Los Alamos Computer Science Institute

FY 2006 Proposal

The principal goal of the *Los Alamos Computer Science Institute (LACSI)* is to conduct basic research in computer and computational science that has relevance to the mission of Los Alamos National Laboratory. The funding is not intended to support advanced development efforts, although some such efforts may be spun out to produce software and tools that will be directly useful to end users. Secondary goals of the Institute are to increase the national and international stature of LANL computer science and to engage the high-performance computing research community in problems of importance to the LANL.

This proposal is the product of a year-long planning effort that began in Spring 2004 with the annual LACSI Priorities and Strategies Meeting, in which plans for FY05 activities were developed. As a result of discussions at that meeting, we resolved to take two actions:

1) We would move to a process that included an annual review of activities by an independent review committee that included members from within DOE as well as industry and academia. The first such review was convened in November 2004.

2) Based on the review and the annual planning meeting, LACSI would submit a new proposal for research funding each year. The proposal would cover three years of activity, even though the funding would be provided one year at a time. The review and planning process should include mechanisms for phasing out projects and initiating new starts.

This proposal is the first produced by the new process. For reference, we have attached the report of the review committee from November 2004.

For FY06, LACSI activities would cover a broad spectrum of areas of research, including strategies for measuring, analyzing, and improving application and system performance; components and component integration systems; computer systems and networking; and computational science. These research areas are covered in four subsequent sections of the proposal. Every section includes two subsections, each describing a coherent, coordinated research activity, along with the budget, a set of tasks to be carried out, and a discussion of the relevance of the proposed work to the Weapons Program.

In addition to the research, there is an additional section that covers management and community interaction. The management section includes a detailed discussion of the new management and planning cycle for LACSI. Community interactions include the enormously successful LACSI Annual Symposium and several other outreach programs carried out by the Institute. Finally, there is a section summarizing the budget

Taken together, the proposed LACSI activities represent a broad and deep collection of projects that can have an enormous positive impact on high performance computing and communications within the DOE Weapons Program.

1. Application and System Performance

1.1 Integrated Performance and Reliability of Extreme-scale Systems

Supported Personnel:

Project Leads: John Mellor-Crummey, Rice, johnmc@cs.rice.edu; Jack Dongarra, UTK, dongarra@cs.utk.edu; Daniel Reed, UNC, Dan_Reed@unc.edu, Patrick Bridges, UNM, bridges@cs.unm.edu

Other Supported Personnel: *Rice*: Rob Fowler

LANL Point of Contact: Adolfy Hoisie, hoisie@lanl.gov

1.1.1 Vision and Motivation

Building scientific applications that can exploit extreme-scale parallel systems effectively is difficult. The goal of research on performance and reliability of large-scale systems is to develop tools and technologies that help application scientists harness the power of extreme-scale parallel systems for solving computation-intensive problems.

The level of parallelism in extreme-scale systems and the need to exploit parallelism on multiple levels poses a formidable challenge to achieving scalable performance. Furthermore, the architectural complexity of extreme-scale systems makes it hard to write programs that can fully exploit their capabilities. Today's extreme-scale systems include complex processors, deep memory hierarchies and complex heterogeneous interconnects. 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, large communication bandwidth requirements and ineffective network utilization. To achieve a significant fraction of a system's potential performance requires careful scheduling of an application's operations, data accesses, and communication.

On extreme-scale systems, performance and reliability are inseparable. The large number of components in extreme-scale parallel systems makes component failure inevitable; therefore, long-running computation-intensive applications must be resilient to transient and permanent hardware faults or risk being unable to run to completion. The most popular parallel programming paradigm, MPI, has little support for reliability. When a node fails, all MPI processes are killed and the user loses all computation performed since the last application-initiated checkpoint.

Many challenging problems must be solved to understand how to implement scientific applications so that they can achieve scalable high performance and are resilient to failure on extreme-scale parallel systems. We propose a program of research that aims to develop technologies that support measuring, analyzing, modeling, and improving the performance and reliability of applications on current and future generations of extreme-scale parallel architectures. The principal aims of this research effort are to

- understand application and system performance and failure modes on present-day extreme-scale architectures through the development and application of technologies for measurement and analysis of program and system behavior,
- model the behavior of applications to understand factors affecting their performance and reliability on current and future generations of computer systems,
- devise software strategies to ameliorate application performance bottlenecks and failure susceptibility on today's architectures, and
- explore technologies to support fault tolerance, including capabilities for monitoring the health of system components, predicting impending faults, and proactively avoiding faults.

This work will address aspects of performance and reliability spanning both applications and underlying hardware.

1.1.2 Research Plan

To address the problem of developing applications for extreme-scale systems that deliver high performance and are resilient to failure, we propose research in the areas of measuring performance and component health, analyzing performance and reliability, modeling application and system performance and reliability, exploring techniques for constructing failure-resilient programs, and evaluating alternative architectures that can yield increased performance and greater reliability.

Measuring Performance and Reliability

Extreme-scale systems require monitoring for a variety of purposes. Performance monitoring is needed to evaluate program behavior and resource utilization to understand why and where applications are inefficient. Health monitoring of status information such as temperature and power consumption is necessary on extreme-scale systems to anticipate or detect failures, which are inevitable on such systems because of their scale.

For measurement tools to be useful for monitoring production programs on extreme-scale systems, they must

- have low overhead that does not distort application performance or system behavior,
- work with multi-lingual, parallel, optimized code without modification,
- analyze not only user programs but their interplay with binary-only libraries, the operating system kernel and system processes,
- support programs with large-scale, possibly heterogeneous parallelism,
- record data of only modest size, and
- scale to tens of thousands of nodes.

In general, performance problems on extreme-scale systems are varied and complex. To understand the performance of parallel applications, one must capture detailed information about their behavior, including the interplay of computation, consumption of and contention for resources (e.g., network and I/O), communication, synchronization, and serialization. The key research challenge is to devise effective strategies for integrated measurement of such aspects of program behavior. These measurements must be sufficient to provide insights into the interplay between these different aspects of program behavior and yet introduce only low monitoring overhead so they can be used on production runs. For performance measurements to be most useful to application developers, it must be possible to correlate them with the application's source code.

The appropriate strategy for collecting performance measurements depends upon the aspect of program behavior under study and the intended uses of the performance data. Performance data for an application may be collected for off-line analysis or it may be collected to enable an application to evaluate its own efficiency and adapt its behavior in response to its findings. Key strategies for gathering performance data include statistical sampling of events (sampling may be either system wide or within individual application processes), using instrumented communication libraries, and inserting instrumentation into programs via source code transformations, or by rewriting application and library binaries at link time, program launch, or during execution.

Measuring and Attributing Node Performance

For modern modular programs, knowing the context in which costs were incurred is vital to understanding how to address performance problems. For instance, rather than knowing that a program spends time in a particular solver, it is much more informative to know that when the solver is called with

preconditioned matrices, it is fast; otherwise, it is much slower. Similarly, for parallel programs that invoke communication operations in many places, knowing that the program spends a lot of time waiting for communication is not particularly helpful for pinpointing the problem. In contrast, knowing that most waiting time is associated with a particular communication event is the first step to determining whether load imbalance, serialization or communication frequency is to blame.

To attribute costs precisely to execution contexts within programs, we propose to design, build, evaluate, and disseminate open-source compiler and runtime mechanisms for efficiently collecting calling context for both synchronous and asynchronous events. Synchronous events include memory allocation and communication. Asynchronous events include hardware performance counter overflows and timer expiration. We need to collect information about both classes of events during the execution of *optimized* code. Rather than instrumenting each function to log its entry and exit so that calling context information is always available (as `gprof` does), we collect calling context on demand by stack unwinding. Collecting calling context efficiently for optimized code is tricky. In particular, for asynchronous events, monitoring code must be able to unwind the call stack from *any* point in the program execution, even procedure prologues or epilogues. For this case, compiler support or binary analysis is needed to determine exactly where in the machine state the return address of the caller resides. This work will build on prior work at Rice on AlphaServer platforms, which lacked sufficient compiler support to fully realize this vision, and ongoing work on Opteron platforms for which we are prototyping necessary compiler support in the `gcc` compiler infrastructure.

To date, our research on measuring application node performance has focused on static parallelism of threads and processes. Programming models and architectures under development as part of DARPA's HPCS project heavily rely on dynamic parallelism. We propose to explore strategies for understanding system performance for applications and systems making use of dynamic parallelism.

Measuring Performance Beyond Processors

The PAPI library has standardized access to hardware event counters available on most modern microprocessors by providing standard routines and a standard set of performance metrics. Counters on processors measure events such as cycle and operation counts, functional unit status, cache and memory accesses, and branch behavior. Other components besides the processor, such as memory chips, network interface cards, and network switches, also have counters that monitor various events related to system performance and reliability. Unlike on-processor counters, off-processor counters usually measure events system wide rather than for a process or thread-specific context. However, when an application has exclusive use of a machine partition, it may be possible to interpret such counts in the context of the application. It may also be possible to correlate process-specific events counts with system-wide counts using a "data mining" approach based on multivariate statistical analysis.

The current situation with off-processor counters is similar to the situation that existed with on-processor counters before PAPI. There are many platform-specific interfaces, some of which are undocumented. We propose to develop standard routines for accessing off-processor counters on a range of platforms and networks. We have begun working on an interface to counters available on Myricom network switches. The Infiniband standard also has definitions of measurement hooks. The Cray X1 has E-chip and M-chip counters. The challenge will be to try to develop as portable an interface as possible while still providing access to information that is necessarily somewhat platform-specific. In addition to routines to control and access counters, we plan to provide query routines that will provide information about what counters and events are available, including both standardized and platform-specific events.

Measuring Communication

To understand the performance of parallel applications on extreme-scale systems, one must understand the end-to-end costs of communication and delays that result from load imbalance or serialization. For this purpose, we have begun to explore a message-centric monitoring approach that involves gathering

and propagating information as messages traverse the system. Using this approach, each message is augmented with a monitoring summary that compactly encodes information related to the message, e.g., how much time was spent generating the data contained in the message. Message-centric monitoring trades the centralized global view of trace-based systems for a decentralized, weakly consistent view of system behavior to approach the low overheads and online availability provided by statistical monitoring techniques. Making this monitoring approach efficient will require developing low-overhead representations of performance and reliability data, and controlling monitoring overhead by a combination of sampling and adaptive monitoring.

Thus far, we have separately explored two strategies for message-centric monitoring: vertical monitoring of message costs on a single node, and horizontal monitoring that combines monitoring summaries from multiple incoming messages, creates a new monitoring summary from this data, and attaches it to outgoing messages. Our current prototypes show that we can gather vertical monitoring information on Linux cluster nodes using a modified version of the Linux Trace Toolkit with 3% overhead for the NAS parallel benchmarks, and that we can efficiently propagate horizontal monitoring information about LA-MPI runtime performance with similar overheads. We now need to integrate these prototypes, developed at UNM, into a single system for monitoring all operating system and message passing costs for a large-scale parallel scientific application, integrate them will the application-level monitoring tools developed at Rice, and evaluate their accuracy and overhead on large-scale machines with the ASCI Purple benchmarks.

We propose to explore a sample-based approach to message-centric monitoring to enable tunable control of measurement accuracy and instrumentation overhead. This would enable message-centric monitoring to be used on production applications and system software. If message-centric monitoring is available for production applications, it can be used to audit them to verify that their behavior matches the characteristics of previously gathered performance profiles. Such audits could be used to detect performance anomalies caused by new data sets or system software changes. In addition, online availability of message-centric monitoring could also be used to optimize application load-balancing decisions and to drive MPI protocol adaptations to improve application performance.

Measuring Node and System Health and Reliability

Component failure is inevitable on extreme-scale systems. Long-running jobs must expect to encounter faults and take appropriate action to enable systems and applications to continue operating even when some components fail. Without the ability to anticipate or notice and tolerate component failure, the mean time to failure (MTBF) of systems with tens of thousands of commodity components, even when operated as quasi-independent partitions, will continue to decline and will severely limit the ability to solve complex problems. To anticipate component failures, we propose to develop an extended health monitoring infrastructure and support for assessing the thermal environment of extreme-scale systems.

During the past year, Reed's team at UNC has developed a health monitoring system library and a Health Application Programming Interface (HAPI) for discovery and use of health-related diagnostic information on Linux clusters. HAPI acquires fault indicator data via (a) vendor provided Self-Monitoring Analysis and Reporting Technology (SMART) for disk warnings such as seek and read/write retries; (b) Advanced Configuration and Power Interface (ACPI) for temperatures and CPU throttle rates and (c) low-level hardware sensors for motherboard temperatures and power supply voltages. By operating as a uniform software layer that sits above low-level health measurement tools, HAPI hides the idiosyncrasies of measurement and allows fault detection, prediction and recovery techniques to access data via standard mechanisms. HAPI supports both standard Linux clusters and the LANL Clustermatic toolkit.

To enable effective monitoring of systems containing thousands or tens of thousands of nodes, fault tolerance data collection must be scalable and integrated with low overhead performance measurement systems. We propose to develop statistical sampling schemes for this purpose. These sampling schemes

will be adaptive, enabling configuration of sampling frequencies and sample sizes based on historical failure probabilities and user experiences.

Measuring Environmental Health and Impacts

Thermal stresses are also a major cause of component failures. An enterprise machine room contains a complex hierarchy of "micro-climates," ranging from within a single microprocessor, a motherboard and node, and a set of racks through raised floors, cooling ducts and ambient room temperatures. Often, physical configuration and placement can lead to unexpected and unknown (at least until failures occur) thermal stresses.

Building on the UNC HAPI infrastructure, we propose to develop a distributed measurement toolkit that exploits mobile environmental sensors that can be placed in machines, racks and air ducts. This toolkit will capture time varying temperature, humidity and vibration, enabling construction of machine room micro-climate maps and displays. This toolkit will be integrated with the health monitoring infrastructure and provide the raw data needed for constructing more accurate hardware failure models. In addition, it will enable more intelligent management of machine rooms by tracking the effects of hardware changes on the surrounding systems; often placement of new equipment can create thermal hotspots where none were expected.

Analyzing and Modeling Performance and Reliability

The data collected for a parallel application on an extreme-scale system can be overwhelming. Analysis and presentation techniques must support top-down analysis to cope with the complexity of programs containing millions of source lines running on thousands or tens of thousands of processors. To understand executions and reliability on such systems, it is not practical to inspect processors and tasks individually. Instead, improved analysis techniques are needed that can identify and correlate data from key subsets.

Analyzing Performance with Compile Time Information

Understanding the performance of programs with thousands of "moving parts" requires understanding the relationship of interactions between processes to the context in which these interactions occur. Resource contention (e.g. for access to I/O), serialization, and load imbalance are three problems that are difficult to understand. Serialization needs to be attributed to synchronization points. Resource utilization and contention must be attributed back to resource requests in context. Even understanding computation costs in modular programs can be difficult when multiple instantiations of code are present because of inlining and templates. In each case, the first step to understanding behavior is assessing the context in which it occurs. We propose to use calling context information collected with call stack unwinding as the basis for attributing costs to contexts. For programs with inlining or template instantiation, it is important to identify and attribute performance metrics correctly to inlined abstractions as well as the contexts into which they are inlined. To address this problem, we propose to extend capabilities in Rice HPCToolkit for binary analysis of executables to identify and accurately attribute performance for inlined code.

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, on an extreme-scale system, an application developer cannot inspect data from every application or system component. 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.

Better statistical techniques are needed for analyzing performance data and for understanding the causes and effects of differences among process performance. We have begun investigating bi-clustering techniques for identifying key differences between groups of processes. A second technique for coping with the number of components and the amount of performance data that can be collected on extreme-scale systems is to select a statistically valid subset of the components and model the members of that subset in detail. Properties of the subset can be used as a basis for estimates of the entire system. Our preliminary research in this area, so far, has explored using statistical techniques for analyzing system availability. The main objective of this research will be to evaluate how well application performance can be characterized and understood without exhaustive examination of performance data by developers.

An analysis challenge using message-centric monitoring is to account for indirect second-order performance effects without complete global information. Trace-based systems are able to find such indirect performance effects by paying the cost associated with keeping a global record of system activities across all nodes. Systems based purely on statistical monitoring cannot normally find such problems because there is no causal data and no cross-node data available. Unlike current statistical approaches, message-centric monitoring should be able to account for some of these effects because performance data follows the potential paths of indirect performance interaction.

The grand challenge for analysis of performance data to fuse data from processors, other hardware resources, software components, and communication into a form that provides insight into the full range of performance problems in parallel applications. A particular challenge is to achieve the insight possible with space-time diagrams from message traces without exhaustive tracing (which is expensive) or diagrams too large to be rendered.

Modeling Application and System Performance and Reliability

The modeling of high performance software and hardware systems is highly complex and requires a deep understanding of the interplay of architecture and software at a variety of scales. The PAL team at LANL has focused on manual techniques for macroscopic modeling of parallel programs to gain insight into their expected performance on a range of parallel systems. A major thrust of modeling efforts so far at Rice has been to construct semi-automatically models of application performance at a fine-grain level to understand the interplay of the application's instructions with a microprocessor core and memory subsystem.

The resulting performance models can be used for analyzing node performance on both existing and proposed future architectures. In software development, these models can be used to pinpoint particular interactions between hardware and software features that are causing performance loss. Accurate models are a unique tool for architecture design. We propose to develop performance models of programs representative of ASC codes to understand their key performance characteristics and to understand how alternative architectures could improve application performance.

A focus of ongoing work is to assess the potential performance benefits and bottlenecks associated with emerging multi-core processors. Key issues of importance include understanding the impact of shared memory hierarchy components. Analysis of application models for multi-core processors should yield insight into what new compiler strategies will be needed to boost efficiency of multi-core chips. This problem may be complicated by active power management of multi-core chips – improving the compiler to generate tighter code for multi-core chips may ultimately hurt performance if this causes the core to be clocked slower because of high power dissipation.

Modeling Communication Performance

We intend to predict the performance of a parallel application as a function of its single node performance and its communication cost. This work will build on Rice's infrastructure for constructing models of single node performance. To understand the cost of communication, we must model the communication

patterns of the application. We plan to extend our current binary analysis and instrumentation infrastructure to measure the volume of communication at each synchronization point. A synchronization point is defined by a call to a communication function. The user can specify the set of communication functions to be monitored at instrumentation time.

For each process of a parallel application, we propose to collect a trace of synchronization events interleaved with the histogram of basic blocks that were executed in each synchronization interval. At each synchronization point, we will collect the size of messages exchanged with each communication partner. We will aggregate the collected data per computation interval, where a computation interval is uniquely determined by the addresses of the call instructions at its two endpoints.

For each such interval, we will know the communication functions invoked at its starting point and its ending point, the number of times the interval executed, the distribution of message sizes sent and received at its starting point (stored as a histogram), and the histogram of basic blocks and edges executed during the interval.

For each synchronization interval we plan to model the distribution and structure of the histogram of message sizes exchanged at its starting point as a function of node problem size, or/and as a function of the number of processes. This problem is similar to modeling the distribution of memory reuse distances seen by a memory reference, and we plan to use a similar modeling approach. Like reuse distance, communication volume is a machine independent application characteristic. Thus, the models of communication patterns constructed in this fashion are portable across architectures, and for predictable applications, we expect the models of communication histograms will be highly accurate.

To compute computation cost between synchronization points, we will need to modify our instruction scheduler to reconstruct executed paths in partial control flow graphs. To determine the overlap between communication and computation, the user must specify for each communication function if it is synchronous or asynchronous. For synchronous synchronization points, there is no overlap between communication and computation, and the execution cost of an interval is the sum of its aggregated communication cost and its aggregated computation cost.

For asynchronous synchronization points, a simple model for the execution time is to consider the maximum between the communication and computation costs, plus an eventual communication start-up cost. However, such a model is likely to under-predict the execution time because for each synchronization interval we have a distribution of exchanged message sizes, and an aggregation of basic blocks executed during all traversals of current interval. This information is not enough to match the communication cost and computation cost of each traversal. It may happen that for some executions of the interval the cost of communication is larger than the computation cost, while for other executions the computation cost is higher. In such a scenario neither aggregate cost is completely hidden. Understanding the level of overlap for arbitrary programs is a difficult problem and will require further study.

Modeling Failure and Integrated Performance/Failure

During the past year, the UNC team has implemented a framework where failure and performability models can be implemented and evaluated. The Failure Indicator Toolkit (FIT) is a library that supports gathering of health-related diagnostic information, developing statistical models of normal system health, and determining the probability that recent health information indicates a failure mode. FIT collects health data via the HAPI library, and it can also be stored for offline analysis and experimentation.

FIT currently features a single failure predictor that makes a binary failure/no failure prediction based on a threshold applied to a single health information datum. We propose to implement sophisticated statistical models for health classification including hidden Markov models, neural networks, and Bayesian probability approaches. Each model will be automatically trained on known good health data. Once the model is trained, failure prediction takes place whenever new health data is available.

The FIT framework makes comparing the predictive accuracy of different models on the same health data an easy task, allowing for quick determination of which model(s) work best with the data. As part of this work, we will develop rules-of-thumb for matching health information sources to appropriate failure predictors. Finally, we will develop and expand performability models that combine both fault-tolerance and performance for systems containing thousands of nodes. These models will include total time to solution as a function of failure modes and probabilities.

Improving Performance and Reliability

Raw performance and reliability data are the inputs to analysis tools. From manual, semi-automatic and automatic analysis come insights into the causes of poor performance or reliability and opportunities to address these problems. Our goal is to develop tools and approaches that can help applications achieve high performance even when system components fail or applications are subject to other system constraints. Strategies for automatic performance steering based on performance and fault models offer the potential to enable long-running programs to repeatedly adjust themselves to changes in the execution environment – perhaps to opportunistically acquire more resources as they become available, to rebalance load or adapt to component failures. Moreover, measurement of environmental conditions on nodes promises to allow users and schedulers to balance checkpoint frequency and partition allocation based on failure likelihood.

In addition, validated performance "contracts" among applications, systems, and users that combine temporal and behavioral reasoning from performance predictions, previous executions, and compile-time analyses are one promising approach. This work 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.

Improving Reliability through Fault Tolerant Algorithms

Coping with the failure of system and application components is difficult. We propose to create a framework, which we call FT-LAP, that will simplify the development and implementation of fault tolerant software libraries by providing support for detection process faults, and recovery of data in the presence of multiple process failures and high failure rates. The FT-LAP framework will

- exploit *diskless checkpointing*, which offers unique advantages for resiliency and performance in the massively parallel environments that we are targeting,
- build on fault tolerant MPI (FT-MPI)—a state of the art implementation of the MPI standard constructed by a team led by Dongarra that has extended failure semantics and corresponding mechanisms to enable applications to detect and recover from faults at levels of functionality previously unattainable, and
- take an *algorithm-centered approach* to fault tolerance that maximizes programmer opportunities to utilize application specific knowledge to achieve the kind of unprecedented efficiency and performance that future high end computing environments will demand.

To survive multiple process failures by using as small processor redundancy as possible, we propose to use error-correcting techniques to encode the local checkpoint data and store the encoded data into some dedicated checkpoint processes.

Diskless checkpointing requires that each process maintain a local copy of the checkpoint data in the local memory of each process, so that a consistent state can be reconstructed during recovery. Diskless checkpointing approach is especially appropriate for iterative methods. In most iterative methods, the application will synchronize on each iteration and modify only a small amount of its state between iterations. This implies that only a very small amount of data needs to be checkpointed and hence a very low fault tolerance overhead is possible. If a smaller amount of extra memory is available in each

process, then we propose to use a reverse computation-based technique combined with error correction coding.

To enable a seamless computing environment offering interchangeability of compute resources and scalability from the desktop to thousands of (heterogeneous) processors including coarse grained multiprocessor supercomputers, we will develop a framework for the development and implementation of fault tolerant software libraries for the core linear algebra area that will enable easy diskless checkpoint, detection process faults, and recovery of data in the presence of multiply process failures We propose to incorporate into libraries the techniques runtime support for adaptive strategies to allow diskless checkpointing during the course of execution, produce a design for extensions to provide recovery and process initiation in the presents of faults, exploit both loss-less and lossy approaches in recovery from faults in the execution of the libraries, and produce a set of "recovery libraries" for use in application studies, and evaluate their use in practical settings.

Improving Application Reliability via Adaptivity and Recovery

To support applications on large-scale systems, more active techniques are needed that enable systems and applications to continue operating even when some components fail. Without such schemes, the mean time to failure (MTBF) of systems with tens of thousands of commodity components, even when operated as quasi-independent partitions, will continue to decline and will severely limit solution feasibility for complex problems.

As noted earlier, health monitoring and failure prediction tools alone enable post-mortem analysis of failures. To increase the MTBF for applications and systems, online techniques are also needed that enable developers and applications to adapt program behavior. The use of tens of thousands of commodity components to increase peak performance will lead to intolerable failure rates without appropriate power management. Monitoring and analysis of power-performance profiles of scientific applications will be needed to optimize power consumption while maintaining performance. This will be particularly true for systems with multi-core chips.

A second way health monitoring information can be used is to improve reliability is to let an application use it to choose hardware partitions and checkpoint frequencies based on the application's own resilience to failures and models of anticipated failures. We propose to develop batch queue selection mechanisms that will allow application developers to select batch queues based on likely failure probabilities and modes. Intuitively, applications that are more resilient to transient errors (e.g., memory word or message bit upsets) due to internal consistency checks or varying memory footprints are more likely to complete successfully on less reliable hardware.

We will also develop an adaptive toolkit for application checkpointing that will choose checkpoint frequencies automatically as multiples of a user-specified threshold. For example, if an application could potentially checkpoint its state every K iterations, the adaptive toolkit would trigger a checkpoint every cK iterations, where c is determined based on acceptable overhead and probability of failure.

Evaluating the Promise of Architectural Alternatives

As architectural complexity rises, with multicore chips, network interface processors, complex storage hierarchies (memory and I/O), and the rise of special purpose co-processors, understanding the relative performance and reliability advantages that accrue from these alternatives is increasingly critical. We propose to develop a suite of tools that can be used to assess these tradeoffs.

In particular, Los Alamos and other laboratories are evaluating alternate architectures that exploit FPGAs and graphics processors (GPUs) for high-speed computing. These systems combine FPGAs or GPUs with standard nodes, offloading computation intensive routines to the auxiliary processor. With this additional complexity comes additional failure modes, power and temperature management (GPUs are notoriously hot) and options regarding data reuse and computation overlap.

We propose to explore the monitoring, adaptation and reliability issues associated with these hybrid systems, using the tools and techniques outlined in this proposal. In particular, we plan to use data gathered using message-centric monitoring to analyze application performance and to analyze architectural issues to understand performance tradeoffs among improvements in network bandwidth, computing power, and storage performance.

In addition, by understanding the interplay of instructions in an application's core loops, understanding the temporal and spatial locality (or lack thereof) of memory reference patterns, one can assess whether an application can be more effectively performed with an FPGA, a GPU, a processor-in-memory or a vector unit rather than a microprocessor core. Our aim is to provide semi-automatic tools for assessing alternatives for arbitrary applications by combining static and dynamic information into detailed loop-level models of application performance.

1.1.3 Project Tasks

FY06:

- Develop and deploy techniques and tools for call-stack sampling based monitoring and analysis of parallel codes.
- Design and prototype measurement and analysis techniques for emerging architectures. Focus: multi-core processors.
- Design and implement software interfaces for using off-processor counters to assess application performance and reliability.
- Develop and validate stratified sampling for scalable monitoring of performance and system health.
- Evaluate message-centric monitoring of LANL-relevant applications on large-scale systems.
- Use message-centric monitoring to drive load-balancing decisions on dynamic applications.
- Integrate tools for vertical and horizontal message-centric monitoring.
- Evaluate statistical methods for analyzing performance data from extreme-scale systems.
- Develop a prototype that semi-automatically produces models of communication performance for parallel applications using a combination of static and dynamic analysis.
- Develop basic "micro-climate" sensor infrastructure for rack and machine room monitoring.
- Assess prototype failure indicator and prediction mechanisms for predicting likely component failures.
- Develop adaptive checkpoint schemes based on failure indicators.
- Assess and couple prototype batch queue selection mechanisms with failure indicator toolkit.
- Design and prototype diskless checkpointing in which the checkpoint processes participate in the computation and rearrange data when a failure occurs. Use p processors for the computation and have k of them hold checkpoint.
- Evaluate algorithms for coordinating local checkpoint/restart in which processors hold backups of neighbors.
- Evaluate processor/co-processor system performance based on FPGAs and GPUs.

FY07:

- Extend and apply micro-climate senor infrastructure and integrate with health/performance monitoring toolkits.
- Adaptively tune online message-centric monitoring to balance accuracy and overhead.
- Use online monitoring information for detecting operating system-caused performance anomalies.
- Design, and prototype integrated analysis capabilities that provide a global view of application performance by combining information from communication analysis with node-level performance data.

- Develop software to determine the checkpointing interval and number of checkpoint processors from the machine characteristics using historical information. Migrate tasks if a potential problem is noticed.
- Coordinate and couple integrated health/performance monitoring and failure prediction with checkpointing, back queue selection and adaptation policies.
- Develop and extend processor/co-processor adaptability (i.e., the ability to trigger execution of adaptive libraries on processors or co-processors based on expected performance gains).

FY08:

- Design and prototype measurement and analysis tools for emerging architectures. Focus: forthcoming HPCS architectures.
- Expanded evaluation of integrated adaptation and performance/reliability schemes with target applications.
- Evaluate processor/co-processor infrastructure with target applications.
- Evaluate checkpointing with "real applications" and investigate "lossy" algorithms.
- Generalize the prior prototypes to provide a library of routines to do the diskless checkpointing.

1.1.4 Relevance to Weapons Program

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. The aim of this research is to provide ASC application teams with

- better methodology and tools for measuring application performance, resource utilization, and system health,
- better techniques and tools for analyzing program and system performance and reliability,
- insights into the interplay between applications and systems that leads to better mapping of applications onto current and emerging extreme-scale systems,
- strategies and mechanisms to support construction of failure-resilient programs, and
- insights into the nature of extreme-scale parallel architectures and applications that will help improve the design of future systems.

Better tools and techniques to monitor and analyze application performance on current-generation architectures are a good investment for both the short and long term. In the short term, they help pinpoint bottlenecks in existing applications; this enables scientists tune applications for better performance. Over the long term, understanding performance bottlenecks and reliability limitations in today's applications and systems will focus research on practical software and hardware technologies that hold the most promise for improving performance and scalability of future applications on the next generation of computer systems.

1.1.5 Budget

Total Proposed Budget: $578,816
Breakdown by Site: Rice: $216,016; UNC: $233,000; UNM: $36,600; UTK: $93,200

1.2 Compiler Technology for High-Performance Computing

Supported Personnel:

Project Leads: John Mellor-Crummey, Rice, johnmc@cs.rice.edu; Ken Kennedy, Rice, ken@cs.rice.edu, Keith Cooper, Rice, keith@cs.rice.edu

Other Supported Personnel: *Rice*: Tim Harvey, Guohua Jin, Robert Fowler

LANL Point of Contact: Rod Oldehoeft, rro@lanl.gov

1.2.1 Vision

Programming extreme-scale systems efficiently is very difficult. Difficulties exist at two levels: writing parallel programs and generating efficient code for processing elements.

Writing parallel programs is hard because parallel programming models today require users to manage resources of parallel systems at a very detailed level. When using MPI – the dominant programming model for writing scalable parallel programs – programmers must partition application data structures and computation, add data movement and synchronization, and manage storage for non-local data. Because of the explicit nature of MPI communication, significant compiler optimization of communication is impractical. For this reason, all responsibility for optimizing communication performance falls to application developers. For high efficiency, application developers must choreograph asynchronous communication and overlap it with computation. This complicates parallel programming significantly.

Higher-level programming models that simplify the construction of parallel programs are needed. However, for high-level programming models to succeed, they must meet a number of requirements. First, they must be ubiquitous; otherwise they will be unattractive to application developers. In practice, this means that in addition to being available on extreme-scale systems, they must also execute on laptops and clusters. Second, high-level programming models must be expressive and conveniently support a wide range of algorithms and programming techniques. Ideally, high-level programs should be easy to write, read, maintain, and reuse. Third, the efficiency of code generated from high-level programs must be comparable to that of the best hand-coded low-level parallel programs. Finally, to attract application developers, high-level models must have a promise of future availability and longevity.

Regardless of how programs are written for extreme-scale systems, solving large-scale problems requires using processor nodes as efficiently as possible. Failure to do so would increase the time to solve problems on processor ensembles of a fixed size or require larger systems to maintain the same time to solution. The effects of efficiency gains made at the level of individual nodes are multiplicative. For this reason, compiler technology for generating efficient code for processor nodes is of critical importance.

1.2.2 Compiler Technology for Parallel Systems

Today, fragmented programming models such as MPI require application developers to explicitly manage separate address spaces for each processor as well as manage communication and data movement. High-level programming models that abstract away the details of how computation is partitioned as well as data movement and synchronization are needed if we want to increase the productivity of application developers.

Whether programming for single chip multiprocessors such as Cell, or large-scale parallel systems, one should be able to use a simple programming model and compilers should take care of the details of managing data movement and synchronization without sacrificing much of the performance that could be achieved by hand coding. Global address space programming models offer the simplest abstraction for expressing parallel programs. We believe that global address space programming models are the appropriate model for programming a wide range of parallel systems from single-chip multiprocessors to

emerging large-scale parallel systems such as Cray's Red Storm and systems that we expect to 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. We propose to explore the utility of SPMD global address space programming models for parallel programs of interest to the LANL.

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. We propose to investigate compiler technology that makes it possible to execute high-level data parallel programs efficiently on systems ranging from single-chip multiprocessors such as Cell to commodity clusters.

1.2.3 Compiler Technology for Node Programs

Achieving high levels of performance on modern computer architectures requires careful planning and execution at every level of the tool chain that translates the application's source text into executable code for the target machine. Various tools must identify large-grain parallelism and use it to occupy multiple processors. They must design appropriate strategies to pass data between the distinct threads of the computation, insert the code to implement the needed communication, and adjust the strategy to balance communication and computation. The tools must ensure that the code, as translated, has a memory reference pattern that takes appropriate advantage of the multiple levels of memory hierarchy, blocking for one or more levels of cache and for register reuse. The final steps of translation map this carefully planned behavior into the instruction set of the target machine and try to balance the needs of the computation against the processors' actual resources. To achieve high performance, all of these steps must be done well.

At this final level of translation, backend optimization and code generation algorithms strongly influence the final performance of an application. The major concerns at this level of translation include: 1) keeping the functional units busy, 2) hiding the latencies inherent in the processor and memory system, and 3) ensuring that the final code uses the processor effectively. The current generation of open-source compilers (and their commercial cousins) routinely produces code that obtains 5 to 10% of the processor's peak performance. In part, this underperformance arises from the use of backend optimization and code generation algorithms that were developed for systems with much simpler performance characteristics.

In this project, we will examine several specific backend compiler problems that appear to have an impact on the performance of ASC applications. In particular:

- *Changing the instruction mix*: In the instruction stream of a running application, the mix of operations that the processor encounters should match the set of functional units that the processor provides. A mismatch between application operations and processor capabilities leads to saturation for one set of functional units and starvation for another. We have begun an investigation into automatic techniques that use backend optimization to alter an application's instruction mix. Our initial work has focused on critical loops of Sweep3d. This example suggests that the problem will require a coordinated attack by a small suite of backend optimizations.
- *Aggressive rematerialization*: The high costs associated with accessing memory present a significant performance challenge on modern systems—one that will continue to grow as memory latencies increase and as the number of CPU cores per chip increase. Techniques that can reduce the number and frequency of memory accesses should improve application performance. Rematerialization is a code generation strategy that recomputes certain values when doing so is less expensive than accessing them from memory. Today, it is applied in register allocators to values that can be computed in a single operation and whose operands are all available — a standard that misses many profitable opportunities. We will develop a standalone rematerialization algorithm (outside the allocator) that uses a tunable metric to determine profitability. The resulting technique should directly reduce memory accesses and indirectly reduce cache disturbance from scalar loads.
- *Scheduling and register allocation*: A major open challenge in code-generation algorithms lies in coordinating the behavior of instruction scheduling and register allocation. Prior work in this area has produced algorithms that coordinate their behavior but favor one process over the other—either underallocating registers to reduce constraints on the scheduler or limiting the scheduler's ability to move operations when such motion increases demand for registers. Early work by Schielke shows the potential for a truly fair technique to find improvements in both scheduling and allocation — reducing stalls and spills at the same time. Unfortunately, his technique relied on extensive randomization and hundreds of trials, making it impractical for routine use. We will explore algorithms that solve these problems together, working in the context of the Callahan-Koblenz register allocator that we have built for LLVM.

All of these explorations will be conducted in the code context of one of the two open-source compiler systems, Open64 or LLVM, that we believe are viable candidates for future development. We will work with external collaborators on LLVM and Open64 to help ensure that there are viable open source compilers for directly transferring compiler research to the ASC program. In each investigation, we expect to apply the lessons that we have learned in designing adaptive techniques as part of the LACSI Autotuning effort. We will test the algorithms on ASC-related codes.

1.2.4 Proposed Milestones

FY06:

- Report intermediate results of ongoing work on techniques to change the instruction mix in performance critical loops. (Quarter 1)
- Perform experiments with additional codes for instruction-mix modification (Quarters 2—4) and develop a strategy to deliver this technology. (Quarter 4)
- Develop an initial aggressive scheduler for LLVM. (Quarters 1 and 2)
- Refine parallel compiler analysis and code generation techniques for dense matrix codes to efficiently support programs with strided alignments.
- Design and prototype data-parallel compiler technology to map application codes to single-chip multiprocessors with explicitly-managed memory (such as Cell).

FY07:

- Design and implement a tunable rematerialization pass for LLVM (Quarter 2), followed by experimental evaluation and report (Quarter 4)
- Initial version of algorithm for combined scheduling and allocation (Quarter 2)
- Design and implement prototype compiler and run time support for general block distributions.
- Design and implement a prototype compiler that uses data-parallel compiler technology to map application codes to ensembles of single-chip multiprocessors with explicitly-managed memory (such as Cell).

FY08:

- Refine a combined scheduler and allocator (Quarters 1 and 2); distribute as part of the LLVM compiler (Quarter 4)
- Distribute aggressive rematerialization pass for LLVM (Quarter 2)
- Develop prototype compiler and run time support for user-defined block distributions in high-level models for parallel programming.

1.2.5 Relevance to Weapons Program

One aim of this research is to produce efficient high-level programming models suitable for future ASC applications. A second aim of this research is to develop compiler technology that will make it possible to use compute nodes in extreme-scale systems efficiently. The findings from this research and software infrastructure, developed as products of this effort, are expected to benefit all ASC application teams.

1.2.6 Budget

Total Proposed Budget: $346,925

Breakdown by Site: Rice: $346,925

2. Components

2.1 Component Architectures for High Performance Computing

Supported Personnel:

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

Other Supported Personnel: *Rice*: Zoran Budimlic, Rob Fowler, Guohua Jin, Cheryl McCosh, John Mellor-Crummey, *UH*: Lennart Johnsson, Postdoc/Research Scientist to be hired

LANL Point of Contact: Craig Rasmussen, rasmussn@lanl.gov

2.1.1 Vision

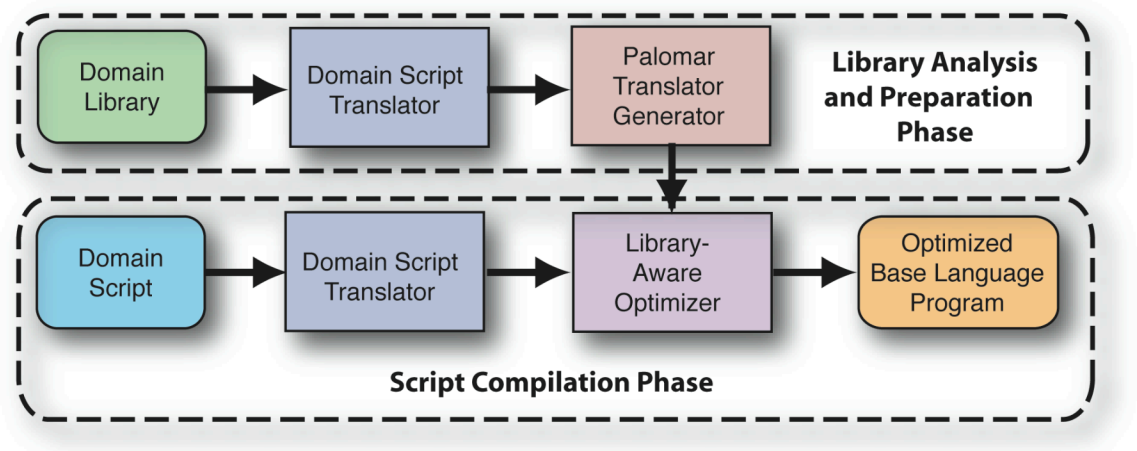
The goal of component architectures research is to make it easy to develop high-performance applications by using compiler-based tools to integrate components from pre-existing libraries. By providing modularity through well-defined component interfaces, component integration systems permit scientists and software developers to focus on their individual areas of expertise: scientists can concentrate on integrating applications by gluing domain-specific components together using some form of scripting language, leaving the development of those components to expert programmers. In addition, because components foster more code reuse, they provide an increased economy of scale, making it possible for

resources to be shifted to areas such as performance engineering, testing, and platform adaptation; the result is improved software quality, portability, and application performance.

A fundamental obstacle to this strategy is that scientific application developers, particularly those at Los Alamos National Laboratory, cannot afford to sacrifice significant amounts of performance for this clearly useful functionality. An important objective is to produce applications without sacrificing performance relative to hand-integrated codes. To address this problem, the component integration research effort is exploring 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.

In the strategy we envision, applications would be defined using high-level scripting languages such as Matlab or Python to coordinate invocation of library operations, although traditional languages such as Fortran and C++ could also serve this purpose. Because scripting languages typically treat library operations as black boxes that cannot be optimized to the contexts in which their invocations appear, these systems often fail to achieve acceptable performance levels for compute-intensive applications. One solution strategy that has been successfully employed in previous research is to translate scripts to a conventional programming language and use whole-program analysis and optimization to improve performance to acceptable levels. The main drawbacks of this approach are long script compilation times and the absence of an effective way to exploit the domain knowledge of library developers.

To address these problems, this project is 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, depicted in the figure below, language implementation consists of two phases. The offline *library analysis and preparation phase* digests annotations describing the semantics of library routines, combines them with its own analysis to generate specialized variants of each component in the library. In addition, it produces a *library-aware optimizer* that understands library entry points as language primitives. The *script compilation phase* can then use the generated compiler to produce an optimized base language program.



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). We will also explore the use of high-level prototyping languages, such as Matlab and Python, as the scripting languages for application definition.

If this effort is to succeed, it must take into account an important reality: many components will be constructed using object-oriented languages, so techniques for optimizing such languages are critical. We

plan to invest a significant effort in analysis and optimization of components written in C++, Java, and Python.

2.1.2 Research Plan

To achieve the project vision, we plan to conduct research that focuses on *supporting technologies for component integration*, which include:

- *Transformation systems to eliminate overheads due to abstraction.* These systems will employ high-level context information to select the right variants of components for a given calling context and employ macro-transformations that replace expensive sequences of component invocation with more efficient sequences in the given context.

- *Component integration systems that automate specialization.* We will also explore component integration frameworks that generate specialized versions of certain components for common calling context and substitute those specialized versions, statically or dynamically, in applications that employ those components.

A driving problem for this research is to construct component integration frameworks that can efficiently integrate data structure components with functional components. An example would be the integration of sparse versus dense array data structures with certain types of linear algebra operations. This has been an important goal of the generic programming research area, and success will require advances in both component design and transformation-based integration frameworks.

In this work we will continue our efforts to collaborate with weapons code projects at LANL. Of particular importance is the collaboration with the Marmot Project, because that project is attempting to employ modern software design and integration strategies to code development. We will continue to execute the draft collaboration plan (described below) that was developed in workshops with the Marmot team. In addition, we will develop new collaborations with the traditional code teams.

As a driving problem, our work will focus on an unclassified, export-restricted test code that is representative of LANL codes that combine hydrodynamics with radiation transport. We will also explore important related codes from the Telluride and Marmot projects.

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

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 LANL weapons-related 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.

Some specific research challenges in this area include:

Exploration of array and mesh abstraction. We will investigate the issue of separating content and layout in an array abstraction. The specific issue is whether it is possible to easily substitute different data layouts for the same abstract data structure without substantively changing the functional components that use them. This research touches on the extensive research in “generic programming” because it may require some coding standards for functional components to achieve maximum flexibility. However, all data-layout-specific code will be encapsulated with the data structure component.

Preliminary work on this effort will be conducted in the context of Matlab (or Python), beginning with the definition of a generalized array data structure that incorporates data distribution. Specific array

distributions for sparse matrices will be explored as a way of understanding the crucial performance issues. (This may contribute to work on a parallel version of Matlab, described below.) Once the Matlab array prototype has been explored, we will focus on ASC 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.

The development of new specialization strategies for components. As a specific challenge, we will explore static and dynamic approaches to reduce the time and space required to deal with specialization of multiple materials in cells. This problem arises because many codes iterate over all potential materials that might occur in a cell, even though most cells will have far fewer than the maximum number of materials. If pre-specialized code can be generated for the cells with small numbers of materials, it may be possible for the right specialized variant to be selected statically or dynamically, saving substantial space and time. The goal is to permit the developer to specify the handling of materials in the most general way possible, and let the compiler and run-time system handle the specialization. These strategies can be further enhanced by compiler-based specialization to sparse data structures and by reorganization of computation to deal with block of cells with similar properties at the same time.

Demonstration of these techniques in specific applications of interest to ASC and LANL. We will continue to seek appropriate applications within the weapons program. To that end, we will use Sage and Sweep3D to represent a standard hydrodynamics and radiation transport application. We will also continue to work with the Marmot and Telluride projects to find representative applications that can be analyzed at the participating universities under export restrictions.

Construction of Efficient Specialized Libraries for Component Integration: This effort will focus on the generation of component libraries that can be incorporated into high-performance applications and can be the bases for domain specific problem-solving environments. 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. The effort will also focus design strategies for efficient data structure components.

Under this project we plan to pursue the following specific research directions

- *Tools for Pre-optimization of Libraries:* We will explore some component libraries that are used within the weapons program to identify opportunities to pre-specialize them to expected calling context. One early target for this work will be Trillinos.
- *Mining of Traditional Applications:* We will explore automatic strategies for mining traditional applications and libraries to construct component libraries that might be employed in other applications. In particular, we are interested in exploring the automated translation of older applications into script-based domain languages, in which components of the original program form the library underlying the domain language.
- *Annotation of component libraries for efficient integration via a telescoping languages framework:* This effort will develop annotation strategies, along with the associated transformation systems, that will make it easier to optimize component accesses within application and to replace sequences of calls to library components by more efficient sequences.
- *Factoring of Components for Efficient Recombination:* This work will involve research into the proper structure of libraries so that efficient variants can be synthesized from component pieces once the calling context and target platform is understood.

Compilation and Optimization 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.

Rapid Prototyping in High-Level Scripting Languages: In this effort we will explore the use of scripting and problem-solving languages such as Matlab and Python for the development of high-performance applications. A major problem is how to implement such languages on parallel machines. We plan to pursue two interrelated strategies in this effort. The first will explore a client server architecture in which the server is a parallel cluster and the client is a desktop computer. The second is a compiler-based strategy. These are described in more detail in the following paragraphs.

Matlab optimization via cluster-based computation. We focus our attention on how to provide parallel computation capabilities to such environments that would allow for seamless access to hardware and software resources for numerical linear algebra. Instead of focusing on a particular implementation, we are exploring the design space of such an interactive environment such as Matlab, Python, and Mathematica and the consequences of particular design choices. We have developed a prototype implementation of our ideas with emphasis on the performance perceived by the end user. In our design, we consider primarily a client-server architecture where the server is envisaged as a cluster and the client is a simple desktop computer.

At the moment, the basic infrastructure of our design has been implemented and successfully applied to a dense matrix factorization and iterative solution method in Matlab and Python environments. Our preliminary tests show that the overhead of remote execution can be offset when problem sizes become prohibitive for a sequential environment and it is possible to reap the benefits of parallel computation.

A compiler strategy for scripting languages. This sub-project will explore translation of applications from well-know rapid prototyping languages, such as Python and Matlab, to C and Fortran. The goal of this effort is to avoid the necessity for hand recoding of Matlab or Python components to achieve high performance within conventional applications. We have already developed a prototype framework for compiling Matlab that employs a powerful type analysis algorithm. We propose to expand this work in two directions. First, we will explore extending the compilation strategy to Python. Second, we will explore extending parallelism to Matlab and Python by building on the High Performance Fortran compilation technology we have already built at Rice. The result will be a Matlab variant from which efficient Fortran plus MPI can be generated. (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.)

2.1.3 Tasks

FY05 tasks under the previous LACSI funding for components:

- 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)

FY06:

- Produce a more sophisticated prototype component integration system that would support both Matlab and C++ components. (Quarter 4)
- Develop a plan for component specialization, with a focus on the multiple materials effort and deliver a technical report on this strategy. (Quarter 2)
- Produce a report on specialization strategies for Trillinos and other libraries. (Quarter 3)
- Produce a definition of the first annotation language for component integration. (Quarter 2).
- Develop a preliminary version of the JaMake optimization system for optimization of components in C++, including object inlining. (Quarter 4).
- Demonstrate a more sophisticated implementation of distributed matrices in Matlab and possibly Python, supporting more than one distribution. (Quarter 4)

FY07:

- Demonstrate advanced component integration on one ASC application.
- Produce a hand-coded version of the proposed multiple-materials strategy and deliver a report on how this will be automated.
- Develop, using a combination of hand and automatic methods, specialized versions of components from Trillinos or another suitable library and demonstrate these in a prototype application.
- Produce a report on potential for mining existing codes for use in modern component libraries.
- Develop a more sophisticated version of the JaMake optimization system for optimization of components in C++, including an advanced type analysis strategy, and produce a report on its effectiveness.
- Demonstrate translation of prototype parallel Matlab and Python programs to C and integration with existing ASC frameworks.

FY08:

- Conduct experiments to explore the effectiveness of advanced component integration strategies, including optimizations based on annotations, on a variety of applications, producing reports as appropriate.

- Produce a prototype system for automatic specialization of programs and demonstrate it on a simplified multi-materials code.
- Deliver a component transformation and optimization tool for C++ based on JaMake.
- Conduct experiments on the efficiency of integrating prototypes written in Matlab and Python within realistic code frameworks.

2.1.4 Relevance to Weapons Program

- Software quality is a critical concern for weapons codes. Component integration systems provide a widely-accepted approach to software organization, increasing both reliability and flexibility.
- Performance, both in terms of speed and memory requirements, remains an important concern for weapons codes. This research will explore how to increase code quality and programmer productivity without undue sacrifice of performance.
- Rapid prototyping is an important tool for exploring new algorithms and implementations in scientific codes. Matlab and Python are becoming widely accepted for prototyping. Automating their translation and integration into C and Fortran frameworks without loss of performance (even parallel performance) could dramatically increase productivity of weapons code developers.

2.1.5 Budget

Total Proposed Budget: $315,180

Breakdown by Site: Rice: $213,080; UH: $55,500; UTK: $46,600

2.2 Automatically Tuning and Retargeting High-Performance Components and Libraries

Supported Personnel:

Project Leads: Keith Cooper, Rice, keith@rice.edu; Jack Dongarra, UTK, dongarra@cs.utk.edu; Lennart Johnsson, UH, johnsson@cs.uh.edu; Ken Kennedy, Rice, ken@cs.rice.edu

Other Supported Personnel: Rice: Rob Fowler, Tim Harvey, John Mellor-Crummey

LANL Point of Contact: John Thorp (acting), thorp@lanl.gov

2.2.1 Vision

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.

The fundamental question addressed by this research is: Can we build components with the built-in ability to adapt with high performance to new computational platforms? One model for the kind of component tuning we envision 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 human costs involved in retargeting applications to different machines.

To date, we have been able to extend the success of automatic tuning to a few domains such as linear algebra and fast Fourier transforms. We propose to change that by pursue a project to use advanced compilation strategies and sophisticated component designs, along with extensive amounts of computing,

to accelerate the process of restructuring and tuning components for each new high-performance architecture.

2.2.2 Research Plan

One way to produce excellent code for a new platform is to expend substantive computation time to try different restructuring and compilation strategies on smaller test data sets, picking the best combination for each component on the target architecture. We proposed to pursue three general strategies that use this approach. In each of these, a common methodology is to formulate the problem as an optimization in which the running time of the component on the test data is the objective function. Because the parameter spaces are so large, strategies are needed to reduce the tuning time: these strategies include the use of heuristic search and pruning the search space using compiler models. We propose to continue the work on how to reduce the cost of empirical tuning strategies to manageable levels.

Structural reorganization. Structural reorganization is typically applied to loop nests within a component. Each computation-intensive loop nest is restructured to take advantage of a variety of parameterized degrees of freedom, including amount of parallelism and cache block size. Then, based on an automatically generated set of experiments operating on small training data sets, the tuning system picks the best parameters for a given new machine. This approach has been used in the Atlas system to produce tuned versions of the LAPACK BLAS needed to port that linear algebra package to new systems. We propose to extend this work to systems that can automatically generate the tuning search space for new library components, applying standard tuning methodologies, such as loop blocking, unrolling, unroll-and-jam, loop fusion, and storage reduction.

This work will build on preliminary research at Rice (LoopTool) and Tennessee (Atlas 2) that has constructed prototype systems to search through the parameter spaces for selected optimizations. To date, this work has demonstrated that a significant fraction of optimal performance can be achieved with heuristic searches that require a factor of ten less tuning time.

The preliminary work, some of which was reported in the 2004 LACSI Symposium, employed only two transformations. We propose to extend these methods to the full range of transformations that can improve the performance of loop nests through source-to-source transformations. In addition, we proposed to explore ways to prune the search spaces further through static compiler models and by partitioning the individual searches for different loop nests so that the tuning can be done more efficiently. This latter strategy will build on the performance measurement and analysis tools described elsewhere in the LACSI proposal.

Finally, the Rice and Tennessee efforts will collaborate on methodology and strategies and will use the results of this effort to construct at least one retargetable application of interest to DOE and the ASC program.

Function decomposition. In some specialized domains, notably FFT, large computations can be decomposed into smaller specialized "codelets," tuned for the target platform, which can be recomposed into excellent algorithms when the actual data sizes are known at program initiation time or run time. This strategy has been successfully employed in FFTW and UHFFT to produce excellent results. We propose to explore how this can be extended to other computational domains with well structured (de)composable operations, such multi-grid techniques, approximation schemes for various forms of discretizations, such as finite element, finite volume and finite difference discretizations, and the discretizations and operators in the methods developed by Yuri Kuznetsov and LANL collaborators. After an evaluation of the applicability of the technique in important LANL applications including the methods developed at UH by Kuznetsov, one application will be chosen as a target for applying our techniques for code adaptation.

Compilation reorganization. A modern optimizing compiler consists of a series of passes that translate and improve the code. Each pass performs some specific function. The user can control some aspects of

the compiler's behavior; other aspects of its behavior can only be changed through source-level modification of the compiler. Variations in the applications presented to the compiler and the architectures of the machines that it must target ensure that no single setting for all of these options is "best" for all situations.

Prior research, conducted in part with LACSI funding has shown that using adaptive search to discover the "right" compiler configuration can improve performance over any compiler that uses a single configuration for all programs. Specifically, this work has shown:

- A factor of 20 reduction in function evaluations is needed to find good solutions for the compilation sequence problem from improving the search techniques.
- A factor of 2 to 4 reduction in the overall running time of the search resulted from using models (rather than execution) to evaluate the impact of adaptive choices. (Note that this reduction is orthogonal to the factor of 20 from improved search.)
- Using adaptive techniques inside a single complex optimization, such as inline substitution, can produce significant improvements over fixed-heuristic solutions. Preliminary (unpublished) results show that simple adaptive schemes provide significant improvements over traditional heuristics such as those found in gcc's inliner.
- The specifications found by adaptive search, either for the compilation sequence problem or the inlining problem, vary significantly from application to application. Thus, no single "universal" heuristic or solution works as well as the application-specific answer found by adaptive search.

Other researchers have shown results consistent with ours, using a variety of compilers, environments, and optimization criteria; these papers suggest that the expected range of improvements is 20 to 30% when optimizing for performance and 10 to 20% when optimizing for code space.

In this work, we will explore mechanisms for delivering adaptive tools to end users, particularly the ASC user community at LANL. We envision two approaches:

- Our work on automatically derived optimization sequences, requires a complete compilation system to deliver. We are working to build adaptive choice into the LLVM system, a new open source compiler developed at Illinois; our goal is to make it an attractive alternative to gcc that happens to support adaptive compilation.
- Our work on adaptive inlining fits nicely into a source-to-source framework. Unfortunately, we know of no suitable open-source Fortran 90 parser to serve as the base for such a system. We will build a Fortran 90 source-to-source inliner that includes a robust open-source parser for the language. This tool will let ASC developers use the adaptive inliner on their codes.[1]

We will spend the first year of this project developing infrastructure that builds toward these two goals. We will continue to work inside LLVM to improve overall code quality and to install the features needed to support adaptive compilation. We will develop a Fortran 90 parser and prettyprinter to serve as the base for the adaptive, source-to-source inliner for Fortran 90.

Autotuning Workshop. To facilitate collaborations on automatic tuning strategies, we will hold a workshop at one of the LACSI sites (or at the LACSI symposium) on automatic tuning. All of the LACSI-funded efforts described in this section will participate. In addition, we will invite other nationally prominent researchers in this field.

[1] Finding a good specification requires an adaptive search; that specification can be reused repeatedly at low cost. Thus, the code can be maintained in its original, modular form while obtaining the performance benefits of the adaptive optimization.

2.2.3 Tasks

FY05 (what we promised in the current fiscal year):

- 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)

FY06:

- Develop a prototype automatic code improvement tool, based on LoopTool that includes loop and register blocking plus loop fusion and produce a report on its effectiveness. (Quarter 2)
- Investigate search space methods to optimize the choice of "best" implementation.
- Assess the benefits of UHFFT style code generation and adaptation for components in key ASC applications involving (de)composable operations; Target one type of component for implementation and evaluation of our code generation and adaptation technique.
- Distribute improved back-end components for LLVM. (Quarter 1)
- Thesis on the algorithms needed for adaptive inline substitution and preliminary experiments with optimization selection and ordering in LLVM. (Quarter 2)
- Develop a source-to-source front-end for Fortran 90 that can be distributed under an open-source license. (Quarter 4)
- Host a LACSI Workshop on Automatic Tuning. (time to be determined).

FY07:

- Develop a more sophisticated automatic code-tuning tool, based on LoopTool, that incorporates model-based pruning and includes more optimizations; deliver a report on its effectiveness on applications representative of ASC codes.
- Investigate ATLAS-like optimization for collective message operations to tune for the specifics of the parallel system.
- Extend and assess the code-generation and adaptation techniques to a dominating range of multi-grid methods for weapons codes.
- Begin work on embedding adaptive behavior into another complex optimization.
- Build adaptive inliner for Fortran 90 using source-to-source Fortran 90 system produced in FY06 and test it on ASC codes.
- Experiment with adaptive optimization and selection using LLVM and ASC codes.

FY08:

- Deliver a sophisticated automatic component tuning tool that incorporates loop and register blocking, loop fusion, array copying, scalar replacement and array restructuring; demonstrate it on at least one ASC code.

- Produce a production quality multi-grid adaptive library with assessment of use in at least two weapons codes.
- Develop and test techniques for distributing adaptive search across multiple compile steps in the code development process.
- Distribute adaptive inliner.
- Demonstrate second complex adaptive optimization on ASC codes.

2.2.4 Relevance to Weapons Program

- Retargeting ASC applications to new platforms is a labor-intensive task, with most of the effort going to tuning; automating this task will provide a huge boost in programmer productivity.
- The Atlas experience demonstrates that automatic tuning can provide performance advantages over simple hand tuning (in situations where extensive hand optimization is impractical); hence, this approach could deliver much better performance for ASC applications on new platforms.

2.2.5 Budget

Total Proposed Budget: $290,207

Breakdown by Site: Rice: $188,107; UH: $55,500; UTK: $46,600

3. Systems

3.1 Scalable Optimized Network Protocol Stacks

Supported Personnel:

Investigators: Patrick Bridges, UNM, bridges@cs.unm.edu; Alan Cox, Rice, alc@cs.rice.edu; Arthur B. Maccabe, UNM, maccabe@cs.unm.edu; and Scott Rixner, Rice, rixner@rice.edu

LANL Point of Contact: Ronald Minnich, CCS-1, rminnich@lanl.gov

3.1.1 Vision

The number of clusters in the top 500 supercomputers has been growing since 1983. Currently 60% of the top 500 supercomputers are clusters and 35% of these clusters use commodity interconnects such as Gigabit or Fast Ethernet and run commodity system software and protocols such as Linux and TCP/IP. The low cost and easy availability of these hardware/software combinations makes them useful for the capacity computing systems that support everyday scientific development and lead to later capability computing. Despite this, commodity cluster hardware and system software is still under-optimized for the parallel computing needs of LANL application scientists, limiting the applications that run well on these systems. In addition, current commodity network software does not have optimized support emerging commodity hardware such as single chip multiprocessors (CMPs).

The main objective of this research is to improve the scalability and performance of network protocol processing within the operating system to increase the performance of existing high-performance applications in commodity clusters and expand the set of applications that can viably run on current and emerging versions of these systems. To do so, research is needed on a variety of related issues, including:

- comparison and quantification of the performance of custom and commodity protocols on HPC applications,
- placement and scheduling of a single connection's protocol processing functionality across network interface cards and a host processor, i.e. protocol splintering and offload, and
- scalability of protocol processing across multiple host processors in emerging CMP systems.

3.1.2 Proposed Research

Protocol Research Testbeds

HPC systems have historically offloaded portions of protocol processing onto the network interface card (NIC), and used custom protocols for data transport. This trend continues today; Infiniband NICs, for example, are currently supported by a custom software stack comprising hundreds of thousands of lines of code.

Well-tested commodity protocols such as TCP, in contrast, do not have the large development and maintenance overhead of these custom software stacks, but have typically run only on the host processor. Several groups including our own have shown benefits to both mainstream and HPC applications from offloading portions of commodity protocol processing onto dedicated processors or ASICs.

While commodity protocol offload research is important and has great potential for improving the performance and cost-effectiveness of commodity clusters, there are few testbeds for continuing this research. NIC vendors have closed or are closing their on-NIC programming interfaces, and the remaining systems that do support on-NIC processing are increasingly out-of-date. The ACENIC Gigabit Ethernet card, for example, has substantially higher per-packet latencies than current Gigabit Ethernet cards such as the (non-programmable) Intel EEPRO 1000, and several times less bandwidth than modern Infiniband cards.

To address this problem and facilitate further research on offloading commodity protocol processing using techniques such as splintering, checksum offload, and NIC virtualization, we propose to construct two different networking testbeds:

1. an FPGA-based Ethernet NIC, and
2. an Infiniband-based software/hardware testbed based on dual Opteron motherboards with bus expansion slots and Infiniscale NICs from PathScale, Inc.

The FPGA-based NIC will be an appropriate platform for exploring NIC technology realizable in the near-term and the resource limitations inherent to such platforms. In contrast, the Infiniscale-based testbed will give us a platform for exploring longer-term offload and OS-restructuring issues.

In addition to research protocol splintering and protocol scalability described below, which will be enhanced by the availability of this testbed, other HPC-related research will also be based on this testbed. For example, by making our device driver for the custom NIC present itself as a straightforward high-speed Ethernet device, this testbed will allow the direct comparison of the performance of Infiniband protocols with TCP/IP protocols running on the same fabric. If current TCP/IP implementations are not sufficient to compete with the Infiniband protocol stack, this testbed will also allow us to characterize what optimizations are necessary to both the operating system software and the protocol itself to achieve competitive performance while maintaining compatibility with commodity systems.

Protocol Offload and Splintering

Offloading protocol processing onto a NIC has been very successful both for traditional applications using TCP/IP and for HPC applications, though it has been explored primarily in the context of traditional applications. Recently, we have studied the effect of a particular offload strategy, splintering, which drastically reduces the protocol processing overhead associated with TCP/IP, freeing up CPU resources for use by HPC applications. More work is needed to precisely understand the costs and benefits of different protocol offload strategies, including splintering, traditional full-protocol offload, partial protocol offload techniques such as ack and checksum offload, and other techniques such as offloading only demultiplexing.

Our first goal is to precisely quantify the benefits of protocol splintering to existing HPC application. Our previous research has shown, using models and prototype implementations, the potential benefits of splintering under the assumption that there is blocking time in most HPC applications for hiding the overheads of protocol processing. To fully understand the benefits of splintering, we need to measure existing HPC applications starting with well-known benchmarks such as the NAS and ASCI Purple benchmarks to find the extent to which protocol processing can be hidden in blocking application calls. This work will start by measuring protocol processing overheads and opportunities for protocol overhead hiding in these applications. As the networking testbeds described in the previous section become available, we will validate these measurements on the testbed and measure the achieved performance improvement of TCP/IP splintering in real HPC applications.

Our second goal is to study other methods of offloading portions of individual connections besides splintering. For example, past work on early demultiplexing by the PIs and other researchers has shown benefits from using the NIC for demultiplexing packet arrivals. These benefits were primarily to resource management in network servers; in the commodity HPC world, however, effective MPI matching with early TCP demultiplexing has the potential to avoid large memory copy costs currently incurred by TCP-based MPI implementations. Longer term, we would also like to explore more dynamic methods of splintering that determine at runtime which parts protocol processing, and its associated data, should be offloaded to the NIC, which should be run on the host processor, and which should be deferred until the application blocks. Prior to conducting this research, however, we must finish characterizing existing offload systems as described previously.

Protocol Processing Scalability in CMP Systems

As mentioned previously, current technology trends are pushing microprocessor architectures towards single chip multiprocessor systems. In CMP systems, each processor on a node may actually have less performance than a monolithic uniprocessor, but each node will likely include tens of processors. Therefore, the scalability of network protocol processing is an important direction for research in network protocol performance optimization. Ideally, a scalable network protocol stack would improve performance for all of the servers' network connections and support a higher connection capacity as processors are added. The purpose of this research is to improve the scalability of state-of-the-art operating system techniques through improved hardware and software design.

Operating systems researchers and designers have used two different synchronization approaches to achieve scalable networking performance: data synchronization and control synchronization. Both techniques focus on enabling parallelism by allowing independent intrakernel functions to proceed simultaneously, but they differ in the way they organize per-thread access to kernel functions and data structures. Restricting the manner in which threads are allowed to access kernel resources can reduce synchronization overheads, but such restrictions may lead to load imbalances. The data synchronous approach, as taken by FreeBSD 5 and Linux 2.6, allows multiple user-level thread contexts to access kernel functions and data as necessary to service network requests. However, thread accesses to shared data structures and the network interface require synchronization. Scalability and performance efforts have been focused largely on creating fine-grained data structures that reduce the likelihood of lock contention and increase the number of accesses that are thread-private. The control synchronous approach, as taken by DragonflyBSD and Solaris 10, remaps accesses to major kernel subsystems to dedicated kernel threads based upon connections. Upon entry into the network subsystem, a network request is remapped to a kernel thread via the scheduler; accessing the scheduler requires synchronization, but by isolating connections to specific kernel threads, all subsequent accesses to connection metadata are thread-private and require no synchronization. Incoming packets are uniquely mapped to connections via the (source address, destination address, port) tuple, and outgoing requests are mapped to connections via the socket interface.

We intend to quantify and compare the overheads and scalability of the two parallelization techniques. To isolate the effects of parallelization techniques on network performance, we intend to modify the FreeBSD 5 kernel to implement the control synchronous mechanisms in addition to the existing data synchronous mechanisms. With this infrastructure we can measure and compare the two techniques on a 4-way AMD Opteron multiprocessor system over a 10 Gigabit Ethernet network. Furthermore, we expect the communication between the network interface and the kernel to limit the scalability of both schemes. The control synchronous scheme opens the opportunity for the network interface to streamline that communication by demultiplexing connections to their controlling threads directly on the network interface. With the FPGA-based NIC described above, we will develop the appropriate support for such parallelized network stacks.

3.1.3 Breakdown of Work and Yearly Tasks

Rice University is focusing primarily of horizontal decomposition of the protocol stack for multiprocessor scalability and the associated near-term FPGA testbed. University of New Mexico is focusing primarily on vertical decomposition of protocols through techniques such as offload and splintering, as well as the Infiniscale testbed that will enable further research into more aggressive offload techniques and high-end NIC capabilities. Both Universities, however, will utilized the techniques and testbeds developed by the other through close collaboration to enhance their research and, as a result, their contributions to LANL-relevant codes and programs.

FY06:

- Implement control synchronization mechanisms within the FreeBSD kernel. [Rice]
- Explore the scalability limits of data synchronization and control synchronization. [Rice]
- Quantify the scalability limits and determine the bottlenecks of each scheme for publication. [Rice]
- Implement an Infiniscale-based testbed for exploring various techniques for protocol offload. [UNM]
- Use this testbed to implement various levels of intelligent NICs to quantify the costs and benefits of different levels of protocol processing, including NICs that do only unreliable data transport (e.g. Ethernet), NICs that provide RDMA in addition to simple Ethernet-like semantics, and NICs that provide more general early demultiplexing capabilities. [UNM]
- Compare the existing Infiniband protocol stack with a TCP/IP stack over different virtual NICs to quantify the potential costs and benefits of the custom Infiniband protocol stack. [UNM]
- Analyze NAS and ASCI Purple benchmarks to determine the amount of protocol overhead currently seen by the application and the amount of time available for hiding protocol processing overhead in blocking application calls. [UNM]

FY07-FY08:

- Implement an FPGA-based network interface to enable the exploration of alternative communication mechanisms between the operating system and the network interface. [Rice]
- Explore and implement improved communication mechanisms between the operating system and the network interface to improve the scalability of both the data synchronization and control synchronization schemes. [Rice]
- Experimental measurement of the value of splintering to HPC applications. [UNM]
- Performance comparison of full protocol offload, splintering, and demultiplexing-only offload. [UNM]
- Dynamic protocol offload and splintering based on measured application and protocol performance. [UNM]

3.1.4 Relevance to Weapons Program

- The proposed testbeds will drive the design of next generation networking hardware.
- Commodity clusters are vital for providing the necessary computation capacity for the weapons program. Improving the communication performance of commodity networks for these clusters will significantly increase the performance/price ratio of these systems.
- Multi-thread, multi-core processor architectures are rapidly emerging as the dominant architectural organization. Improving the scalability of network processing is critical in order to exploit the performance improvements provided by these architectures.

3.1.5 Budget

Total Proposed Budget: $153,029

Breakdown by Site: Rice: $61,529; UNM: $91,500

3.2 Fault Tolerant Messaging

Supported Personnel:

Project Leads: Jack Dongarra, UTK, dongarra@cs.utk.edu; Arthur Maccabe, UNM, maccabe@cs.unm.edu

Other Supported Personnel: *UNM:* Patrick Bridges

***LANL Point of Contact**:* Tim Woodall, twoodall@lanl.gov

3.2.1 Vision

As systems are constructed from larger numbers of components, long running computations will need to become much more tolerant of hardware faults encountered during their execution. In this proposal we focus on the faults that are observed during message passing and propose to explore two dimensions of this space. First, we consider approaches to masking transient communication failures and, in particular, the correct placement of protocol processing to mask these failures and provide a reliable message transport. Second, we consider process fault tolerance at the application level.

A Framework for End-to-End Reliable Message Transport in Open MPI

Current MPI implementations make static design tradeoffs between end-to-end reliability and performance, emphasizing one over the other. Emphasizing performance alone assumes a completely reliable network, a potentially unrealistic assumption in the case of newer, unproven networking technologies. Over emphasizing the end-to-end principal may result in duplicating a completely reliable transport protocol in application space even when the network experiences very low bit error rates. End-to-end reliability comes at a cost and this cost should directly reflect the operating environment. Our goal is to develop a flexible reliability framework for Open MPI that allows system and application designers to choose the reliability/performance tradeoff appropriate to their hardware and software configuration.

There are currently no MPI implementations that address both end-to-end data reliability and adaptable network failover. MPI designers are influenced first and foremost by performance considerations. End-to-end data reliability and network failover may be considered a high cost item, unnecessary in the presence of reliable network interfaces. Such a viewpoint disregards the end-to-end principle, the long runtimes of many MPI jobs where bit error rates become a factor, and the early adoption of many unproven networking technologies in the HPC world. With this in mind, we have developed a reliability framework that allows the system and application designer to choose the appropriate level of reliability detection and recovery. This framework allows for heterogeneous network device failover and guarantees

end-to-end reliability. The framework is also adaptable to various network topologies, allowing varying granularity of error detection and recovery.

Open MPI is a community MPI implementation that is built using a flexible modular framework that allows complete subsystems to be chosen and changed at runtime. Our current design is based on several key Open MPI abstractions. In Open MPI, the Point-to-Point Transport Layer (PTL) is an abstraction of a single transport type, such as TCP over Ethernet or GM over Myrinet. The Point-to-Point Management layer (PML) is responsible for managing one or more PTLs. The PML is responsible for delivery of an MPI level message; the PTLs are responsible for delivery of fragments of the MPI message.

The first design decision was to move as much of the reliability as possible into the PML, which we have named RPML for (Reliable Point-to-Point Management Layer). This was done to simplify the PTLs as much as possible; implementing reliability in each PTL requires duplicate functionality to manage and adds a substantial hurdle for PTL writers to overcome. This choice does come at a cost, we must abstract the reliability layer to accommodate a variety of network transports, potentially loosing transport specific optimizations.

First we introduced an RPML abstraction, a Virtual Fragment (vfrag). Each MPI level message is segmented into one or more RPML level vfrags. The vfrag acts as both the unit of retransmission and acknowledgment, allowing the application designer or system administrator to choose the maximum vfrag size up to the size of MPI level message. Each vfrag is assigned to a single PTL and may be larger than the PTL's internal fragment size. This allows the PML a convenient method of monitoring the "health" of a single PTL in terms of number of dropped or corrupted vfrags. When the health of a PTL is in question (due to excessive dropped/corrupted vfrags), the PML may choose to retransmit the vfrag on another PTL if available. While this functionality has not yet been implemented, the overall architecture has been designed to allow for this. Each vfrag also has one or more checksums associated with it. This allows for meaningful error detection regardless of the size of the vfrag.

If the sender is using rendezvous, a list of vfrag headers is sent to the receiver along with the send request. On the match, the receiver initializes these vfrag descriptors. If the sender is using an eager send, the vfrag headers are sent with the first fragment of the message.

When a PML schedules some portion of the vfrag to be transmitted, a timer is set. This allows us to detect when a PTL is no longer making progress on the vfrag. On expiration of the timer, the vfrag is retransmitted. On scheduling of the last bytes of the vfrag, the PML sets an ACK timer, expecting an acknowledgment of the vfrag from the receiver. On receipt of the ACK, the timer is removed; on expiration of the timer, the vfrag is retransmitted. Retransmission of a vfrag requires a handshake or "reschedule request". This is required to allow the PML to potentially choose a different PTL to retransmit on, and the receiver must be notified of this.

3.2.2 Research plan

We propose to work on improving and developing the following subsystems within Open MPI and Open RTE (Run Time Environment):

- Tunable Collective Communications. We have developed several optimized collective communication packages (ATCC/OCC) together with testing and optimization harnesses. These packages allow for the optimal choice of collective communication algorithm and parameters depending on both the applications needs as well as the target systems parameters. This work will involve adapting the OCC algorithms to work within the current and future collective communication subsystems of Open MPI. Additional work will also include adapting SAIS (Scalable Application Instrumentation System) to OCC. (SAIS instruments communications in much greater detail than defacto instrumentation packages, and allows a collectives package to be

tuned in optimal time for a set of target 'production' applications, rather than exhaustively for all possible communications.

- Building a high bandwidth and low latency point-to-point driver for the IBM LAPI interface. This will allow Open MPI to be used efficiently on and across IBM Scalable Power systems. Work will also include a POE interface to Open RTEs resource management system.
- Low level profiling of (near hardware) point-to-point communication subsystems. This can be thought of as PAPI for RDMA based systems. Possible integration with the KOJAK system in order to allow the user to precisely identify the bottlenecks in the application communications.
- Design a message scheduler manager that is optimized for driving RDMA point-to-point subsystems to support MPI-2 Single sided communications efficiently. This work would be the layer above the LAPI interface (described in second bullet).
- Create a new message routing layer for the Open RTE system that provides for scalable fault tolerance and dynamic routing of messages in large-scale systems. Open RTE is responsible for startup, connection information transfer, state management, IO forwarding, etc. All of these functions require scalability and resilience in internal communication.
- Vertical fault tolerance. Most fault tolerant schemes are horizontal in their effect and action across all the processors during check pointing, restarting, etc. When combining more than one fault tolerant approach and method it becomes necessary to integrate the actions of the processes both horizontally across the entire application as well as within each application process vertically. Integrating the actions of the different layers will require a modular framework that both coordinates the different layers as well as makes choices about which combinations of methods are appropriate. We propose to build a framework and integrate process level fault tolerance (as in FT-MPI) with system level checkpointing (as used in LAM/MPI and MPICH-V).

3.2.3 Tasks

FY06:

- Design of a framework that allows developers to choose the appropriate tradeoff between reliability and performance.
- Integration of reliable transport protocols into Open MPI.
- Tunable Collective Communications.
- Building a high bandwidth and low latency point-to-point driver for the IBM LAPI interface.

FY07:

- Demonstration of the reliability framework, illustrating benefits for reliable and unreliable networks.
- Low level profiling of (near hardware) point-to-point communication subsystems.
- Design a message scheduler manager that is optimized for driving RDMA point-to-point subsystems to support MPI-2 Single sided communications efficiently.

FY08:

- Integration of fault masking and fault tolerance approaches.
- Vertical fault tolerance.

3.2.4 Relevance to weapons program

An integrated approach to fault masking and fault tolerance will be critical to the ability to complete long running computations on systems that are constructed from many thousands of components.

3.2.5 Budget

Total Proposed Budget: $101,500

Breakdown by Site: UNM: $54,900; UTK: $46,600

4. Computational Science

4.1 Advanced Numerical Methods for Diffusion Equations in Heterogeneous Media on Distorted Polyhedral Meshes

Supported Personnel:

Project Lead: Yuri Kuznetsov, UH, kuz@math.uh.edu

LANL Point of Contact: Misha Shashkov, shashkov@lanl.gov

4.1.1 Vision

The diffusion equation is one of the most frequently appearing partial differential equations in applications ranging from radiation transport and heat transfer to flows in porous media and fluid dynamics. A number of new discretization methods for the diffusion type equations have been invented and utilized recently. The mixed finite element and mimetic finite difference methods on arbitrary polyhedral meshes are among them. These methods were developed and evaluated on ASC relevant test problems by scientists at LANL and the University of Houston within the LACSI supported project.

Accurate discretizations and efficient iterative solvers for diffusion equations on distorted polyhedral meshes in heterogeneous media are a challenging problem. Major ASC codes utilize polyhedral meshes to provide flexibility for numerical simulation in complex shaped geometry. The faces of a distorted polyhedral cell usually are not flat. Then, a natural question concerns the minimal number of DOF (degrees of freedom) for the flux vector function and for the solution function (intensity, concentration, temperature, etc.) needed for the accurate discretization of the diffusion equation.

Another problem arises when the medium inside a polyhedral mesh cell is split into two different types of states, for instance, solid and liquid. Accurate discretizations for the case of mixed cells are not known. Important applications to the problems with free boundaries and internal interfaces are obvious, in particular, in the TELLURIDE project.

The physical solutions (density, intensity, etc.) and the solutions of the underlying diffusion equations are positive functions while the solutions in most of the discretization methods may have negative values. This contradiction is a serious problem for discretization methods.

Finally, the development of efficient preconditioned iterative solvers for the algebraic systems arising in polyhedral discretizations of the diffusion equations remains a very important topic for the ASC related applications.

4.1.2 Research Plan

To address the problems involved in obtaining accurate and efficient solution methods for diffusion equations in strongly heterogeneous media on distorted polyhedral meshes we propose the following topics for a three-year (2006–2008) research project. This plan is a natural continuation of the recent successful cooperation of numerical mathematicians at LANL and the University of Houston.

Polyhedral Discretizations with Minimal Number of DOF

The interfaces between neighboring polyhedral cells are usually approximated by sets of triangles. In the known discretizations, to guarantee the accuracy of the discrete solution we have to assign separate flux DOF for each of the triangles. For instance, for a distorted cubic mesh we assign four DOF for each of the interfaces between cells. Recently, it has been observed theoretically that only three independent DOF on an arbitrary interface are sufficient to guarantee the required accuracy. In the case of the distorted cubic meshes, we expect that only two independent DOF would be needed. In this project, we plan to develop, theoretically investigate, and evaluate on test problems relevant to ASC applications new polyhedral discretization methods with minimal DOF for the flux vector function on the nonflat interface boundaries between arbitrary shaped polyhedral cells.

Polyhedral Discretizations for Mixed Cells

Discretization methods for the diffusion equations with mixed polyhedral cells is a topic of growing practical interest. We propose to extend the recently developed polyhedral discretizations to the case when the diffusion tensor, the absorption coefficient, and the right-hand side in the diffusion equation are discontinuous inside polyhedral cells. We have several ideas regarding how the extension can be done, including a special homogenization procedure. The algorithms to be developed will be evaluated on test cases relevant to the TELLURIDE project at LANL.

Nonnegative Discrete Solutions to the Diffusion Equations

The solution function of a diffusion equation with a nonnegative source function and right-hand side functions in the boundary conditions is always a positive function. The solution of the discrete equations that approximate the diffusion equation on a polyhedral mesh frequently has negative values in the nodes where the solution is close to zero. Those values do not have physical meaning. In this part of the project, we propose to investigate several approaches to designing nonnegative discrete approximations to the positive solutions of the diffusion equation. These are

- to reformulate a discrete problem as a constrained minimization problem in terms of variational inequalities and to develop iterative algorithms for its implementation,
- to develop and investigate projection/repair algorithms for obtaining the discrete solutions, and
- to investigate the range of applicability of the existing polyhedral discretizations and their possible modifications to guarantee nonnegativity of the discrete solutions.

The algorithms to be developed are subject to preserving the discrete conservation laws.

Efficient Preconditioned Iterative Solvers for Polyhedral Discretizations

In this part of the project we continue our recent efforts to design and investigate efficient and robust preconditioners for the matrices arising in polyhedral approximations of the diffusion equations. Priority will be given to multilevel substructuring preconditioners with applications to problems in strongly heterogeneous media. The proposed preconditioners will be evaluated on test cases relevant to ASC applications.

4.1.3 Tasks

FY05 (These are from the existing project):

- Implement the proposed polyhedral discretization method and evaluate its accuracy and efficiency on 3D test problems relevant to ASC applications. Deliver a technical report with results of numerical experiments.

- Investigate convergence properties of the proposed methods and to derive a posteriori error estimation for polyhedral discretizations of the diffusion equations.
- Develop, investigate and evaluate on selected test problems the new multilevel preconditioner. Deliver report with description of preconditioners to be developed and results of numerical experiments.

FY06:

- Develop and investigate discretization methods with minimal number of DOF for 3D diffusion equations on strongly distorted polyhedral meshes.
- Evaluate the method with minimal number of DOF on test problems relevant to ASC applications and deliver report with results of numerical experiments.
- Develop, evaluate, and investigate algorithms of constructing nonnegative discrete approximations to the solutions of 2D diffusion equations.
- Develop and investigate polygonal discretization methods for 2D diffusion equations with mixed materials inside mesh cells.

FY07:

- Develop, investigate and evaluate new multilevel preconditioners for mixed finite element method with minimal number of DOF; deliver report with results of numerical experiments.
- Develop and investigate polyhedral discretization methods for 3D diffusion equations with mixed cells.
- Evaluate polyhedral discretization methods for diffusion equations with mixed cells on test problems relevant to ASC and TELLURIDE projects, and deliver report with results of numerical experiments.

FY08:

- Develop and evaluate efficient preconditioned iterative solvers for 3D diffusion equations with mixed cells; deliver report with numerical results.
- Develop and investigate algorithms for constructing nonnegative polyhedral approximations to the solutions of 3D diffusion equation in heterogeneous media.
- Evaluate algorithms for nonnegative approximations on test problems relevant to ASC applications; deliver report with numerical results.

4.1.4 Relevance to Weapons Program

- Polyhedral discretization methods with minimal number of DOF will be used in Project B to reduce the size of the systems of algebraic equations.
- Polyhedral discretizations for diffusion equations with mixed cells will be used in TELLURIDE and other ASC related projects to improve accuracy of discrete solutions.
- Nonnegative discrete approximations to positive solutions of diffusion equations are essential to provide the physical meaning of simulation results for several ASC related projects.

4.1.5 Budget

Total Proposed Budget: $150,000

Breakdown by Site: UH: $150,000

4.2 Validation and Verification Using Code-Based Sensitivity Methods

Supported Personnel:

Project Lead: Mike Fagan, Rice, mfagan@rice.edu

Unsupported Collaborators: *LANL:* Rudy Henninger

LANL Points of Contact: Rudy Henninger, rjh@lanl.gov; Ralph Nelson, ran@lanl.gov; Dave Korzekwa, dak@lanl.gov; Jim Sicilian, sicilian@lanl.gov; Doug Kothe, dbk@lanl.gov (administrative contact)

4.2.1 LANL Priority: Validation and Verification

Los Alamos develops and employs computer models of complex and dangerous physical phenomena to augment (and sometimes replace) experimental methods. Therefore, the correctness of the computer simulation results is crucial. Consequently, LANL has established a priority labeled 'Validation and Verification' to ensure the viability of computer results.

Based on the LANL Validation/Verification priority, the overall goal for this project is to develop and improve methods for the validation and verification of numerical software.

4.2.2 General Strategy to Achieve the Research Goal

Before elucidating our project strategy, we note that Validation and Verification is a very broad concern, and there are a host of methods and techniques that might be employed. Some Validation/Verification techniques may be applied in general cases. Other techniques are very specific. Consequently, Validation/Verification practitioners must employ a 'toolbox' approach.

With this observation in mind, our strategy for achieving the research goal is to focus our research effort on a particular set of tools in the Validation/Verification toolbox called 'code-based sensitivity methods'.

Briefly, code-based sensitivity methods combine techniques from program analysis, numerical analysis, and symbolic computation to generate *augmented* computer models that enable various validation and verification techniques.

One widely-used augmentation is automatic differentiation. Automatic differentiation tools augment users' programs with code that computes (true analytic) derivatives. This augmentation is extremely useful for Newton's method-based validation techniques. To date, Los Alamos has employed Adifor, an automatic differentiation tool for Fortran 77 programs, in Newton's method-based validation efforts for some hydrodynamics codes and some reactor safety codes. In addition, Adifor90, an automatic differentiation tool for Fortran 90, is being tested on a metal casting code.

4.2.3 3-Year Horizon

The 3-year horizon for this research expands on the success of Adifor-based validation efforts. We envision the results of the project in 4 major categories: Improved differentiation algorithms, improved language coverage, improved integration of code-based sensitivity with the software development process, and new sensitivity augmentations.

Improved Differentiation Algorithms

The current Adifor differentiation algorithms can be (and are being) improved in the following areas:

- better automation of the recompute vs. store for adjoints,
- better detection and use of sparsity,
- detection of symmetry (self-adjoint) and linearity,
- improved higher order derivative methods,

- better handling of pointers and dynamic memory management, and
- direct augmentation of macros and templates.

Improved Language Coverage

We would like to cover more of the programming languages used by designers. The most popular requests are C/C++, Python, and Java. In addition, we would like to cover assembly language so that 3rd party code (without source) can be validated and verified as well.

Improved Integration of Code-based Sensitivity with the Software Development Process

We would like to integrate code-based sensitivity into the software development process, probably as a plug-in to a programming environment. For example, the integrated environment could enable:

- unit tests verifying that roundoff error for the test cases is acceptably low,
- method of manufactured solutions (MMS) unit tests that use differentiation-enabled forcing functions, and
- Newton's method-based validation test suites that are automated.

New Sensitivity Augmentations

Besides derivatives, there are many other sensitivity measures used in validation and verification. We plan on implementing code-based sensitivity calculations for the following measures:

- *Intervals*. Interval methods compute provable upper and lower bounds for a computation. Code-based methods are not well explored. Some of our current research effort constitutes a preliminary investigation, but much more needs to be done.
- *Stochastic Calculus*. As an analog to classical calculus, mathematicians have developed calculi for stochastic processes. For example, the Ito calculus handles random walks. Similarly, the Mallevin calculus handles more general processes.
- *Statistical Expansion*. As an analogy to Taylor series, random simulations employ other expansions to determine sensitivity properties of functions. One such example is the Edgeworth expansion. We propose to augment programs with statistical expansions, in the same way that classical differentiation can be seen to augment programs via Taylor series.

4.2.4 Proposed Research Activities

In general, our research activities proceed in 3 steps:

1) determine the mathematical algorithm appropriate for the augmentation under study,
2) implement the step 1 algorithm in our Adifor software framework, and
3) deploy the step 2 prototype tool on a LANL problem of interest.

The specific research areas that we plan to work on in the near future are as follows:

- roundoff error calculations using forward mode;
- backward error analysis using reverse mode;
- differentiation of simulations that use random number generation as part of the computation, (Treatment of stochastic simulation is an open problem in AD. Deterministic, Ito calculus, and stochastic expansion (like Edgeworth) are all possible candidates.); and
- interval analysis, in particular, applying compiler analysis techniques to improve the error bounds of classical interval analysis.

4.2.5 Tasks

FY05 (Current Year):

- Apply forward mode AD to a (subset of) Truchas metal casting application.
- Deploy Adifor 90 to LANL-specified platforms.

FY06:

- Finish forward mode differentiation of remaining Truchas code.
- Conduct reverse mode AD on a (subset of) Truchas, using improved recalculation.
- Develop and deploy a preliminary AD tool for C & C++ (Get LANL help for pilot project).
- Develop and deploy a preliminary AD tool for Python (Get LANL help for pilot project).
- Initiate a roundoff error bound augmentation pilot project (LANL help).
- Begin investigation of programming environment integration (eclipse plug-in?).

FY07:

- Continue development of C, C++, Python AD tool suite.
- Initiate a pilot project for object code differentiation.
- Begin a study for stochastic calculus via AD.
- Continue roundoff error bound augmentation.
- Begin automatic backward error analysis project.
- Continue programming environment investigation.

FY08:

- Continue development of C, C++, Python AD tool suite.
- Continue stochastic calculus augmentation.
- Continue object code augmentation.
- Preliminary interval augmentation.
- Continue roundoff error bound augmentation.
- Continue programming environment investigation.

4.2.6 Relevance to Weapons Program

Scientists and engineers in the weapons program use computer modeling to study and certify (almost?) all aspects of weapon design and performance. This implies that the computations of the various weapons modeling codes must be trustworthy. Consequently, validation and verification of weapons modeling codes is crucial to the weapons program.

This project directly addresses the validation and verification concern. In particular, this project focuses on a specific set of validation and verification techniques called “code-based sensitivity”.

4.2.7 Budget

Total Proposed Budget: $119,384

Breakdown by Site: Rice: $119,384

5. LACSI Management and Community Interaction

Supported Personnel:

Management Leads: Ken Kennedy, Rice, ken@rice.edu; Linda Torczon, Rice, linda@rice.edu

Unsupported Collaborators:

Management Leads: Andy White, LANL, abw@lanl.gov; Rod Oldehoeft, LANL, rro@lanl.gov

LANL Point of Contact: Rod Oldehoeft, rro@lanl.gov

LACSI has a strong record of productive collaboration between academic researchers and LANL/ASC researchers. With the advent of the WSR program, and the reorganization of the Simulation and Computer Science Program in ASC, ongoing attention needs to be paid to keeping these interactions strong and relevant to ASC. Operational aspects also need to be maintained. Activities in this area include:

- maintaining contract operations,
- promoting interactions between LANL and the LACSI academic sites,
- improving the visibility of LACSI as a whole, and
- keeping the LACSI management structures operating.

The remainder of this section describes the LACSI management structure and planning process, plans for community interaction, proposed tasks, relevance to the weapons program, and budget requirements.

5.1 Management and Planning Process

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

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

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

The EC meets either in person or by teleconference quarterly and communicates regularly by e-mail. The EC also meets in person at the LACSI Symposium every fall and at the spring review and planning meeting. In FY05, LACSI EC meetings were held on October 13, 2004 (Santa Fe, NM) and February 7, 2005 (Los Alamos, NM). In FY06, the LACSI EC meetings will be held on October 12, 2005 (Santa Fe, NM) and February 8, 2006 (Los Alamos, NM).

The LACSI directors, the LACSI executive directors, and key research and administrative personnel from LANL and Rice meet by teleconference at least once per month and correspond by e-mail to handle administrative matters related to proposals, contracts, reviews, invoicing, meeting arrangements, reporting requirements, and other administrative issues that arise.

LACSI members also communicate via project planning documents, statements of work, and the results of meetings and reviews, which are posted on password-protected pages on http://lacsi.rice.edu.

5.1.1 LACSI Annual Planning and Research Cycle

The annual planning and funding cycle consists of an annual review followed by a planning meeting (called the *Priorities and Strategies Meeting*), followed by the preparation of a proposal for funding for the entire Institute. This proposal will be submitted to the appropriate research funding entity; under the current plan this is *Weapons-Sponsored Research (WSR)*.

Annual Review

The LACSI annual review will be conducted by the LACSI Review Board, which will consist of members selected by LANL in consultation with the LACSI Executive Committee. The LACSI Review Board will include members who are both external and internal to LANL; in addition it will include representation from customers as well as computer and computational scientists. Review Board members will serve terms of three years, with approximately a third of the board starting new terms each year. The LACSI Annual Review will typically be conducted in February.

The goals of the Annual Review will be to provide feedback to ongoing projects concerning their overall direction and approach, and to review proposals for new projects. New projects that are approved will be folded into the proposal for the next fiscal year, contingent on the availability of funds. To assist in the process of allocating limited funds, the LACSI Review Board will be asked to prioritize the proposed projects. All reviews will be based on two inputs: a proposal (for new projects) or a progress report (for continuing projects), and presentations made during the review.

LACSI Priorities and Strategies Meetings

In March 2002, the LACSI EC met with LACSI researchers at LANL to discuss methods of addressing issues raised in the 2001 LACSI contract review. The group developed a framework to address long-term strategic thrust areas. Specific objectives were called out as near-term priorities. The objectives were folded into the framework to form a coherent planning view. A description of the long-term vision, framework, and objectives developed in 2002 is available in a document (LAUR # 02-6613) titled *Priorities and Strategies*. Since 2002, the LACSI EC and Principal Investigators have met annually with

senior LANL personnel to revise the framework, priorities, and strategies established at the 2002 planning meeting. The results of each meeting have been incorporated into *Priorities and Strategies*. The results of the April 8-9, 2003 planning meeting at Rice are included in LAUR # 03-7355; the results of the February 19-20, 2004 planning meeting at Rice are captured in LAUR # 04-7982. Relevance to the LACSI priorities and strategies outlined in the document continues to be a key evaluation criterion used when the EC evaluates progress on LACSI projects.

In February 2005, the LACSI EC met with senior LANL personnel to incorporate feedback from the 2004 LACSI contract review in to the planning process. The results of the February 7-8, 2005 planning meeting, which was held at LANL, were folded into the LACSI proposal for the next fiscal year.

Future LACSI Priorities and Strategies meetings will typically be conducted in February immediately after the LACSI annual review. The goal of those meetings will be to fold feedback from the LACSI review board into the LACSI planning process, and to prepare a LACSI proposal for the following fiscal year.

LACSI Proposal and LACSI Academic Statement of Work

Each year, the Executive Committee for the Los Alamos Computer Science Institute will produce a proposal for funding for the next fiscal year. Preparation of this proposal will begin with the Priorities and Strategies meeting. After that time the proposal will be developed into final form over the following 2-3 months for submission to a research funding program (e.g, WSR) in May.

Contingent on budget, the LACSI proposal will be funded as a whole. If the final budget is significantly less than proposed, the Executive Committee will collaborate with representatives of the funding program on a reduced statement of work that will constitute the final plan for the next fiscal year's activities. This plan will be adapted to form the statement of work for the academic contract and the appropriate input to the planning process for LANL as a whole.

The academic statement of work will be finalized by August and funding will be in place in time for the start of the new fiscal year on October 1. Ideally, a standardized mechanism, such as the currently proposed "task order" strategy, will streamline the process and avoid the necessity for a new contract to be negotiated each year, which has been a major source of problems in the past.

Note that LACSI will be funded as a single proposal rather than a selection of projects. This is in keeping with the partnership model described earlier. Reductions in statement of work due to budget considerations will be made by the Executive Committee in consultation with representatives of the funding program.

5.2 Management of Academic Contract and Subcontracts

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

5.2.1 Computational Resources

The academic partners will be provided with access to ASC computing platforms at LANL on a predetermined basis for development and testing. The process will make it possible to allocate a small cluster of nodes each week and a larger cluster of nodes once a month. It is understood that dedicated access may be needed for key tests and performance analyses. To the extent possible, development work will be centered on the Clustermatic testbed systems installed at Rice, UNC, and UNM. Access to large Clustermatic systems at LANL may be required for full-scale tests.

5.3 Community Interaction

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.

During the next year, LACSI members will continue to actively engage in activities associated with high-performance computing conferences, particularly SC '05. Participation will include some combination of research paper and poster presentations, exhibits and presentations in research booths associated with LACSI sites, workshops, and birds-of-a-feather (BOF) sessions. These activities will focus either on presenting LACSI research results to the high-performance computing community or on engaging the high-performance computing community in discussions of topics relevant to LACSI's research mission.

LACSI will continue to update and maintain a project Web site (http://lacsi.rice.edu) where members of the computer science community can peruse descriptions of LACSI projects, access both a list of project publications and a list of academic project participants, and download software produced by LACSI. Information about the LACSI symposium and other LACSI events is also available from http://lacsi.rice.edu.

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.

5.4 Tasks

- Work with LANL SUP to keep the academic contract on track, and handle budgetary change requests.
- Organize at least one specialized meeting each quarter at LANL between academic and lab collaborators.
- Increase the number of summer students at LANL from collaborating universities
- Organize and operate the annual LACSI Review and Priorities and Strategies meeting at LANL.
- Organize and operate the annual LACSI Symposium in Santa Fe, including publicity, planning, and follow-up special journal issue.

5.5 Relevance to Weapons Program

The LACSI contract with WSR will have components that have been selected for their technical excellence and relevance to the weapons program. This component keeps the academic research focused and maintains the technology transfer opportunities to the benefit of the weapons program.

5.6 Budget

Total Proposed Budget: $237,059

Breakdown by Site: Rice: $237,059

6. Budget Summary

FY06 Budgets by Institutions and Project			
1.1 Integrated Performance and Reliability of Extreme-scale Systems	Rice		216,016
	UNC		233,000
	UNM		36,600
	UTK		93,200
		Total	578,816
1.2 Compiler Technology for High-Performance Computing	Rice		346,925
		Total	346,925
2.1 Component Architectures for High Performance Computing	Rice		213,080
	UH		55,500
	UTK		46,600
		Total	315,180
2.2 Automatically Tuning and Retargeting High-Performance Components and Libraries	Rice		188,107
	UH		55,500
	UTK		46,600
		Total	290,207
3.1 Scalable Optimized Network Protocol Stacks	Rice		61,529
	UNM		91,500
		Total	153,029
3.2 Fault Tolerant Messaging	UNM		54,900
	UTK		46,600
		Total	101,500
4.1 Advanced Numerical Methods for Diffusion Equations in Heterogeneous Media on Distorted Polyhedral Meshes	UH		150,000
		Total	150,000
4.2 Validation and Verification Using Code-Based Sensitivity Methods	Rice		119,384
		Total	119,384
5. LACSI Management and Community Interaction	Rice		237,059
		Total	237,059
FY06 Total Budget			**2,292,100**