Priorities and Strategies

Los Alamos Computer Science Institute

On March 18-19, 2002 the Los Alamos Computer Science Institute (LACSI) Executive Committee and Principal Investigators met to discuss methods of addressing issues raised in the 2001 LACSI Contract Review. The body was tasked to develop priorities and strategies to meet future programmatic and LANL computer science needs.

A framework was developed to address long-term strategic thrust areas. Specific objectives were called out as near-term priorities. The objectives were folded into the framework to form a coherent planning view. On both April 8-9, 2003 and February 19-20, 2004, the LACSI Executive Committee and Principal Investigators met with senior LANL personnel to revise the framework, priorities, and strategies established at the planning meeting in 2002.

The current framework outlines five strategic thrust areas:

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

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

Components

Ken Kennedy (ken@rice.edu)
Jack Dongarra (dongarra@cs.utk.edu)
Lennart Johnsson (johnsson@cs.uh.edu)
Doug Kothe (dbk@lanl.gov)
Craig Rasmussen (crasmussen@lanl.gov)

The goal of the component architectures effort is to make application development easier through the use of modular codes that integrate powerful components at a high level of abstraction.

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 Los Alamos application developers, and most others in science, cannot afford to sacrifice significant amounts of performance for this clearly useful functionality. Therefore, an important part of the effort 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.

Short-Term Goals

LACSI Component Integration Challenge

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

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.

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)

Long-Term R&D

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

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

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

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

Supporting Technologies for Component Integration

Subproject Leads*:* Ken Kennedy, Rice, ken@cs.rice.edu; Jack Dongarra, Tennessee dongarra@cs.utk.edu

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 the "Modeling of Application and System Performance" section on page 27) 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.

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)

Retargetable High-Performance Components and Libraries

Subproject Leads*:* Jack Dongarra, Tennessee dongarra@cs.utk.edu, Lennart Johnsson, Houston, johnsson@cs.uh.edu, Ken Kennedy, Rice, ken@cs.rice.edu

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

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

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

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 the "Application and System Performance" section, which starts on page 26.

Self-Adapting Numerical Software: We will explore new approaches to building adaptive numerical software that overcome 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.

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 libraries. (Quarter 3)
- Incorporate optimizations into targeted applications; cultivate second round of applications for optimization. (Quarter 4)

Systems

Rod Oldehoeft (rro@lanl.gov)
Rob Fowler (rjf@rice.edu)
Jack Dongarra (dongarra@cs.utk.edu)
Wu-Chun Feng (feng@lanl.gov)
Rich Graham (rlgraham@lanl.gov)
Barney Maccabe (maccabe@cs.unm.edu)
John Mellor-Crummey (johnmc@rice.edu)
Dan Reed (Dan_Reed@unc.edu)
Scott Rixner (rixner@cs.rice.edu)

Overview

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 funding level of LACSI 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.

In each area, three criteria serve as goals and evaluation metrics: performance, reliability, and utility. Since both areas need to competently meet these goals, all three criteria are used to describe the research in both areas.

	Performance	Reliability	Utility
Networking/Messaging			
Clustering			

In addition, each of the resulting six sections that follows is organized into long-term research, and more specific short-term goals that are constrained by available funding and other resources.

Networking/Messaging

By the end of FY05, we expect to see dramatic improvements in the raw capabilities of networking hardware. Initially, the commercial and industrial emphasis will be on the use of this hardware in network infrastructures (backbones) and in commercial servers. These improvements will become available (affordable) in commodity products in the succeeding years. 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.

Networking: Performance

Contributors: Bridges, Feng, Maccabe, Minnich, and Rixner.

Long-Term Goals

Addressing the goal of achieving high network performance with low system overhead in cluster interconnects will require coordinated activities among all of the researchers in LACSI involved in networking. Topics that need to be addressed include:

- Node architectures
- Interfacing the NICS to the nodes
- Protocol design and structure
  - Implementation of protocols for performance
- Assignment of work to the hardware components
  - Assignment of work to NICS (co-processors) to offload CPU
- Zero-copy, zero-map implementations for latency, bandwidth, and efficiency.

Short-Term Activities

System Area Networks/ Cluster Interconnect

From a networking perspective, polling is the most efficient way to minimize latency and maximize bandwidth. However, polling interferes with the application, and as such is undesirable. Furthermore, as the size of the cluster increases the number of memory locations that need to be polled can generally increase. (This depends on the NIC/communications protocol being used.) To eliminate most of this overhead, and so that the data can be available to the application as early as possible, we will investigate an efficient and robust implementation of an event based progress mechanism, and specifically, one that is consistent with MPI semantics. We have already done some initial work on this in the context of a TCP/IP communications layer and shared memory that can take advantage of current kernel level support. We plan to extend this to O/S bypass protocols for which such support is not as well developed.

In addition, we will start to look at moving as much of the messaging processing down to the NIC as makes sense. This presents a couple of challenges:

- Preserving MPI's in-order message delivery while allowing multiple NICs to be used for traffic between two fixed end-points, and

- Data integrity processing when the hardware path from the NIC to main-memory does not handle data corruption (i.e. bus errors). NIC processing is highly desirable as a way to insulate collective communications from slow downs in collective operations due to random OS activities, which become coupled because of the collective nature of the operations. This tends to harm scalability, but also reduces the number of CPU cycles used for communications.

Efficient Zero-copy, Zero-mapped Asynchronous I/O Subsystems

The goal of this research is to investigate the performance tradeoffs of using Ethernet and TCP in cluster computing. Specialized networks, such as Quadrics and Myrinet, are typically used in cluster computing because of their higher bandwidth and lower latency. However, raw Ethernet is extremely competitive both in terms of bandwidth and latency 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, the protocol processing is usually handled directly in the MPI library. By doing so, the transport protocol can be tailored specifically to the cluster-computing domain, which reduces latency, and copying can be minimized by using such techniques as remote DMA.

In TCP, the protocol processing is handled within the operating system, which can be much more efficient. Currently, however, the use of TCP for MPI degrades performance, as TCP is designed to work over the Internet, rather than in the relatively controlled network of a computing cluster. Some of the most significant TCP overheads relate to copying between the application-level and kernel-level on network sends and receives. Although these problems have been solved in the past for network sends, generally applicable and easily programmable zero-copy receives remain an open problem.

Despite these drawbacks, using TCP over Ethernet has several advantages if its performance can be made competitive. First, Ethernet is clearly less expensive than specialized networks. Second, TCP provides reliability and easy portability across systems. Network servers are 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 intend to show that operating system advances for network servers and programmable network interfaces can be used to allow TCP over Ethernet to achieve competitive performance for cluster computing. Specifically, programmable network interfaces can be used as a mechanism for receive copy-avoidance. These network interfaces allow unexpected data to be buffered until they are needed by a receive posted

by the application. Our goals are to encapsulate all the needed changes for high-performance MPI over TCP in the network interface hardware, the network interface firmware, and modified operating system drivers. We aim to avoid any changes to the MPI implementation itself or the application software that are specific to our hardware, thus eliminating the need to continually re-implement MPI for each new class of systems.

To test the effectiveness of these techniques, we intend to add a TCP layer to LA-MPI. This will enable us to understand the bottlenecks related to networking and network interface and how the details of the network interface affect performance. We intend to use that information to develop modified MPI implementations that use TCP efficiently and can be well integrated with the newly designed network interfaces. Furthermore, we are currently implementing a Gigabit Ethernet NIC using the Avnet Virtex II development board, which includes an FPGA and an SO-DIMM slot. This will enable us to show that the bottlenecks encountered by LA-MPI over TCP can be alleviated by intelligent and programmable NICs and by improving the way the MPI library uses TCP. We expect the innovations we propose using our flexible NIC to migrate into mainstream network interfaces.

FY05 Tasks:

- Q1: Participate in the MPI community discussions on the semantics of process failure and MPI, an initial implementation of a process recovery mode.
- Q2: Work on asynchronous progress and moving message processing to communications co-processor (the Network Interface Card (NIC) in most cases).
- Q3: Investigate and experiment with strategies for moving the messaging processing down to the NIC as makes sense in the implementation and hardware layers.
- Q4: Explore elimination of overhead in OS support for messaging whenever possible.

Wide Area Networking

Beyond high performance networking in cluster interconnects, the ASC program is increasingly relying on wide area networking to move data between researchers at their home laboratories and the large shared facilities of the ASC program. To address this, Feng (CCS-1) is working towards applying dynamic right-sizing to the Tri-Lab WAN, thus automatically adapting to bandwidth availability. Part of this effort involves the deployment of a network-monitoring infrastructure (MAGNET+Autopilot).

FY05 tasks:

- Q1: Port dynamic right-sizing (DRS), a technique for dynamic flow-control adaptation from Linux to FreeBSD.
- Q3: Continue to research, test, and benchmark DRS with 10-Gigabit Ethernet in a wide-area network-emulated environment.
- Q4: Analyze and improve how buffer space is managed and used in kernel- based DRS.

Future tasks:

- FY06Q3: Export the DRS technique to use space.
- FY06Q4: Adapt the MAGNET monitoring tool to enable resource-aware visualization and applications in wide-area networks.

Messaging: Reliability and Utility

Contributors: Graham, Dongarra, Reed, Rixner

The general area labeled as "messaging" will address several items in fiscal year 2005. The first quarter will focus primarily on finishing a working version of MPI for the cMPI project. This will include a full implementation of all of MPI 1.2 and most of MPI-2 (less the one-sided communications, for lack of time). This work will include, among others, contributions from the FT-MPI project at the University of Tennessee, and the Resilient Technologies team from LANL. The implementation is really aimed at putting in much of the infrastructure needed for a fault tolerant MPI implementation, as well as a base that can be used to experiment with alternative communication models. It will include the hooks in the run-time system needed for creating new processes, moving processes, and assisting in restoring lost processes. In addition, data fault tolerance, a reworked point-to-point communications design that is better suited for network device fail-over, and the data structures are needed to support MPI recovery from lost processes.

Other work will focus on process fault tolerance and scalability issues. The process fault tolerance work will include participating in the MPI community discussions on the semantics of process failure and MPI, an initial implementation of a process recovery mode. We will continue to quantify failure modes on extreme-scale systems, as a guide to software implementation. The scalability work will focus initially on two areas, asynchronous progress and moving message processing to a communications co-processor (the Network Interface Card (NIC) in most cases). Later scalability work will also encompass the fault tolerant algorithms used, as previously this has involved global coherency state models, which have traditionally affected scalability.

A longer-term issue will be the support of other protocols. For example, ARMCI is emerging as a strong candidate for effective and efficient single-sided communication.

Clustering

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 need be engaged in the effort. It is therefore important that Clustermatic test bed systems be placed at each of the academic institutions. This will expose the academic community to the issues (research, development, and operational) of building and using SSI systems, and it will ensure that software efforts be consistent with mainstream Clustermatic development.

Clustering: Performance

Participants: Minnich, Bridges, Fowler, Maccabe, Reed

Long-Term Issues

The community needs a robust, well-structured, and high-performance software stack to drive high performance network interfaces, *e.g.*, Infiniband, in the context of high performance clusters for scientific computing. Existing open-source software is not suitable. This will require a broad open-source effort.

FY05 Issues

An important goal is to improve the inherent scalable performance of Clustermatic systems. Current Clustermatic systems use a single control node. This can become a bottleneck and, while nodes have been very reliable, it represents a single point of failure. In FY05 there will be an effort to extend Clustermatic to use multiple control nodes. This will include both static and dynamic allocation of compute nodes to control nodes, and it will involve the use of redundant sets of control nodes.

Application performance engineering requires a good performance instrumentation and analysis infrastructure. 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.

FY05 Tasks:

- Add multiple control node capability to Clustermatic (LANL).
- Integrate a pervasive performance instrumentation system such as oprofil with Clustermatic, and layer HPCToolkit on top (Rice).

Clustering: Reliability

Participants Include: ACL staff, Dan Reed, Jack Dongarra

As we noted in the application and system performance section, which starts on page 26, 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 partition allocation and scheduling based on performance and fault models offer the potential to enable long-running programs to react more intelligently. 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 collaboration with performance measurement and modeling activity, we will develop a set of failure indicators, based on monitoring of the "node environment" (i.e., disk and memory errors, temperature, etc). We will also explore how these indicators can be used to differentially schedule applications based on checkpoint needs and expected execution time.

The *supermon* facility needs a redesign. It needs to be able to incorporate user-defined events and to correlate these with system events. The existing information delivery mechanism needs work; the intention is to use V9FS as the information delivery mechanism. This will help in the implementation of a download mechanism for setting and changing configurations. The current sensor interface uses lmsensor, an interpreted language; it is desirable to generate machine code here.

There is a need to address application reliability through support of compiler-driven (assisted) checkpointing mechanisms. An alternative is "buddy" checkpointing. It is also desirable to reconsider the "run-through" concept.

Support for dynamic application reconfiguration is desirable. This is a long-term goal.

As mentioned under the Performance section, above, multiple Bproc master/control nodes on each Clustermatic system are needed for both performance and reliability. A fail-over capability will be part of this.

Experience over the past year indicates that currently compute nodes have turned out to be too reliable to worry about. Efforts at monitoring compute node health are therefore a low priority task this year.

Clustering: Utility

Participants: ACL staff and academic participants using "LACSI standard Clustermatic"

For Clustermatic systems to truly enter the mainstream will require more than addressing the performance and reliability issues of the core system, as discussed previously. It will require adding functionality through the development of system services that are well integrated with the single system image (SSI) model presented by Clustermatic. It will also require the addition of software tools to improve the productivity of users, programmers, and administrators.

During FY05, the main work on improving the Clustermatic software environment will be conducted by CCS-1 (ACL) staff using non-ASC funding. At the same time, LACSI is procuring Clustermatic clusters at several of the academic sites to help build the Clustermatic user community. The results of research and of software development at the academic sites will thus be usable immediately on Clustermatic systems. Furthermore, by introducing developers to Clustermatic, we will be exposing them to the issues of using SSI systems and inspiring solutions to problems that arise. While we expect that there will be some development of Clustermatic infrastructure as a side effect of research and development in other areas, it is our intent that in the future this exposure will lead to specific, funded activities aimed at extending Clustermatic functionality.

The following issues have been identified:

- Improved system administration on Clustermatic systems
    - Work on tools needed to improve administrator/user productivity
- Improvements in the Single System Image
    - Integrated (parallel) global file system
    - Scripting in SSI vs. "pile of workstations" models
- File system semantics
    - Provide private namespaces through the V9 FS. User-mode mounts add to the namespace
    - Improve the functionality of private namespaces so additions can be made during execution and those additions propagated to all node namespaces
- Extend the BJS fast scheduling facility with familiar user interfaces, *e.g.*, the LSF API
- Users need ability to tailor runtime environments as needed
    - "Modules" brought into the file system namespace
    - Modules now set up environment variables; they need to set up the whole environment

Computational Science

Doug Kothe (dbk@lanl.gov)
Beth Wingate (wingate@lanl.gov)
Bill Symes (symes@rice.edu)
Dan Sorensen (sorensen@rice.edu)
Mike Fagan (mfagan@rice.edu)
Lennart Johnsson (johnsson@cs.uh.edu)
Deepak Kapur (kapur@cs.unm.edu)

Overview

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. Continuum Dynamics, Energy Transport, and Materials Science;
2. Multi-Physics Coupling; 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 for the resulting physical models, new algorithms that possess good single processor performance and are 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 Goals

In the short term (< five years), the *Computational Science* effort must complement the LANL Computational Sciences effort, including the LANL ASC Computational Sciences Program Element (CompSci PE). As one of eight PEs within the LANL ASC Program, the principle mission of CompSci PE is to deploy verified and validated software components embodying shock hydrodynamics, radiative and neutron transport, and linear/nonlinear solvers. It must also deliver simulation tools for weapons performance (the Marmot Project) and weapons casting and welding processes (the Telluride Project). A notable short-term goal of the LACSI Computational Science effort is to deliver software components to the three critical ASC "weapons performance code projects", known collectively as the Crestone, Shavano, and Marmot Projects. Short-term goals for the computational science effort include:

- Development of a hybrid Monte Carlo deterministic transport capability
- Simulation of casting of the Qual Type 126 pit and comparison of the results of the simulation to the available experimental data for the same process
- Development of a Capsaicin Project transport capability to the Marmot Project (Capsaicin is a software project for a verified deterministic transport capability.)
- Development of an interface tracking interface component to the Crestone Project.

These short-term goals feed into and provide enabling technology for other, higher-level milestones within the LANL ASC Program. These technologies will also form the core constituents of next-generation LANL weapons performance and manufacturing simulation tools.

Research Thrust Areas

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

Investigators: Yu. Kuznetsov (UH), J. Morel (CCS-2), G. Olson (CCS-4), and M. Shashkov (T-7)

Efficient numerical methods for the diffusion and radiation transport equations in highly heterogeneous media on general distorted polyhedral meshes 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, Yuri 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.

Short-Term (FY 2005):

- To implement the proposed polyhedral discretization method and to evaluate its accuracy and efficiency on 3D test problems relevant to ASC applications.
- 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 posteriori error estimators for the polygonal discretizations of the diffusion equations and to evaluate it on test problems relevant to ASC applications.
- 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.

Long-Term:

- 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.

Computational Methods for Free Boundary Problems: Application to Telluride

Investigators: R. Glowinski (UH), D. Kothe (LANL), S. Poole (LANL), A. Cabossat (UH Research assistant), G. Guidoboni (UH Research assistant)

Telluride is an important LANL research project. It includes, among other components, the numerical simulation of *melting* and *free surface* phenomena. The role of the UH *free surface group* will be to investigate the implementation in the Telluride framework of finite element and operator-splitting-based solution methods developed at University of Houston and the Swiss Federal Institute of Technology in Lausanne by R. Glowinski and A. Cabossat; indeed the VOF method developed at LANL is one of the ingredients used by Cabossat for the simulation of melted aluminum flow. The finite element to finite volume conversion of the above methodology will be investigated and its possible parallelization as well.

Short-Term Goals:

- Identify significant two-dimensional free surface test problems.
- Develop finite-element-based solution methods.
- Investigate the finite element to finite volume partial or total conversion of these methods.

Long-Term Goals:

- Generalization to 3-D phenomena. One of the major difficulties in this direction is the accurate computation of the *mean curvature* of the free surface in order to evaluate the surface tension forces. This problem is well understood in 2-D, where the interface is a curve, but it is quite a challenge in 3-D.

Software Design for Coupled Simulation and Optimization

Investigator: Bill Symes (Rice)

Simulation driven optimization (SDO) is the core computational task in numerous important technologies bearing on the LACSI mission, such as seismic risk analysis, meteorological and oceanographic data assimilation, control of fluid flow in ducts and channels, and design of castings and other manufactured items. Its salient characteristic is the coupling of complex simulations with optimization software, bridging a wide variety of abstraction levels. Conventional (procedural) programming of SDO applications almost unavoidably transgresses these abstraction levels, bringing, for example, the data structures of simulation into the optimization code, and vice versa, leading to code that is difficult to maintain and usually impossible to reuse outside of its originating context.

DoE and academic researchers have recently turned to modern software techniques, including object and component orientation, to create reusable, maintainable, and extensible code libraries for the SDO problem domain. An earlier project of this group, the Hilbert Class Library, has served as a prototype for the Trilinos Solver Framework (TSF), a major DoE SDO library. Interoperability is a critical issue in the construction of libraries of this type, as all tend to implement the same abstractions incompatibly. We have demonstrated interoperation of our current project, the Standard Vector Library (SVL), with TSF and other DoE solver libraries, and identified design principles that enable efficient interoperation. Current SVL-based projects include AlgPack, a framework for iterative algorithm construction, and TSOpt, a timestepping library largely automating implementation of adjoint state computations. TSOpt is specifically designed to ease application of Automatic Differentiation to SDO problems.

SVL's design is intrinsically compatible with client-server applications, and we have demonstrated parallel servers. A very important immediate goal is construction of SVL-compatible interfaces for DoE distributed computing libraries, so that existing simulators may be easily ported into SVL-based SDO applications.

Short-Term Goals:

- Parallel server design using DoE libraries
- Implementation of time-dependent flow control in parallel
- Identification of LACSI-focused application

Long-Term Goals:

- Integration of SVL into Trilinos, either as a package or as a set of ideas; dissemination within labs
- Collaboration with Telescoping Languages project to enable cross-method optimizations, including loop fusion

Numerical Linear Algebra for Large Systems

Investigators: D.C. Sorensen (Rice), Nadiga (CCS-4), Jim Morel (CCS-4), Rob Lowrie (CCS-2), Dana Knoll (T-3), John Turner (CCS-2), and Beth Wingate (CCS-2)

This project is concerned with the development of methods and software for large eigenvalue problems and related applications. The project will extend the capabilities of the highly successful P_ARPACK eigenvalue software. This software executes on massively parallel systems and provides enabling technology in numerous application areas. It also can serve as an excellent test bed and point of interaction for new compiler technology.

In particular, we are interested in subspace projection techniques for the development of dimension (order) reduction methods in dynamical systems and control. We have been developing projection methods that are closely related to the Krylov projection techniques used in large eigenvalue computations.

Dimension Reduction Software as a Test Bed for the Telescoping Language Project

In support of the telescoping languages project, we are developing a collection of software for model order reduction of dynamical systems in Matlab. Model reduction seeks to replace a large-scale system of differential or difference equations by a system of substantially lower dimension that, ideally, has the same response characteristics as the original system, yet requires far fewer computational resources for realization. Such large-scale systems arise in circuit simulation; they also arise through spatial discretization of certain time dependent PDE control systems. Our work is focused on the development, analysis, and implementation of reduction methods for very large problems, control problems in particular.

This software will be an important contribution in its own right, but our intent with this work is to develop the software in a style that is amenable to the telescoping language approach. It should serve as a prototype and as a test bed for the development of compiler technologies and library designs that will make it possible to automatically construct domain-specific development environments for high-performance applications. A goal of the project is demonstrate the feasibility of developing and maintaining an application library written in Matlab that can be used to automatically generate a platform specific optimized version. Such a demonstration would validate the potential of the telescoping language approach.

This project builds upon and extends the applicability of our prior work on large-scale eigenvalue methods. We have developed reduction schemes based upon Krylov projection and also upon gramian-based methods for dimension reduction. Our codes will include methods for approximate balanced reduction of large-scale linear dynamical systems. It will also include methods for structure preserving reduction. Reduction of second order systems that preserves the second order form and reduction of passive systems that preserves passivity are two important examples.

Consultation on the Numerical Solution of Large-Scale Eigenvalue Problems

Our past work on eigenvalue methods and software are somewhat mature at this point. Nevertheless, there are still important research topics to investigate. Most notably, the development of techniques for accelerating convergence, such as approximate shift-invert preconditioners, is still an active research area.

Within the scope of this project, the focus will be less in the development direction and more in the consultation arena. Some areas that might benefit by this activity are:

- *Linear and Nonlinear Solvers*

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

We believe one can use eigenanalysis to help design improved linear system preconditioners that are adaptive. These preconditioners could be integrated into the design of a Newton-Krylov solver. One approach is to introduce eigenvalue deflation techniques as a way to mitigate restarted GMRES convergence problems.

- *Eigenanalysis in Transport*

  The k-eigenvalue estimations utilized in neutron transport would be an interesting application of our eigenvalue software. It would be interesting to obtain from Jim Warsa sample linear systems arising in neutron transport k-eigenvalue calculations (performed by LANL's Attila software).

If the need for consultation or collaboration arises in these areas, we will address those needs.

Short-Term Goals:

- Develop an implementation of a Modified Multi-shift Smith Method for approximate solution of large Lyapunov equations.
- Develop a code for balanced reduction based on the Modified Smith Method.
- Collaborate with the Telescoping Languages project on refining the structure of these codes.

Long-Term Goals:

- Develop a complete collection of model reduction software in a style that is compatible with the constructs required by the Telescoping Language project.
- Collaboration with Telescoping Languages project to enable cross-method optimizations, including loop fusion.

Code-Based Sensitivity Analysis

Investigators: M. Fagan (Rice), R. Henninger (CCS-2), Ken Hanson (CCS-2), Jim Sicilian (CCS-2), John Turner (CCS-2), Ralph Nelson (X4)

One of the major priorities of the Los Alamos mission is to validate and verify ("V & V") the highly complicated computer programs used to model equally complicated physical processes. Moreover, one of the major tools in the verification and validation process is sensitivity calculation. Consequently, the overall priority of the code-based sensitivity analysis project is to develop methods for accurately and efficiently computing sensitivities of complex scientific simulation programs.

Short-Term Goals:

The short-term goals for the code-based sensitivity effort are related to the following projects at Los Alamos: The Telluride Project directed by Jim Sicilian (technical point-of-contact is Rudy Henninger), the Shavano Project (POC Ralph Nelson), and the Marmot Project. The short-term goals are:

- Assist in the application of Adifor to current Fortran 77 codes throughout LANL.
- Continue to extend the Adifor technology to cover more of Fortran 90, so that accurate sensitivity calculations can be easily generated for the Truchas code of the Telluride Project.
- Ensure that Adifor technology may be applied to the extended Fortran 77, Fortran 90, and C mechanisms used in FLAG (and FLAG-like) codes of the Shavano Project. The main thrust of this effort is to provide seamless, integrated AD across commonly used languages.
- Ensure that Automatic Differentiation-based methods for estimating roundoff error in simulation codes are available. In particular, roundoff error estimation should be part of the unit testing for codes (like Marmot) that are under development.

Long-Term Goals:

- Applying automatic differentiation for verification of ODE-based and PDE-based computer models.
- Applying automatic differentiation in the construction of Newton-Krylov solvers. Newton-Krylov solvers need directional derivatives. Typically, these directional derivatives are computed using a matrix-free finite difference method. Our proposed alternative is to use automatic differentiation to compute the directional derivatives.
- Applying AD to programmatic right-hand sides in the context of verification by the method of manufactured solutions (MMS).
- Developing ATOM-style automatic differentiation for object code.
- Extending Adifor-style automatic differentiation to additional programming languages, notably Fortran 2000 (when it becomes available), Java, C, and C++. The CartaBlanca project, in particular, has expressed interest in Java.
- Supplying automatic differentiation for scripting languages such as Python (and possibly Perl and Ruby). At this point, the question of Adifor-style AD versus some other method is not clear.
- Extending the augmentation paradigm to include other sensitivity measures such as intervals or probability distributions.

Application and System Performance

Adolfy Hoisie (hoisie@lanl.gov)
John Mellor-Crummey (johnmc@rice.edu)
Keith Cooper (keith@rice.edu)
Jack Dongarra (dongarra@cs.utk.edu)
Robert Fowler (rjf@rice.edu)
Guohua Jin (jin@rice.edu)
Dan Reed (Dan_Reed@unc.edu)
Linda Torczon (linda@rice.edu)

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 performance research thrust are:

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

A broad spectrum of issues affects application performance, 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.

Modeling of Application and System Performance

Investigators: Adolfy Hoisie, John Mellor-Crummey and Robert Fowler

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. As a result, accurate models such as the ones developed by PAL at Los Alamos are the performance tool of choice in gaining insight into the performance of applications and systems.

The research in this area will span a wide range of topics. First, we will continue the application modeling work so that all important types of computations (and their associated application software) are accurately modeled. The next step in this endeavor is modeling of non-deterministic applications such as Monte-Carlo transport.

Second, a major thrust will be in the enhancement of the models to include detailed effects of the operating system, architectural features, and system activity. Based on a novel methodology developed by PAL (which led to major performance improvements on the ASCI Q machine) we will include this capability directly in our application models.

Third, we will look into using our models to do application steering, complementing other proposed steering approaches being explored by Rice and UNC. For this to be possible, models will have to become dynamic and incorporate runtime information from the hardware performance monitors and other sources (e.g., NICs) as the application executes.

Fourth, we will undertake the task of trying to simplify model creation. We will attempt to provide a mechanism for: aiding model creation, enabling model description, undertaking model evaluation, and allowing for model incorporation into code. These capabilities will provide a “tool” basis for performance modeling, easing their creation and use, while at the same time allowing performance models to accumulate within a

coherent structure rather than having a sequence of one-off studies. The focus of this aspect of the research will be on designing, building and evaluating semi-automatic tools for synthesizing models and model components, as well as on exploring how to integrate model components synthesized automatically into hand-crafted model frameworks.

Fifth, we will concentrate on the numerous applications of the models we developed. The resulting performance models 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 a valuable tool for architecture design. We plan to apply models of the ASC workload to propose and design advanced architectures that maximize the performance of this workload.

Short-Term Tasks:

- Continue to work by LANL PAL team on analysis and modeling of applications in the ASC workload. That work will be broadened to include other ASC types of computations such as non-deterministic applications.
- Continue the PAL team's active research effort in expanding the models to include an accurate account of system effects.
- Further enhance LANL's performance modeling methodology by expanding the research into building tools designed to simplify the modeling methodology.
- Evaluate relative accuracy of cross-architecture predictions of node performance for different architectures.
- Refine capabilities for modeling and prediction of an application's memory hierarchy performance for a range of architectures for different problem sizes.
- Explore combining single-processor/node modeling efforts of Rice and LANL with the goal of creating a predictive single processor performance capability applicable to ASC applications and microprocessors of interest to the program.

Building applications that can effectively exploit extreme-scale parallel systems has proven very difficult: Massive parallelism poses major challenges in achieving scalable performance; architectural complexity (complex processors, deep memory hierarchies, heterogeneous interconnects) is difficult for applications to exploit effectively; at this scale component failure is inevitable, so long-running applications must be resilient in order to run to completion.

Better Tools for Measurement and Analysis of Application Performance

Investigators: Robert Fowler, John Mellor-Crummey and 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, or binary modification before or 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.

Short-Term 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.
- Explore using statistical clustering of node performance measurements of parallel applications to reduce the complexity of analyzing programs with large-scale parallelism.
- Investigate statistical sampling for extreme-scale systems, emphasizing tunable control of measurement accuracy and overhead; integrate ideas and software for sampling with the SvPablo toolkit.
- 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.
- 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.
- 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. These workshops serve three purposes. First, they help application teams use tools developed by the academic members of LACSI to understand how their choices of data structures and algorithms affect performance. Second, they provide an opportunity for cross-disciplinary working groups to examine ASC workload exemplars and exchange ideas. Third, they provide valuable feedback to the performance team about opportunities for enhancing tool capabilities.

Automatic Application Tuning

Investigators: Keith Cooper, Ken Kennedy, John Mellor-Crummey and 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 described previously, 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.

Short-Term 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.
- 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. We will experiment with adaptive compilation sequences in Microsoft's new Phoenix compiler infrastructure.

- Explore adaptive techniques for control of an individual optimization. We will complete our study of adaptive control of source-to-source inline substitution. We will release our prototype adaptive inliner as a standalone tool.
- 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.

Compiler Technology for Exploiting Modern Processors

Investigators: Keith Cooper, Ken Kennedy, John Mellor-Crummey, and 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.

- 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.
- 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.
- Finally, 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.

Short-Term Tasks:

- Continue our experiment with automatic choice of command-line parameters for IA-64 compilers (both ORC and the Intel compiler). As a first step, we will quantify the potential improvement from manipulating parameter settings. As a second step, 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. We will release new (and improved) register allocators for LLVM in source-code form. We will begin development of an advanced scheduler for LLVM.
- 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 and develop reassociation strategies to reduce the operation count.

Application Mapping, Dynamic Adaptation and Steering

Investigators: Dan Reed, Ken Kennedy and John Mellor-Crummey

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. 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 to 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 will explore using 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.

Short-Term Tasks:

- Explore techniques for measuring environmental conditions on the nodes of extreme-scale systems, as a basis for guiding scheduling and resource partitioning decisions.
- Investigate extended partition allocation and scheduling based on environmental predictors, including failure probabilities, for integration with batch schedulers
- Demonstrate adaptation techniques for multi-attribute system behavior, including power management.

Compiler Technology for Extreme-scale Systems

Investigator: John Mellor-Crummey

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. 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.

Long-Term Tasks:

- Explore data-parallel language constructs that make it simple to express a wide range of parallel programs at an appropriate level of abstraction.
- Evaluate the expressiveness of these languages for problems of interest to ASC and the broader high-performance computing community.
- Devise effective code generation techniques for these languages with an eye towards the full range of future parallel systems, namely very tightly-coupled extreme-scale systems envisioned in DARPA's HPCS program, moderately tightly-coupled systems composed of commodity processors with high-performance interconnects (e.g. Red Storm and Blue Gene/L), as well as more loosely coupled commodity clusters.
- Evaluate the how well code generation strategies deliver high performance.
- Understand the interplay between language features, application structuring techniques, code generation challenges and application performance.

Short-Term Tasks:

Global address space languages such as UPC and CAF are relatively immature, as is compiler technology to support them. Research on UPC and CAF will focus on the following issues:

- *Language primitives.* UPC and CAF need refinement so that they can conveniently and efficiently express a wide range of programs and parallelization schemes in a portable fashion. Both need support for collective communication and perhaps atomic operations. CAF needs primitives for split-phase synchronization and locks. Language support for hierarchical locality domains may be useful for large-scale NUMA systems. Augment synchronization primitives with optional tags to show how synchronizations may match at runtime.

- *New analysis and optimizations for SPMD programs.* Optimizations that transform or combine synchronization and data movement (e.g., synchronization strength reduction or combining a put with a notify) will boost performance. For UPC, more effective analysis and transformation of code using global pointers is needed.
- *Compiler support for latency tolerance.* Today's parallel programs often communicate or perform I/O in synchronous bursts, stalling computation until the operation completes. Compiler-directed latency hiding (transforming synchronous communication and I/O into asynchronous form and overlapping them with computation) is needed to increase code efficiency.
- *Performance portability.* For SPMD language models to be widely adopted, they must be suited for high performance on lower-cost clusters as well as tightly coupled HPCS systems. Satisfying this goal will require research and development of code generation algorithms for tailoring code to architectures with a broad range of different communication latency, bandwidth and granularity characteristics.
- *Runtime mechanisms.* A variety of issues will be explored to optimize execution efficiency including communication mechanisms (e.g. fine-grained direct load/store communication) and using virtualization to improve load balance and latency tolerance.

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. 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.

- Explore algorithms for effectively partitioning computation in the presence of complex data partitionings and dependence patterns.
- Continue to investigate analysis and code generation strategies with the aim of improving node performance of code for sophisticated algorithms such as multigrid.
- Explore the implications of multiple levels of parallelism, e.g. combining task and data parallelism within an application.

Computer Science Community Interaction

Linda Torczon (linda@rice.edu)
Jack Dongarra (dongarra@cs.utk.edu)
Rob Fowler (rjf@rice.edu)
Lennart Johnsson (johnsson@cs.uh.edu)
Deepak Kapur (kapur@cs.unm.edu)
Ken Kennedy (ken@cs.lanl.gov)
Rod Oldehoeft (rro@lanl.gov)
Dan Reed (Dan_Reed@unc.edu)

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

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

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

We will also coordinate a technical infrastructure between LANL and the academic partners, enabling web broadcasting of local technical talks, workshops, and the annual symposium to an off-site audience.