balance checkpoint frequency and partition allocation based on failure likelihood.

In addition, 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. This work continues to explore the use of performance contracts to guide the monitoring of application and resource behavior; contracts will include dynamic performance signatures and techniques for locally (per process) and globally (per application and per system) evaluating observed behavior relative to that expected.

FY05 Tasks:

- Explore techniques for "performance contracts" (i.e., application and system level techniques for closed loop performance monitoring and adaptation to approximate fixed performance levels) for use with Clustermatic. (Quarters 1-3)
- Demonstrate adaptation techniques for multi-attribute system behavior. (Quarter 4)

1.4.6. Compiler Technology for Extreme-scale Systems

Investigators: John Mellor-Crummey, Ken Kennedy, Guohua Jin

Today, MPI is the dominant programming model for writing scalable parallel programs. MPI has succeeded because it is ubiquitous and it makes it possible to program a wide range of commodity systems efficiently. However, as a programming model for extreme-scale systems, MPI has numerous shortcomings. For instance, when using MPI, the programmer must assume all responsibility for communication performance including choreographing asynchronous communication and overlapping it with computation. This complicates parallel programming significantly. 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. Also, when one uses MPI, only coarse grain communication is efficient; this has a profound impact on the way programs

are structured. When an architecture supports a global name space and fine-grain low latency communication, other program organizations can be more efficient.

Global address space programming models are likely to emerge as the simplest to program and most efficient for emerging systems such as Cray's Red Storm and future systems that arise out of DARPA's HPCS project. SPMD global address space 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 on modern computer systems because copies are costly.

However, global address space languages such as UPC and CAF are relatively immature, as is compiler technology to support them. Making these languages simple and efficient to use will require refining language primitives for efficiency and performance portability, developing new analysis and optimizations for SPMD programs, developing compiler support for tolerating latency and asynchrony, as well as developing supporting run-time mechanisms

Beyond explicitly parallel SPMD programming models, data-parallel models such as High Performance Fortran and Cray's Chapel language 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. For parallel programming models to succeed, their use and appeal must extend beyond just extreme-scale machines; therefore, sophisticated compiler technology is needed for these languages to make them perform well on today's relatively loosely-coupled clusters as well as tightly-coupled petascale platforms of the future.

Higher-level data-parallel programming models such as HPF and Chapel pose significant challenges to compilers. Generating flexible high-performance code that runs effectively on a parameterized number of processors is a significant problem. We will continue to investigate analysis and code generation techniques with the aim of having compilers transform complex programs that use sophisticated algorithms into parallel programs that yield scalable high performance on a range of parallel systems.

FY05 Tasks:

- Evaluate compiler technology for retargeting IMPACT 3D, an HPF application written for the Earth Simulator, to microprocessor-based systems. (Quarter 1)
- Explore compiler-based strategies for managing communication buffers in Co-array Fortran to better support pipelined applications such as Sweep3D. (Quarter 2)
- Design, evaluate and report on analysis and code generation strategies with the aim of yielding efficient parallel code for sophisticated algorithms such as multigrid. (Quarter 3)

- Refine algorithms for effectively partitioning computations expressed using a global memory model in the presence of complex data partitionings and dependence patterns. Write a paper about the refined partitioning and code generation strategy. (Quarter 3)

1.5. Computer Science Community Interactions

Fostering collaborative relationships between LACSI participants at LANL and at the LACSI academic sites is a principal LACSI goal. Because LACSI is a collaborative research effort, effective means of supporting collaborations are important to LACSI's success. To encourage collaboration, the LACSI academic institutions support a variety of opportunities for researchers and students from Los Alamos and the academic partner sites to visit each other, to share ideas, and to actively collaborate on technical projects. LANL has also hosted speakers from the LACSI academic sites as part of the ACL Seminar Series. Researchers from the LACSI academic sites are available to speak in the ACL Seminar Series during the FY05 project year.

In addition to hosting visitors and speakers, the LACSI academic partners in conjunction with LANL organized and hosted technical workshops and meetings held at LANL during FY04 on topics related to the LACSI technical vision. Specifically, a performance modeling and prediction meeting and a Components mini-workshop with the Marmot group at LANL were held in June 2004. During the FY05 project year, the LACSI academic partners in conjunction with LANL will organize, host, and otherwise support technical workshops and meetings on topics related to the LACSI technical vision. For example, plans are underway to hold a performance modeling and prediction meeting at Rice in October 2004. We are actively seeking additional funds to support larger meetings on topics relevant to LACSI.

To reach a broader community, LACSI hosts 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 is a traditional conference-style meeting with participation by both LACSI members and scientists from the community at large. The FY04 LACSI Symposium was held October 27-29, 2003 in Santa Fe, New Mexico. The FY05 LACSI Symposium will be held in Santa Fe, New Mexico, October 12-14, 2004. Details related to the LACSI symposium are available at http://lacsi.lanl.gov/symposium. In addition, the LACSI academic partners in conjunction with LANL disseminated information about LACSI and its research results at Supercomputing 2003 and will disseminate information at Supercomputing 2004.

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

2. Management and Administration

Andy White (LANL) directs LACSI in conjunction with Ken Kennedy (Rice), who serves as co-director of LACSI and director of the academic portion of the LACSI effort. Rod Oldhoeft (LANL) and Linda Torczon (Rice) assist the directors as executive directors. The directors make significant decisions with the advice of the LACSI Executive Committee (EC), which includes the site director for each of the six LACSI sites, key LANL personnel, the project directors for the academic portion of each of the strategic thrusts, and the executive directors. The EC currently consists of the following members:

- Andy White, Chair, *Los Alamos National Laboratory*
- Ken Kennedy, co-Chair, *Rice University*
- Jeff Brown, *Los Alamos National Laboratory*
- Jack Dongarra, *University of Tennessee at Knoxville*
- Bill Feiereisen, *Los Alamos National Laboratory*
- Rob Fowler, *Rice University*
- Adolfy Hoisie, *Los Alamos National Laboratory*
- Lennart Johnsson, *University of Houston*
- Deepak Kapur, *University of New Mexico*
- Doug Kothe, *Los Alamos National Laboratory*
- Yuri Kuznetsov, *University of Houston*
- John Mellor-Crummey, *Rice University*
- Rod Oldehoeft, *Los Alamos National Laboratory*
- Dan Reed, *University of North Carolina at Chapel Hill*
- John Thorp, *Los Alamos National Laboratory*
- Linda Torczon, *Rice University*

The EC is responsible for planning and reviewing LACSI activities on a regular basis and establishing new directions, along with new goals and modified milestones. The EC evaluates progress based on the quality of the research performed and its relevance to LACSI goals. Based on the outcomes of its reviews of LACSI research and other LACSI activities, the EC might identify projects to phase out and propose a collection of projects to be undertaken, along with goals for those projects. LACSI researchers to lead the new efforts would be identified and work would be initiated. The resulting work would be evaluated in subsequent reviews.

In March 2002, the EC met with LACSI researchers at LANL to discuss methods of addressing issues raised in the 2001 LACSI contract review. The group developed a framework 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. A description of the long-term vision, framework, and objectives is available in a document (LAUR # 02-6613) titled *Priorities and Strategies.*

In April 2003, the EC met with senior LANL personnel to revise the framework, priorities, and strategies established at the planning meeting in 2002 and *Priorities and Strategies* was revised to incorporate the results of the April 2003 planning meeting (LAUR # 03-7355). In February 2004, the EC again met with senior LANL personnel to revise the framework, priorities, and

strategies established in previous planning meetings. *Priorities and Strategies* is being revised to reflect the results of the February 2004 planning meeting. Relevance to the LACSI priorities and strategies outlined in the document continues to be a key evaluation criterion used when the EC evaluates progress on LACSI projects.

The EC meets by teleconference bi-monthly and communicates regularly by e-mail. The EC also meets in person at the LACSI Symposium every fall and at the spring planning meeting. In FY04, LACSI EC meetings were held on October 28, 2003 (Santa Fe, NM) and February 19, 2004 (Houston, TX). In FY05, the LACSI EC meetings will be held on October 13, 2004 (Santa Fe, NM) and February 7, 2005 (Los Alamos, NM).

2.1. Management of Academic Subcontracts

Rice is the lead site on the contract for all academic partners, with Ken Kennedy serving as director. Linda Torczon assists him as executive director. Rana Darmara assists him as senior project administrator. Each academic site has a site director: Ken Kennedy (Rice University), Lennart Johnsson (University of Houston), Deepak Kapur (University of New Mexico), Dan Reed (University of North Carolina at Chapel Hill), and Jack Dongarra (University of Tennessee at Knoxville). Each of the following strategic thrust areas has a project director: Ken Kennedy (Components), Rob Fowler (Systems), Yuri Kuznetsov (Computational Science), John Mellor-Crummey (Application and System Performance), and Linda Torczon (Computer Science Community Interaction). Significant decisions related to the management of the academic subcontracts are made by the director with the advice of the academic site directors and the academic project directors.

Due to recent changes in the funding level for the Computational Science research thrust, the Computational Science research thrust currently does not have an academic project director. An academic project director will be chosen when the FY05 funding level for the Computational Science research thrust is finalized. The academic project director will also serve on the LACSI EC.

The LACSI directors, the LACSI executive directors, and key research and administrative personnel from LANL and Rice meet monthly by teleconference to handle administrative matters related to contracts, invoicing, meeting arrangements, reporting requirements, and other administrative issues that arise.

2.2. Computational Resources

The academic partners will be provided with access to ASC computing platforms at LANL on a predetermined basis for development and testing. The process will make it possible to allocate a small cluster of nodes each week and a larger cluster of nodes once a month. It is understood that dedicated access may be needed for key tests and performance analyses.

To the extent possible, development work will be centered on the Clustermatic testbed systems being installed at Rice, UNC, and UNM. Access to large Clustermatic systems at LANL may be required for full-scale tests.