Proposal for Program Announcement LAB 01-07
SciDAC Integrated Software Infrastructure Centers

March 7, 2001

Scalable Systems Software Enabling Technology Center

Al Geist (Center Coordinator)
Oak Ridge National Laboratory
Work: (865) 574-3153
FAX: (865) 574-0680
gst@ornl.gov

Principal Investigators:

Stephen Scott - Oak Ridge National Laboratory
Rusty Lusk – Argonne National Laboratory
Paul Hargrove – Lawrence Berkeley National Laboratory
Rob Pennington – National Center for Supercomputer Applications
Brett Bode – Ames Laboratory
Scott Jackson – Pacific Northwest National Laboratory
Neil Pundit – Sandia National Laboratories
Ron Minnich – Los Alamos National Laboratory

Field of Research:

Operating System Software and Tools

Scalable Systems Software
Enabling Technology Center

Executive Summary: In order to address the lack of software for the effective management and utilization of terascale computational resources, we propose the creation of a Scalable Systems Software Enabling Technology Center. The virtual center will be a multi-institution, multi-disciplinary group composed of experts from around the country working as single team to develop an integrated suite of machine independent, scalable systems software components needed for the Scientific Discovery through Advanced Computing (SciDAC) initiative. The goal is to provide open source solutions that work for small as well as large-scale systems.

High-end systems software is a key area of need on the large DOE systems. The systems software problems for teraop class computers with thousands of processors are significantly more difficult than for small-scale systems with respect to fault-tolerance, reliability, manageability, and ease of use for systems administrators and users. Layered on top of these are issues of security, heterogeneity and scalability found in today's large computer centers. The computer industry is not going to solve these problems because business trends push them towards smaller systems aimed at web serving, database farms, and departmental sized systems. In the longer term, the operating system issues faced by next generation petaop class computers will require research into innovative approaches to systems software that must be started today in order to be ready when these systems arrive.

The creation of a Scalable Systems Software Center is a critical, long term investment that will benefit not only the DOE high-end computational science and simulation needs of SciDAC, but also other government agencies with large-scale computer centers such as DOD, NSF, and NASA. In addition, the software produced by the Center will provide many benefits for the myriad of smaller scale systems as well as providing a basis for the eventual adoption and deployment of these technologies in the commercial marketplace, as has happened with the smaller scale clusters currently being deployed.

The Scalable Systems Software Center will do the research for and produce an integrated suite of systems software and tools for the effective management and utilization of terascale computational resources particularly those at the DOE facilities. The first step in this process will be to work together with vendors and system administrators to specify an agreed upon set of interfaces between the system software components. The Center will make a standard software distribution available as open source based on these standard interfaces. Wherever possible the components will be machine and operating system independent.

In addition to the standard systems software distribution, the Center will be continually involved in the research and development of more advanced versions of the components as well as OS modifications required to support the scalability and performance requirements of SciDAC applications. Activities within the Center will often be focused on the most critical system software needs of SciDAC's *flagship*, *topical,* and *experimental* computing centers.

As part of its software lifecycle plan, the Center will engage vendors such as IBM, Compaq, SGI, Scyld, and other major Linux vendors to extend the Center's system software distribution to their platforms and support the specified component interfaces on their systems.

1.0 Introduction:

Presently, one has the ability to physically connect a virtually unlimited number of computers into what appears to be one monolithic machine. However, the lack of scalable software systems and environments has kept us from effectively exploiting the power of such a configuration. The difficulty of dealing with tera-scale systems has relegated those few machine builds to large research facilities with a staff to match. One goal of this Scalable Systems Software Center is to build an environment whereby a cluster of machines may scale to a very large physical size without the requirement that the staff scale along with the machine. The identification and development of key system components that enable this scalability is one step in this goal.

Improved scalability is not an optional feature: it is a mandatory requirement. At current system growth rates, average high-end machine size will exceed 8000 processors by 2003. The flat, serial communication model used by existing resource managers is already being stressed by machines with only a few hundred processors and will fail well before they reach a few thousand. The problem lies in the fact that the amount of data, the number of connections, the amount of time spent handling failure conditions, and the like currently all scale linearly with the number of nodes. A vastly improved system architecture needs to be developed, based on a robust, possibly hierarchical infrastructure. This system would not only scale logarithmically, but will also be designed to intelligently compress state and resource configuration data. Additionally, node and communication failures need to be automatically handled and routed in a non-serial manner preventing failures from bottlenecking communications performance. Security must be incorporated into the system components and interfaces to prevent unauthorized access to resources and sensitive information. In order to be of maximum benefit to the high performance technical community, these programs should be freely distributed, open source where possible and have proper documentation and support.

The importance of operating systems and tools research and development for high end scientific computing has been identified in the PITAC report [1]. However, the nation currently does not have software infrastructure in place to ensure that the near term issues related to multi teraop operating systems and tools and the longer- term requirements of petaop scale operating systems and tools are addressed. The technology trends and business forces in the United States computer system industry have resulted in radically reduced development and production of systems targeted at meeting the most demanding requirements of scientific research and government mission agency applications. In essence, the US computer industry has become focused on computer hardware and software needs of business applications and smaller scale scientific and engineering problems. Minimal attention is paid to the special computational needs of the high-end scientific community. Consequently, achieving the performance levels required for agency missions and world leadership in computational science currently requires combining large numbers of smaller systems to produce ultra scale (teraop and above) computers. The problems faced by system software and tools for teraop class computers are significantly different from the solutions provided by today's system software. In addition, the operating system issues faced by next generation petaop class computers are extremely difficult and require innovative approaches to system software in order to achieve high-sustained application performance.

To address this gap in the continued progress of high-end computing, it is necessary to begin creating scalable systems software components, which ideally work for small as well as large-scale systems.

This proposal goes beyond just creating a bunch of separate scalable components. By defining an architecture and interfaces between system components, the Scalable Systems Software Enabling Technology Center (ETC) will provide an interoperable framework for the components. This makes it much easier to adapt, update, and maintain the components, in order to keep up with improvements in hardware and other software. Publicly documented interfaces are a requirement because it is unlikely that any package can provide the flexibility to meet the needs of every site. So, a well-defined API will allow a site to replace or augment individual components as needed. Defining the API's between components across the entire system software architecture provides an integrating force between the system components as a whole and improves the long-term usability and manageability of terascale systems at computer centers across the country.

The vision and goal of the ETC is to bring together a team of experts with the single goal of creating an integrated suite of scalable systems software and tools for the effective management and utilization of terascale computational resources particularly those at the DOE facilities. To reach this goal a significant amount of research and development must be performed. To keep on track we have divided this work into four steps:

1. **Collectively (with industry) agree on and specify standardized interfaces between system components** in order to promote interoperability, portability, and long-term usability.
2. **Produce a fully integrated suite of systems software and tools** for the effective management and utilization of terascale computational resources particularly those at the DOE facilities.
3. **Continue research and development of more advanced versions of the components** required to support the scalability and performance requirements of SciDAC applications.
4. **Carry out a software lifecycle plan** for the long-term support and maintenance of the reference systems software suite.

2.0 Benefits to DOE

This proposal has significant long-term relevance to Office of Science related problems. DOE operates many of the largest computers in the world and some of the largest computer centers. But today each computer center uses ad hoc and homegrown systems software solutions. When a problem is solved at one DOE computer center today, there is no way to leverage that solution at the other centers. This Scalable Systems Software Center provides the opportunity to create and support a common set of systems software for large computer centers across the country. Thus allowing the sharing of solutions and/or improved components between sites. It will also provide components that some sites don't presently have such as an integrated system monitor.

By defining standardized interfaces, the ETC provides a means for computer centers or vendors to create custom versions of components that remain interoperable with all the other system software provided by the ETC.

An important benefit of this ETC is that it will create useful software throughout its lifetime beginning within a few months after it is started. There is a critical need for scalable system software and even before interfaces are finalized, key components will be made available as prototypes.

The successful completion of this ETC's goal will fundamentally improve the way systems software will be developed in the future by having well-defined, extensible interfaces and a component architecture. In the long term the standardized interfaces may be the most important output of this ETC because these interfaces are independent of any particular implementation and can live on beyond the lifetime of the ETC.

A final benefit of the ETC is to provide a long-term system software maintenance and support solution. The benefit to computer centers is they are not at the mercy of companies going out of business and potentially leaving the centers with a software infrastructure that no longer can be maintained. Instead, the center can switch to another interoperable component from another vendor or use the open source reference implementation supplied by the ETC.

3.0 Background

System software comprises a broad array of components including those for configuration management, system monitoring, resource management, and the like. Figure 1 shows the primary components of a system software suite required by a terascale computer center. Equally important is need for all these components to interoperate with each other. By defining standard interfaces between the components it becomes possible for an organization to, for example, swap in a local policy accounting system without affecting its interoperability with the rest of the systems suite. A key part of this ETC will be to create an interoperable suite of scalable system components. The first step in this process is to understand the present state of systems software.

One of the most complex challenges in managing multi-user high performance computers is how to efficiently assign the available computational resources (CPU, memory, disk, network) to the users that need them in a manner that ensures fairness, ease of use, quick turnaround, and maximum utilization of the resources. Resource management consists of a collection of components that must work in concert to provide end users the collective resources they need to solve their problem in a timely fashion. Among the components are the scheduler, the queue manager, the job manager, the accounting system, the meta-scheduler and the user interface. The scheduler enforces the site-dependent policies and priorities and decides when, where and how jobs should best be run given these constraints. The resource manager is responsible to collect the resource and job configuration and state information needed by the scheduler to make intelligent decisions. The job manager is responsible for efficiently starting, steering, and terminating jobs. The accounting system encompasses both allocation management and the tracking of resource usage. The user interface is responsible for presenting the current state of the system to the user in such a way that the user can tune a request in order to receive the needed resources in the fastest possible time. Finally the meta-components provide the capability to build hierarchical software systems. They also provide users with a single interface into multiple systems distributed across a computer center.

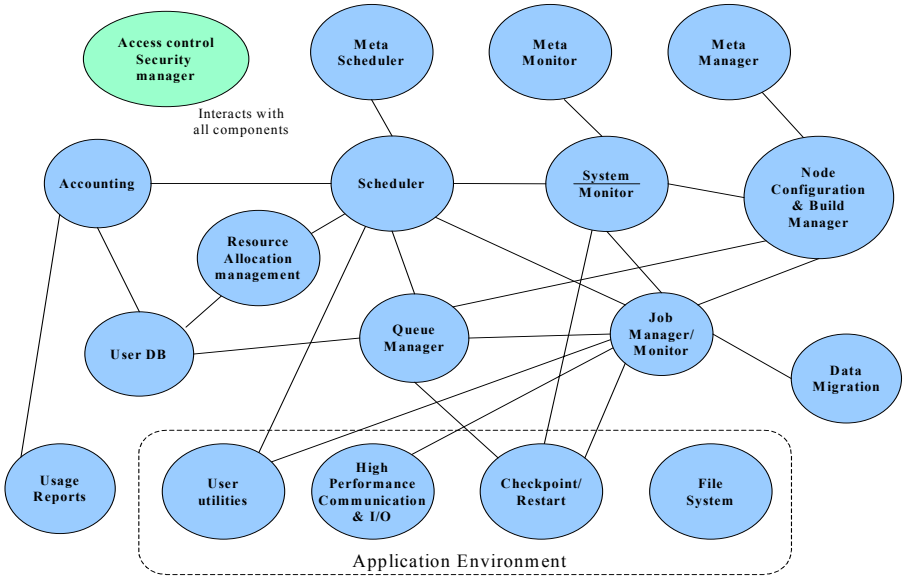

Figure 1. Prototype Architecture for System Software Components and Interfaces

Each of the components shown in Figure 1 serves an important function, but perhaps equally important is the need for them to work together as an integrated system. This diagram presents the major pieces of a systems software effort as well as an initial representation of the interfaces that would allow the components to work together. Two components in Figure 1, high performance communication and file systems, are addressed in other DOE funded projects and thus are not being addressed in this ETC except by their use and in their interfaces to other system components. The current trends in HPC technology require that several significant cross-cutting aspects of the design of the system software be addressed. These are not feature enhancements but rather are areas that are critical to effective utilization of the HPC clusters. These requirements include scalability, robustness, security, and SMP support. The present state and challenges for the pieces shown in the diagram are described next.

Resource manager

The role of the resource manager is to control and manage the local compute resources, whether they be compute nodes, networks, storage systems, etc. It provides an interface to allow job submission, job execution and job and resource tracking. It is also responsible for maintaining access control to the various compute resources.

A number of commonly used resource management systems exist today for MPP systems. These include PBS [2], LSF [3], Loadleveler, DQS, and Condor [4]. Each resource manager has its own particular set of strengths and weaknesses. Although some vendors have provided marginally acceptable resource managers for their platforms, currently available resource management solutions lack critical features needed in the present generation of high performance technical computing systems, such as ultra-high scalability (thousands of nodes), security, robustness, suspend/resume capabilities, and resource utilization and tracking support for clustered SMP nodes. Suspend/resume and/or checkpoint/restart preemption capability has a tremendous potential for dramatically improving

system utilization by eliminating the currently unutilized cycles where nodes remain idle in order to enforce fairness, prevent job starvation or preserve interactive response guarantees, but existing resource managers have only limited checkpoint/restart capabilities.

One of the more complete systems currently available is the IBM LoadLeveler software developed originally for the IBM SP series machines. Unfortunately LoadLeveler does not presently support Linux and is widely considered to have many technical shortcomings, in particular scaling problems on systems with large numbers of nodes (i.e., 512 or more).

The Portable Batch System (PBS) is an open source package originally developed for large SMP type computers as a joint project between the Numerical Aerospace Simulation (NAS) facility at the NASA Ames Research Center and the National Energy Research Supercomputer Center (NERSC) facility at Lawrence Livermore National Laboratory. Currently Veridian is coordinating the development and distribution of PBS. The focus of development has shifted to clusters and basic parallel support has been added. David Jackson at PNNL has ported the Maui scheduler to act as a plug-in scheduler to the PBS system. While this combination is proving successful at scheduling jobs on moderate sized parallel systems, PBS was not designed for a cluster-like computer and thus lacks many important features. The general consensus is that the PBS design will not scale effectively beyond moderate sized cluster systems. The modifications needed to fully meet the requirements stated above would necessitate a rewrite of a large portion of the PBS system and thus require a significant development effort. It does not appear that Veridian will put forth the effort required to build such a robust environment for clusters beyond that typically used in industry.

Job Manager

The job manager component is responsible for starting the processes that make up a parallel job once the job has been scheduled and the necessary resources allocated to it. The job manager is responsible for allowing a parallel job consisting of multiple processes on multiple hosts to be treated as a single entity, so that it can be suspended, continued, or terminated collectively as if it were a single process. Upon termination, all resources allocated for a job, such as temporary files, shared-memory segments, forked processes, etc. must be reliably freed, even if the job terminates abnormally. The system might also keep track of those resources that cannot be released (such as swap, local disk, pinned memory, licenses, etc) to prevent node over-allocation as well as to manage/renew authentication credentials (i.e., DCE/Kerberos5). It is responsible for managing standard input and output in an efficient and scalable way and providing environment variables and command-line arguments to the processes it launches. It can provide services to a running parallel job, such as dynamic resizing, checkpoint/restart, or job steering. The job manager must, like other components of the system, be secure, scalable, and fault tolerant.

The startup of parallel tasks must be done quickly enough to make large interactive jobs feasible. We envision the job manager as a set of persistent daemons, permanently in communication with one another, and available to communicate asynchronously with other components such as the scheduler and queue manager. (Other architectures are possible but less scalable.) The Job Manager itself may consist of separate components to implement security and robustness. For example, the Job Manager might derive its fault tolerance from another daemon that knows how to restart the Job Manager

daemon. It might be enhanced to interface a variety of parallel libraries such as MPI or PVM. MPD [5], designed and developed in part by Rusty Lusk at ANL, is an implementation of a job manager that provides both fast startup of parallel jobs and a flexible run-time environment that supports parallel libraries through a small, general interface.

Scheduler

The local scheduler is responsible for mapping a sites mission objectives into various scheduling decisions. It determines which jobs can run, when and where. It makes optimizations based on local policy and resource information. It coordinates its scheduling decisions with the allocation manager to verify the validity of all compute resource consumption and interfaces to the resource manager to enforce the decision it makes. It also interfaces with the meta-scheduler to provide local resource availability and to accept and manage remote jobs.

The Maui Scheduler [6], developed at MHPCC by David Jackson, has gained wide acceptance as an exceptionally versatile and performance-boosting scheduler available to large parallel clustered systems. It is open-source and is ported to several UNIX platforms including Linux and already contains many of the requisites for large-scale resource management. We believe that further development of the Maui scheduler presents the best path to address the requirements for the scheduling component. Significant gains could be achieved through continued development efforts such as suspend/resume preemption and parallel checkpoint/restart support, resource utilization limit enforcement (memory, disk, %CPU), extended resource scheduling (network, data, licenses), enhanced Quality of Service specification and priority management, and job migration.

Checkpoint/Restart

The ability to perform a system-initiated checkpoint of the running workload on a system, and later restart that workload, provides many benefits to the site administrator. The typical DOE user community, such as that of the LBL-NERSC center, includes a large number of users with applications of all sizes in various stages of development. During early code development, users generally require rapid interactive turn-around for test jobs on a small number of processors. As the code matures, larger problems are run on more processors and for a greater duration. At the far end of the spectrum, users want to run very large applications on all the nodes of the system for long periods of time. Job scheduling with fair resource allocation is nearly impossible without some means of multiplexing the large, long running jobs with the smaller and shorter running development jobs. On the NERSC T3E, checkpoint/restart and gang scheduling have provided the tools necessary to achieve sustained 90-95% utilization with relatively low queue wait times. The site uses checkpoint/restart to schedule multiple workload configurations throughout the week. Large, long running jobs are scheduled during evening and weekend hours while smaller and interactive jobs are run during the workday. System initiated checkpoint/restart allows the administrator to checkpoint one workload to disk and start another. Checkpoint/restart also allows the site to perform needed hardware or software maintenance and testing with minimal impact on the user community.

We propose to evaluate the technical issues associated with performing system-initiated checkpoint/restart on a Linux cluster then implement such a system. We will restrict our focus to parallel MPI applications under the control of the job manager.

Allocation Manager

As the high performance scientific computing resources grow in scale and capability, there is an increased need to manage the allocation of these resources for a large number of users. The site management needs a means to fairly distribute the underlying computing resources (processors, memory, disk) to the various users or projects that have access to them. This introduces the need for a resource allocation management tool, often referred to as an allocation manager or an allocation bank. The allocation manager ensures that users and projects use only the resources allocated to them. Without allocation management, projects and users would consume computing resources based solely on how aggressively they submitted their workload rather than based on any site managed policies and priorities. Besides simply allotting the total amount of a computing resource a user or project can use, the site management needs a way to set timeframes for the expenditure of these allocations. This capability is necessary in order to prevent year-end resource exhaustion when under spent projects simultaneously claim allocation fulfillment. It provides the capability of meting out the resources at a fair and predictable rate. By measuring project usage against allocated amounts it allows for insightful planning of how much more work can be supported by new projects as old ones expire in regular cycles.

There is currently no allocation management utility available that has gained wide acceptance in the high performance scientific computing community. By and large, sites have met this need by writing rather simplistic homegrown scripts that do little more than track project/account CPU usage in a periodic post-processing fashion. In order to fill the need for allocation management, Qbank [7] was developed at PNNL by Scott Jackson. QBank has been tested under both AIX and Linux. It is in production use at PNNL, MHPCC, University of Utah and is being installed at a number of other sites. Much like a bank, where the currency is measured in node-seconds instead of dollars, QBank provides an administrative interface supporting familiar operations such as deposits, withdrawals, transfers and refunds. It provides balance and usage feedback to users, managers and administrators. Some of QBank's existing capabilities include: dynamically expiring allocations and reservations, earliest credit expenditure, quality of service and nodetype billing, multiple accounts per user and multiple users per account, default accounts, security authorization, and database persistence. As part of this proposal we will develop and integrate QBank with the resource management suite of software.

System Configuration, Build, and Management

As HPC systems reach ever higher node counts, a much greater proportion of the system administrator's time and effort is consumed in activities related to building and updating the nodes, synchronizing their configuration and managing the system environment. It is a vital aspect of every cluster, yet most cluster users are unaware of its existence. No common set of software for configuration and node management yet exist. Furthermore, it is often difficult to separate the architecture of a cluster from the tools that build that architecture and the interfaces that support the rest of the system.

One of the challenges in creating a suite of build and configuration tools is that the philosophical approach to node management varies dramatically across sites. Specifically, when a new system is purchased, the managers of the system make a large number of choices about its architecture. Their decisions are based on requirements for their site and users, available technology, funding, and other issues. They must answer questions such as:

- hierarchical vs. non-hierarchical units of nodes
- local disks on each node vs. diskless nodes
- booting from standard BIOS vs. augmented or replaced BIOS
- single cluster-wide process space vs. many process spaces
- cloned node configuration vs. variable node configuration
- private vs. public internal network
- remote power controls or not
- remote serial console or not
- choice of processor technology
- choice of operating system

So, for example, a cluster that only has diskless nodes in it will require a different approach for node configuration management than one with fully-populated nodes. The list of these choices is extensive, and the result is that every large cluster in use today is substantially different from every other large cluster.

One of the reasons that so many of these choices exist is that these clusters are built from commodity parts using, generally, open source operating systems. Choices abound at all levels, from hardware to network to software. In contrast, every IBM SP site uses configuration tools that are similar to every other IBM SP site because the architectural and administrative model for IBM SPs is imposed by IBM as a natural result of their controlling the system. This wide variation of cluster systems, while making it very difficult to use common management tools between sites, has resulted in a rich set of cluster types for the end user, and has opened the door to further research in innovative management techniques and experimental cluster architectures.

One of the many challenges for this center is to develop a suite of common tools and approaches for node management that can both continue to support the wide range of choices yet also be used by many sites on many clusters. This will not be accomplished by making one tool that fits all systems—the history of systems administration has demonstrated that this approach does not work. Rather, it will be accomplished by defining the set of interfaces between the configuration management system and the other portions of the system software, and then developing a set of software tools that both conform to the interfaces and are flexible enough to match a range of architectural choices.

An additional challenge is to support large-scale systems. The configuration management system is the lynch pin between the hardware of the cluster and the system administrator. It can make the difference between a cluster requiring a ten-person team or a two-person team to manage it. Large systems of the future must be manageable by the same number of administrators (or fewer) as today's systems.

The members of this Scalable Systems Software Center are well-equipped to solve these problems. Sandia National Laboratory has demonstrated very large and highly-scalable cluster architectures and management approaches on Cplant. Oak Ridge National Laboratory is a part of the Open Source Cluster Application Resources project [8], giving them experience in defining cluster architectures for use at many sites. Further, ORNL operates HighTORC, a large cluster, using Cluster Command and Control, a suite of tools developed at ORNL to explore management capabilities. And Argonne National Laboratory has developed and released a set of tools called City [9], designed for flexible

management of a highly variable set of cluster architectures, as demonstrated on Chiba City, ANL's scalability testbed cluster.

System Monitoring

Monitoring is a very important yet undeveloped part of cluster management. Failure detection, location, and follow up tests all rely on the system monitoring tools available to the system administrators. The development of scalable tools for the detection of failures, and more importantly potential failures, in tera-scale clusters is a critical research area for the Scalable Systems Software Center. It is also a high pay-off area of research because many of the other system components rely on the information from the monitoring subsystem and because the present state of monitoring tools is low.

The state of the art of cluster node monitoring is extraordinarily primitive. The single most-used monitoring tool is the *ping* command: the command sends a simple packet to a remote node, and the remote node is supposed to reply. Typically, if a ping command fails, the only option is to reboot the node, via reset or power off. Once the node is rebooted, any information about the problem that precipitated the failure is lost.

Even if the ping command succeeds, the node may still be unusable due to other problems. A commonly-used problem determination sequence is as follows:

1. A user notices that a job has not completed in a normal way, and contacts the system administrator.
2. The sysadmin tries to log onto the node, which may or may not succeed. In many cases, even if the login succeeds, the sysadmin can not find a cause, and reboots the node as a prophylactic measure.
3. If the login fails, the sysadmin pings the node. Even if the ping works, there is no way to get onto the node, so the sysadmin reboots the node.
4. If the ping fails, then the only option is to reboot the node.

As this discussion shows, there are only two sensors in use in most clusters: the first sensor is the process that supports remote login; the second sensor is the process that responds to remote pings. These sensors are almost always used in a reactive manner, i.e., in response to perceived problems, and in many cases long after the failure that caused the error. Actually isolating the failure takes second place to getting the node back up and running again; the result is a lack of long-term data to allow precise problem determination.

For large-scale clusters these mechanisms are not sufficient. We need to move to proactive monitoring of cluster nodes, and we need to make much better use of the data that is available.
There have been some recent attempts at developing system monitoring tools. While not proactive in monitoring, VACM [10], an open source project from VAlinux, is one effort to produce a GUI based cluster hardware manager. This tool provides services such as direct monitoring of CPU fan speed, temperature, and voltage rating. VACM's most serious flaw to date has been that it could only be supported by the more expensive server class Intel Intelligent Platform Management Interface (IPMI) compliant motherboards. Other features require even more specific system capabilities such as Intel's Emergency Management Port (EMP). While inexpensive motherboards are beginning to support such

features, other systems at the DOE computer centers do not. Thus the challenge here is to generate a consensus with respect to an interface definition. Our initial development goal will be to implement this interface on the various machines present at DOE sites. We can then combine these tools into a monitoring sub-system that will periodically perform data gathering. Once sufficient data has been obtained, local sites may build a baseline data set of expected sensor ranges for a given node configuration. This will enable one to build a software system to interpret and preemptively diagnose potential hardware failures.

A critical research area the Center will have to address is the monitoring of node components outside the motherboard. For example, power supplies are one of the most common items to fail in a node, but there is no way to presently detect problems with this hardware. This is an area of instrumentation that no vendor has begun to address. This is one area where we will have to initiate lobbying efforts directed toward convincing vendors of the value in instrumentation of such components.

At a higher level the monitoring system must support multiple views of the cluster. There is the individual hardware monitoring, as already described. There is also the view of the entire set of cluster hardware in aggregate. This large-scale perspective is particularly important for large clusters with thousands of nodes. And at a highest level, the monitoring system should include information about each of the components of the cluster, such that an administrator can confirm that the entire cluster, from the network to the storage system, from the motherboards to the scheduler, is working correctly.

Some software that does baseline monitoring and presents these views already exists in open source form, for example the performance monitoring work at NCSA is based on SGI's Performance Co-Pilot [11]. Performance Co-Pilot is slated to go in the next release of OSCAR. The challenge that remains is to scale existing tools to large systems and to develop and test abstractions of the cluster monitoring data that effectively present the information.

Hierarchical Design

The meta-components shown in figure 1 are a necessary part of any scalable systems software solution. First, because DOE computer centers often have several large systems in-house and therefore it is much more efficient for the system administrators if they can monitor, manage, and schedule their resources from a higher level rather than on an individual machine basis. Second, the scalability of the components being addressed by the Scalable Systems Software Center will require a hierarchical design. The meta-components represent the nodes in an arbitrarily large hierarchical tree while components like a job manager are leaf nodes in this tree. The tree may be spanning a single 100 Tflop system or several systems in a single computer center or even resources that are geographically distributed such as a Grid. This Center is focused on the first two cases but the interfaces to the meta-components will be defined so that other groups may use these interfaces to build components that link our system components to a Grid.

Meta-Scheduler

Only recently have numerous underlying technologies advanced to the point where meta-scheduling across loosely coupled and geographically distributed compute systems has become feasible.

Tremendous activity in the design and development of data and compute 'grids' have propelled meta-scheduling to the position of being one of the final major 'missing links', preventing the widespread adoption and use of computational resources across organizational boundaries. A number of projects do exist, such as Condor and Classified Ads, LSF multi-cluster, and Legion [12]. However, adoption of these particular projects has been greatly retarded by their excessive cost, limits of applicability, and/or limits of functionality.

Silver [13] is an open-source prototype meta-scheduler developed by David Jackson. It has evolved with the support of PNNL, CHPC, BYU and other labs and universities. It is used by a number of sites including PNNL and has been demonstrated at several recent HPC conferences. It features advanced reservations, ultra-high scalability and non-intrusiveness. This non-intrusive design allows the meta-scheduler to apply high-level scheduling optimizations based on its multi-site state information while allowing the underlying local schedulers complete freedom to optimize the local workloads utilizing more extensive local information. We propose to conduct research and develop Silver as the meta-scheduling agent in the resource management software suite.

Meta-Monitor

As discussed previously system monitoring is in a primitive state, but researchers on the Center team have experience developing a meta-monitor that can simultaneously display information from several clusters. Al Geist and Jen Schwidder developed a tool called M3C [14] that allows the monitoring and managing of multiple clusters. NCSA has a performance monitoring tool called GL-monitor that has provided a lot of experience in displaying monitoring information from large clusters. The Center will leverage this large experience base in development of the meta-monitor interfaces and components.

Validation and Testing

Testing is likely the most important part of deploying any large-scale system software. This is especially true where the software components are continually being developed and enhanced by a large, possibly diverse, community. In order for the components to provide the scalability, robustness, reliability, and availability of successful vendor-supplied systems of the past, a rigorous testing process needs to be implemented and executed within the Scalable Systems Software Center.

While most vendors are familiar with the stringent testing that needs to occur before a product can be released, most researchers are not. Many software designers and implementers in the research and development world are accustomed to rapid prototyping or proof-of-concept software where thorough and exhaustive testing is left to those who wish to carry the software beyond the research phase.

The lack of testing is also motivated by the small number of (people) resources that are available to perform testing and diagnosis. Software engineering textbooks will likely quote that a successful software development effort will have a tester-to-developer ratio of at least three-to-one. Research and development sites usually do not have the necessary number of people to devote to the testing process. As such, the test group of a home-grown cluster system software is usually composed of the end users. Unfortunately, this is a role that few of them are aware that they are playing.

Despite these shortcomings, members of this Center have a great deal of experience with the development, maintenance, and testing of open source research software, in particular the Maui scheduler, CPlant software, MPICH [15], and PVM [16]. Their experience and skill will be leveraged inside the Scalable Systems Software Center to develop strategies to insure the correctness of software components developed and distributed through the Center.

4.0 Technical Approach:

There are several concepts that are important to the long-term success of any effort to improve system reliability and management. First is modularity, since a modular system has many benefits. For example, it is much easier to adapt, update, and maintain, in order to keep up with improvements in hardware and other software. Publicly documented interfaces are a requirement because it is unlikely that any package can provide the flexibility to meet the needs of every site. So, a well-defined API will allow a site to replace or augment individual components as needed. Defining the API's between components across the entire system software architecture, provides an integrating force between the system components as a whole and improves the long-term usability and manageability of terascale systems at computer centers across the country.

In order for the components developed at the Scalable Systems Software Center to be acceptable to existing computer centers, they must be compatible with the security infrastructures and policies of the computer centers. It would be ineffective for the components to impose a particular security framework that the computer centers would then have to install on top of their existing security. Each of the components in the system software architecture will be evaluated with respect to its security needs and its security interfaces specified such that it will work within existing security infrastructures.

This proposal is a five-year roadmap for system software research and development broken into three phases.

4.1 Phase 1. Standardize Interfaces and Initial Components

The first 18 months of this project has two objectives: First, is to agree on and create an initial integrated suite of conforming components (subset of architecture) that will be released and updated often throughout this phase. The second objective is to collectively agree on and specify standardized interfaces between system components through a series of regular meetings of the principals, industry, and managers of terascale computer centers.

The interfaces will be platform independent wherever possible. Before defining the interfaces the team needs to converge on an architecture for system software, including tools for hierarchical booting, system monitoring, and resource management. As part of specifying interfaces, the homegrown tools at various labs as well as packages like PBS and such will be modified to conform to the architecture, and thus validate the completeness of the interfaces. These homegrown tools are detailed below. The extensions for interoperability are a substantial effort by themselves, but the result will be a fundamental advance in system software.

First, using our background with existing queue systems and considering the additional requirements of scalability, reliability and security on large systems we will fashion a set of interfaces between each

of the components of the queue system. Existing resource management components for which we will develop and test these interfaces include PBS (resource manager), MPD (job manager), Maui (Scheduler), and QBank (allocation manager). These interfaces will include not only the interfaces between the components of the system, but also between the system and the user, (both job-file, command-line and GUI consumable), between the system and the user's application, and between the system and external services (such as the meta-scheduler to information service interface). The user interface will be modeled on existing queue systems such as PBS and will include a job language translation library to ease the transition for users from other systems to our interface. The interface between the user's application and the system will have several features. An important feature is that the system must facilitate a truly parallel job startup. This will require that the system help the user's application perform the initial communication setup among the user level processes.

The second goal will be to determine the optimum way for the components to communicate with each other, particularly with the node daemons. Existing systems utilize a star pattern for communications and all communication pipes are transient. Clearly this is not a scalable approach, but it will take some research to determine the best communication patterns to satisfy our requirements of scalability and reliability. A key to the solution will be to maintain a set of communication pipes among the node daemons in a pattern such as a simple loop or tree based arrangement. This will allow commands to be passed among the nodes quickly, without the need to establish new communication sockets. The pattern must not only produce good scalability, but it must also be a self-healing pattern such that down nodes are quickly identified, removed from the communication chain and communications rerouted around the down node in the pattern.

We will perform an analysis of the technical issues associated with implementing checkpoint/restart within the confines of the software components developed by this ETC on Linux systems. For example, checkpointing and later restarting a process with an open socket connection is, in general, impossible. However, if the socket is connected to the node job manager, a protocol could be defined in which both processes cleanly terminate the connection prior to the checkpoint. A similar argument would be made for the MPI connections between the processes of a parallel application. Intelligence could be added to the MPICH device layer to quiesce the communication channels prior to the checkpoint operation. The deliverable for this phase of the project would be a paper describing, in detail, the technical issues faced, the limitations we would have to impose on process groups and parallel application eligible for checkpoint/restart and the work-arounds possible within the confines of the software environment proposed by this ETC. Included would be the APIs required between the various software components for both user-initiated and system-initiated checkpoint and restart operations.

During phase 1, we will further define the role of the node configuration component, as it will likely consist of many distinct subcomponents working together and then define the interfaces through which the configuration component will communicate with other portions of the system software. Additionally, we will define the range of configuration choices that we will support, expecting that this range will need to be large in order to match the needs of DOE computer centers.

The initial set of solutions will be built from existing tools. A few of these are commonly available as open source (such as CFengine), but the majority of these have been developed at the various

participating labs as a part of installing and supporting real clusters. These tools are described here as they currently exist; during phase 1, these will be modified to conform to the defined interfaces.

The City Toolkit from Argonne National Laboratory is a suite of software that was developed to support Chiba City. The City toolkit supports centralized, scalable and highly flexible management. It includes a database used for cluster configuration, database replication tools, a configuration sanity checker and policy enforcer, a boot mechanism, a set of operating systems used to automatically rebuild a node when necessary, remote power and console management tools, and other software.

The Sandia cluster configuration and management tools that are currently used to build and maintain the production Cplant machines are based on a hierarchical management strategy that allows may of the basic functions, such as booting and installing system software, to occur in parallel. At the heart of the management tools is an object-oriented database that defines roles and groupings for every piece of hardware in the cluster and that allows the machine to have an arbitrary management topology. The Cplant tools are designed specifically for a machine that has a large collection of diskless compute nodes that do not share a flat Ethernet network. The tools are also designed to support some uncommon features, such as the ability to quickly and easily move a collection of compute nodes and their associated management nodes between classified and unclassified networks.

The Open Source Cluster Application Resources (OSCAR) package developed by ORNL, NCSA, IBM, Intel, MSC Software, Veridian, DELL, and SGI is designed to provide straightforward installation services for standard Beowulf clusters. OSCAR uses a GUI based environment to gather the necessary information from a user prior to installation. Information is entered only once and is then subsequently used by all components during the OSCAR automated installation. The lack of scalability is presently the most limiting factor when using the OSCAR tools to install a large cluster.

Management tools such as ORNL's command line based Cluster Command and Control (C3) suite [17] as well as the web based cluster management application Managing and Monitoring Multiple Clusters (M3C) will also be included in the phase 1 standardization process.

Like much of the rest of the Phase 1 activities, the critical goal for monitoring component is to correctly identify the interfaces with the rest of the system software. Existing monitoring tools from the various groups will be improved and extended to integrate information from most of the other components of the system. Through this process right and wrong ways of defining the interfaces will be revealed. The long-term goal for the monitoring system is to settle on one interface (graphical and textual) that is necessary for the basic operation of the cluster. Work will also take place on the system-monitoring infrastructure; ideally there will be one daemon for monitoring purposes per node.

In addition to working on conventional cluster nodes, i.e. nodes that use a hard disk to store the operating system and associated files, we will also ensure that this Center's software works with a new type of cluster node that boots Linux from on-board FLASH (i.e. BIOS) non-volatile memory. These nodes can boot Linux in 1-2 seconds and become active as functioning cluster nodes in less than 10 seconds -- even the very first time they are installed in a cluster. A number of DOE Labs are now working with LinuxBIOS, including LANL, LLNL, and Sandia. Now that feasibility has been demonstrated, the next step is to build on this technology.

These nodes are much more easily controlled and managed than existing cluster nodes. For this reason we call them *manageable nodes*. The issues concern how to make these nodes work with the software to make clusters easier to manage. In Phase 1, we will modify the LinuxBIOS to support the software developed as part of this effort, and at the same time ensure that the software can be used in concert with LinuxBIOS-based nodes.

The Unix environment provides a rich set of cryptic yet familiar commands for operations, such as `cp`, `mkdir`, `ls`, `ps`, `grep`, `find`, etc. In a distributed parallel environment, particularly a large one in which there is no globally shared file system; system administrators as well as application developers require a simple way to invoke these operations in parallel across an entire cluster [18]. In phase 1 a scalable, self-correcting fault tolerant implementation of such user utilities will be investigated.

Validation and verification for systems software is an undeveloped area because computer centers had little to compare against. The Scalable Systems Software Center will have to make testing, validation and verification a key part of its research effort as well as a key part of its software maintenance. First, because the creation of a standard distribution will likely require selection among tools with the same functionality. In order to assess the comparative strengths and weaknesses of these implementations, functionality and performance tests will be needed. Ideally, these tests would be able to assist in quantitatively and qualitatively measuring the appropriateness of one component implementation over another. For many of the components in Figure 1 there isn't even evaluation criteria much less validation tests. In Phase 1 we will do research on the types of tests and evaluation criteria that will be needed over the lifetime of the Center. These evaluation criteria will themselves be evaluated as they are applied to the process of selecting initial components to go in the distribution.

The Scalable Systems Software Center will have three types of deliverables: standard interface specifications, research papers and presentations on scalable system components, and software for distribution for computer centers across the nation. The Center milestones are aligned with the three phases of the research plan. Initial output of software from the Center will begin just a few months after startup and will continue throughout the five-year lifetime of the Center.

Phase 1 Milestones and Deliverables

- (6 months) Distribute a series of interface, component, capability descriptions provided for node build, configure, and manager software.
- (6 months) Detailed technical report on process checkpoint/restart for Linux systems.
- (9 months) Compliant node build implementations from various sites with support for both diskless and local disk support cluster architectures. Tested up to 1024 processors
- (12 months) Establish and release initial resource management interface specifications
- (12 months) Establishment of the CVS repository and module structure, agreement on documentation conventions.
- (12 months) Finalized API for system initiated checkpoint restart of parallel MPI jobs on Linux systems.
- (18 months) Release v1.0 of the Center's resource management system based on existing open source code and the results of the scalability testing.

4.2 Phase 2. New Capabilities and Full Systems Suite

The next 24 months will focus on research and creation of components that have no existing versions, for example checkpoint/restart, as well as continuing improvements in efficiency and scalability of the most critical system components. The determination of which components are most critical will be based on feedback from system administrators, managers, and power users of the SciDAC computer centers. New scalable versions of components will be developed, validated, and swapped into the Center's system software distribution. Avenues for long-term support for the distribution will be investigated through the industry participants in the first phase.

We will fully implement the set of interfaces determined during phase 1, tested as a system and released as a full suite of resource management software. Work will begin on more scalable and fault tolerant versions of the resource manager components and to integrate these components into the security model.

The job manager component will add support for jobs written for multiple libraries, such as MPI-2, Global Arrays [19], and other one-sided communication libraries. It will implement additional services as determined by customer feedback, as well as offer standard interfaces to tools that need the cooperation of the job manager, such as parallel debuggers and steering tools.

A new queue manager component will be implemented that fully tracks the state of both jobs and nodes eliminating the need for other components, such as the scheduler, to directly interact with the nodes. Full versions of the node daemons will provide true parallel command startup, tracking and termination.

The resource manager will implement the scalable communication infrastructure as designed in phase 1. It will provide resource limit enforcement to ensure that users stay within their requested resource allotment (something increasingly important for multiple jobs sharing SMP nodes). Also during phase 2 we will extend the interface between the system and the user's application to allow bi-directional communication. This will be useful in directing application level checkpoint/restart as well as provide new features such as the ability of the job to request an extension of its job reservation, or perhaps to allow jobs to expand and/or contract their resource usage (dynamic job support). Other research will examine the possibility of allowing users to specify fuzzy resource requests (resource preferences).

The scheduler will likewise be enhanced to provide support for these capabilities: resource limit enforcement and tracking, suspend/resume preemption and parallel checkpoint/restart, interactive support/job steering, and dynamic job support. In addition it will provide enhanced Quality of Service specification and priority management.

The meta-scheduler will begin research and development in the area of intelligent data pre-staging, a critical element for the efficient use of distributed resources to be successful. Also, peer to peer interaction between meta-schedulers will enhance the effectiveness of meta-scheduling by extending the view and reach of each meta-scheduler allowing greater resource access, a larger selection of jobs to choose from, and improved load balancing across systems.

The allocation manager will be enhanced to track, allocate and charge for SMP resources such as memory, network and disk. As part of the scalability improvements necessary to support thousands of

processors, we propose to implement in-memory data-caching for time-critical read-only queries like quotes and balance checks. Also a new method to break up large tables needs to be devised to ensure good performance for accounting queries.

During phase 2, the emphasis in the configuration component of the system will be to:

- Merge, where it makes architectural sense, existing tools into only a few sets of tools. One likely approach to this will be to develop a generic framework for tools that supports multiple "plug-ins" appropriate to the type of cluster being managed.
- Enhance reliability by distributing knowledge and supporting fault tolerance.
- Support new capabilities necessary for the rest of the environment.

During this phase of development we will investigate both broadcast and multicast capability of network switches to improve performance when invoking parallel cluster operations. We will also take advantage of hardware support of operations where it is available.

The monitoring subsystem will be expanded, via the interfaces defined in phase one, to present a view of the overall system, including queue information, network performance, storage capacity, state of external links, and detailed node hardware information. It will also be modified in order to support large scalability. This will most likely require an updated architecture for hierarchical monitoring, possibly using multicast as a way of communicating monitoring information.

We will implement checkpoint/restart on a Linux cluster based on the analysis performed in phase 1. This work will build upon the existing infrastructure available at the time. For example, Scyld's bproc can dump and restore the memory image of a process for migration to a remote node or for remote execution. It currently cannot handle open files. During the time frame of this work, global file systems are likely to be available insuring a consistent state and name-space for all files on all nodes of the cluster. We would take advantage of this to allow the restarting of processes on nodes different from where they were checkpointed. The result of this phase of the project would be a working, but possibly limited, implementation of parallel checkpoint/restart on Linux clusters running the software proposed in this ETC.

In phase 2 the evaluation criteria and validation tests will be expanded and utilized to determine when improved components should be swapped into the standard distribution. As part of the Center's software maintenance, the team will document evaluation criteria for new components that are added to the standard distribution. These criteria will be used to develop validation tests for future optimized versions of the components. Testing of a multi-component software system of the type we envision involves both tests of single components and integrated tests of multi-component subsystems. Each single component of the system is complex enough to deserve a fairly sophisticated testing strategy. Many participants in the project have experience in testing software for reliability as it is being developed. We plan to develop a testing harness that can be used by various developers. One example of such a system is that used by the MPICH [20] implementation of MPI.

Once each component and its interfaces have been tested, the more difficult task of integration tests must be performed. The primary approach will be to use existing systems to run real job mixes on. The Chiba City machine at Argonne, for example, is specifically configured so that an entirely new collection of system software, including a new version of the OS itself, can be quickly loaded onto a

collection of nodes for testing. Once the software is demonstrated to be reliable and robust it will be tested in real production environments such as the DOE topical centers. This will allow testing on the largest systems available at any given time. If these tests demonstrate superior components, then the software could be left on these production systems and form a natural process of transitioning the products of the Scalable Systems Software Center into real use.

Phase 2 Milestones and Deliverables

- (20 months) Release v2 of the interface specification adding interfaces for completely new components, such as the Meta-scheduler, and including feedback on the v1 specification from users and the implementation.
- (24 months) Tru64 & AIX initial support
- (24 months) The job manager can start 100 processes in 1 second, and 1000 processes in 5 seconds
- (24 months) Deliver "Plug-in" framework for systems management tools.
- (28 months) Resource manager and scheduler add support for resource limit enforcement, dynamic jobs, and checkpoint/restart
- (30 months) Release a full implementation of the v1.0 interface specification on all components.
- (30 months) Set up user-oriented maintenance system for responding to problem reports
- (36 months) A new meta-scheduler component employing data-staging
- (36 months) Node management software tested on 4000 processors
- (42 months) Able to support loss of up to 25% of the cluster, including the management nodes.
- (42 months) Initial implementation of parallel checkpoint/restart of an MPI application on a Linux cluster.

4.3 Phase 3. Advanced Scalability and Software Support

The last 18 months is marked by getting vendors to support the Scalable System Software distribution on their platforms and by advanced research into the scalability needs of petascale systems software. As vendors take over the support of the stable software release, the Center can begin to focus on the next generation of computers and the creation of system software components needed for this future technology. The Scalable Systems Software Center will be a catalyst for fundamentally changing the way future high-end systems software is developed and distributed.

The full resource management implementation will be updated as needed to run on the current state of the art systems. Some components may need to evolve in order to manage jobs on many thousands of nodes. This will require innovative approaches to fault tolerance, scalability and security. However, the API should remain unchanged.

The resource manager communication infrastructure will be enhanced to intelligently compress state and resource configuration data. Additionally, node and communication failures will automatically be handled and routed around in a non-serial manner preventing failures from bottlenecking communications performance.

This is a period in which enhanced scheduling algorithms for the scheduler and meta-scheduler will be investigated and implemented. Every scheduling decision will consider the cost on underlying network

bandwidth, available data staging space, computational throughput, and start time. The scheduler and meta-scheduler will be enhanced to support co-allocation of scarce resources (network, data, licenses) and locality scheduling. The resource manager and scheduler will be enhanced to allow locally submitted jobs to migrate to systems where they could run sooner. The meta-scheduler and allocation manager will be augmented to allow for monitoring, tracking, allocation and accounting across site and system boundaries.

In order to support petascale systems, the hierarchical model of cluster architecture that is common today will blossom into multiple levels of hierarchy. To keep pace the Center will do research on node configuration systems that are able to operate at multiple hierarchical levels. Any kind of centralized resource (such as a database of configuration information) must be fully replicated and reliably distributed to multiple servers, each of which are fault tolerant and able to fail over to other similar systems.

We will continue to extend and improve the implementation provided in Phase II as well as to experiment with how best to use this capability in production. Tools and scripts to automate the management of various job queue configurations will be provided. The LBL-NERSC center currently uses an integrated collection of PERL scripts to automatically control its batch configuration on its T3E. Multiple configurations can be defined and scheduled throughout the day or week. During a configuration change, the existing workload is checkpointed to disk and the jobs associated with the newly selected configuration are restarted. These tools would be ported to the software environment proposed under this ETC.

The monitoring system will become the primary interface for operational management of the system. It will also have hooks for applications to use, as part of support for the checkpoint/restart system. And, if at all possible, it will be used to support some amount of predictive fault detection, based on analysis of common fault patterns. This ability will be necessary on future systems with tens of thousands of nodes.

The development of scalable systems software, if it is to achieve the highest levels of performance, must proceed in tandem with the development of low-level communications mechanisms. We plan to cooperate with the developers of other cluster software packages to provide higher-level abstract communications and distributed systems facilities where necessary. Examples of such work include the development of a kernel level block device driver or a mechanism for remote procedure calls. These areas require future standardization.

The ideal scalable systems design environment would include a suite of simulator tools that are capable of performing scalability tests for HPC systems that are envisioned a few years into the future. As the system goes into production, these same simulators can give better intuition to the end user about how their application is behaving in this environment, and they can allow the system designers to refine their design for better performance. These tools can run the gamut from full system simulators closely modeling the actual hardware and software in a system to more focused, discrete simulators that address a single design problem. One aspect in our phase 3 research of systems software for petascale computers, we will explore the development of a suite of simulator tools that could be used to test system components at a scale beyond available hardware.

Phase 3 Milestones and Deliverables

- (48 months) Release v3 of the system fully implementing the v2 interface specification.
- (48 months) The resource manager implements a fault tolerant communication subsystem
- (48 months) Maintain web site and mailing list for responding to problems
- (54 months) The job manager runs for 30 days continuously on the largest available system
- (54 months) Incorporate feedback on previous implementations and validate the system on the latest systems available at this point in time.
- (54 months) Production release of parallel checkpoint/restart capability for MPI applications on a Linux cluster.
- (60 months) Administrative tools to automate workload configuration changes using checkpoint/restart.
- (60 months) Able to support loss of up to 50% of the cluster without impacting the remaining nodes or jobs.
- (60 months) The scheduler and meta-scheduler support co-allocation and job migration.
- (60 months) Scalability tests on the largest system in the DOE SC system

5.0 Management Plan: Being a virtual center, the management of the work becomes an important aspect to guarantee success. The need for consensus on interfaces requires numerous group discussions and meetings. Team members will meet by teleconference biweekly and in person quarterly for progress reports and decision-making. Decisions will be made by a consensus of the project PIs with the Center coordinator acting as arbitrator and final decision maker when a consensus isn't reached.

6.0 Primary areas of interest of the team members:

The Scalable Systems Software Center is organized as a single team working together on a collective goal. Primary areas of interest of the team members is listed in the following table.

ORG	Name	Primary tasks
ANL		
	Rusty Lusk	Job manager, testing
	Remy Evard	Node Configuration
	JP Navarro	Configuration, Job manager
	Daniel Nurmi	Config, Monitoring, testing
	Narayan Desai	Config, Monitoring, Job mgr
	David Ashton	App env
PNNL		
	Scott Jackson	Resource mgmt, security
	David Jackson	Resource mgmt, App env
	Kevin Walker	Resource mgmt, testing
Ames		
	Brett Bode	Resource mgmt, App env
	Brian Smith	Resource mgmt, testing
	Daniel Greig	App env, testing
	Cedric Collins	Resource mgmt

	TBD	
ORNL		
	Al Geist	Coordinator, Monitoring
	Stephen Scott	Config, App env, Job mgr
	Jens Schwidder	Monitoring, testing
	Brian Leuthke	Config, security, testing
	Michael Brim	App env, Job manager
LANL		
	Ron Minnich	Monitoring, Node manager
	TBD	
	TBD	
LBL		
	Eric Roman	App env, testing
	Paul Hargrove	App env- checkpt/restart
	Michael Welcome	App env-checkpt/restart
SNL		
	Neil Pundit	Validation, simulation
	Ron Brightwell	Config, monitoring
	Rolf Reisen	App env, Job manager
	Lee Ward	App env, monitoring
	Doug Doerfler	Validation, testing
	Lee Ann Fisk	Config, resource mgmt
NCSA		
	Rob Pennington	Validation, simulation
	Avneesh Pant	Resource mgmt, testing
	Mike Showerman	monitoring

7.0 Facilities and Institutional Resources:
While there are several small systems (less than 200 processors) scattered across the team for software development and testing. For larger scalability tests, time on Chiba City at Argonne will be available as well as time on the NCSA cluster. For even larger scale tests, time on the flagship and topical center computers will be obtained, but only after the software has proven reliable and robust in smaller scale situations.

Chiba City is a 256 node PC cluster designed to support scalable computer science, and as such, has a constantly changing configuration. It ranges from a standard set up in which all of the computation nodes are running a standard Linux distribution, to one in which a user may have root on a set nodes while another is running some other operating system on a different set of nodes.

NCSA is presently installing a 512 node Pentium III Linux cluster. This large cluster will be available for performing scalability studies on the system components under development. In addition, by the end of 2001 NCSA will have a 320 processor Itanium cluster, which will provide another architecture on which to port the scalable systems software.

DOE needs the components developed by the Scalable Systems Software Center to be able to run on the tera-scale machines that exist in the DOE computer centers. The largest machines are IBM SPs and Compaq Alpha clusters. Nodes on the large Compaq Alpha cluster at ORNL have also been offered to the Scalable Systems Software Center for software development and testing on this platform. The 0.5 teraflop Compaq cluster runs Tru64 Unix and standard production software from Compaq. In addition the ORNL computer center has offered access to nodes on their IBM SP. The 1 TF IBM runs AIX and includes the standard production software from IBM.

Between the smaller development clusters, larger PC clusters, and the Compaq and IBM systems, the Scalable System Software Center has access to all the required resources for the research, development, and deployment of the software produced by the Center.

8.0 References

1. PITAC President's Information Technology Advisory Committee Report to the President, "Information Technology Research: Investing in Our Future", http://www.ccic.gov/ac/report, February 1999.

2. Veridian Systems, Portable Batch System (PBS), http://www.OpenPBS.org

3. Platform Computing, LSF (Load Sharing Facility), http://www.platform.com/platform/platform.nsf/webpage/LSF

4. J. Pruyne, M. Livny, "Interfacing Condor and PVM to Harness the Cycles of Workstation Clusters," *Journal on Future Generations of Computer Systems,* Vol. 12, 1996

5. Ralph Butler, William Gropp, and Ewing Lusk, "Components and interfaces of a process management system for parallel programs", *Workshop on Clusters and Computational Grids for Scientific Computing*, Lyon, France, September 2000.

6. B. Bode, D. M. Halstead, R. Kendall, Zhou Lei, and D. Jackson, The Portable batch scheduler and the Maui Scheduler on Linux Clusters. *Proceeding of the USENIX Extreme Linux Technical Conference*, Atlanta, Georgia, October 2000, pp 217-224.

7. S. Jackson, Qbank 2.8 A CPU Allocations Bank, http://www.emsl.pnl.gov:2080/docs/mscf/qbank-2.8

8. M. Brim and S. Scott, "OSCAR: Open Source Cluster Application Resources," 2001 Linux Symposium, July 25-28, 2001, Ottawa, Canada. http://www.csm.ornl.gov/oscar

9. R. Evard, et al, City—The MCS Large Cluster System Software Toolkit, http://www-unix.mcs.anl.gov/systems/software/city

10. S. Mehat, et al, VA Cluster Management (VACM) 2.0, http://valinux.com/software/vacm

11. SGI, Performance Co-pilot, http://oss.sgi.com/projects/pcp

12. A. Grimshaw, W. Wulf, "Legion The next logical step toward the world-wide virtual computer," http://www.cs.virginia.edu/~legion

13. D. Jackson, Silver Metascheduler Project, http://supercluster.org/projects/silver

14. A. Geist, J. Schwidder, B. Luethke, S. Scott, "M3C: a web-based management tool for federated clusters," Proceedings of IEEE International Conference on Cluster Computing, December 2000

15. William Gropp and Ewing Lusk, "Sowing MPICH: A case study in the dissemination of a portable environment for parallel scientific computing," International Journal of Supercomputer Applications and High Performance Computing, 11(2) 103-114, Summer, 1997.

16. Geist, et al, "PVM: Parallel Virtual Machine" MIT Press, 1994.

17. S. Scott, B. Luethke, Al Geist, Ray Flanery. "Cluster Command & Control (C3) Tools Suite." Proceedings of Third Distributed and Parallel Systems Conference (DPYS2000) June 2000

18. William Gropp and Ewing Lusk, "Scalable Unix tools on parallel processors,", in *Proceedings of the Scalable High-Performance Computing Conference*, IEEE Society Press, 1994, pp. 56-62.

19. J. Nieplocha, R. Harrison, R. Littlefield, "Global Arrays: A nonuniform memory access programming model for high performance computers," *Journal of Supercomputing,* 10:197-220, 1995

20. William Gropp, "Coding Standards and Development Framework", MCS Technical Report, 2000.

9.0 Biographies of PIs:

Al Geist, Section Leader Oak Ridge National Laboratory
http://www.csm.ornl.gov/~geist

Al Geist is a senior research staff member at Oak Ridge National Laboratory and leader of the Distributed Computing Section in the Computer Science and Mathematics Division. Al is one of the original developers of PVM (Parallel Virtual Machine) which became a world-wide de facto standard. Al continues to be the technical manager of the Heterogeneous Distributed Computing project at ORNL. He was actively involved in both the MPI-1 and the MPI-2 design teams. He is a member of the IEEE Taskforce on Cluster Computing and is presently involved in the DOE 2000 Common Component Architecture (CCA) forum. The CCA is an attempt to define a standard for interoperability between high-performance (possibly parallel) software components ranging from math routines to computational steering modules. Al is a co-PI on the DOE 2000 Electronic Notebook project and his notebook software is in use by hundreds of research groups in industry, labs, universities, and medical research facilities around the world. In his 15 years at Oak Ridge National Laboratory, Al has published two books and over 150 papers in areas ranging from heterogeneous distributed computing, numerical linear algebra, parallel computing, collaboration technologies, solar energy, and solid state physics. Al is a member of Phi Kappa Phi, Tau Beta Pi, SIAM, and served on numerous conference program committees and has been chairman of four conferences including the Joint PC Cluster Computing Conference (JPC4-5) and the Fifth Distributed Supercomputing Conference.

Al has won numerous awards in high-performance and distributed computing including: the Gordon Bell Prize (1990), the IBM Excellence in Supercomputing Award (1990), an R&D100 Award (1994), two DOE Energy 100 awards (2001), the American Museum of Science and Energy Award (1997), and the Heterogeneous computing challenge several times (1992, 1993, 1995, and 1996). He joined Oak Ridge National Laboratory in 1983 after receiving a BS in Mechanical Engineering from North Carolina State University.

Selected Publications

R. Armstrong, D. Gannon, Al Geist, K.Keahey, S. Kohn, L. McInnes, S. Parker.
Toward a Common Component Architecture for High Performance Scientific Computing
Proceedings of HPDC'99, August 1999

J. Kohl, Al Geist, Monitoring and Steering of Large-Scale Distributed Simulations.
Proceedings of Applied Modeling and Simulation Conference (AMS'99), September 1999

Al Geist, J. Kohl, S. Scott, P. Papalopoulos. HARNESS: Adaptable Virtual Machine Environment for Heterogeneous Clusters. Parallel Processing Letters Vol. 9 No. 2 253-273, 1999

S. Scott, B. Luethke, Al Geist, Ray Flanery. Cluster Command & Control (C3) Tools Suite.
Proceedings of Third Distributed and Parallel Systems Conference (DPYS2000) June 2000

Al Geist, PVM and MPI: What else is needed for Cluster Computing? Proceedings of EuroPVM-MPI 2000. Springer LNCS, September 2000

Al Geist, J. Schwidder, B. Luethke, S. Scott, M3C: a web-based management tool for federated clusters, Proceedings of IEEE International Conference on Cluster Computing, December 2000

Neil Pundit, Department Manager, Scalable Computing Systems, Sandia National Labs
http://www.cs.sandia.gov/~ndpundi

Biographical Sketch
Neil Pundit is a Department Manager at Sandia National Laboratories responsible for Scalable Computing Systems. His department is focused on system software development for the Cplant™ and upcoming supercomputing systems. This department is credited with the operating system development for the ASCI Red. He serves on the Executive Board of Sandia's Computer Science Research Institute. He is an Adjunct Professor of Computer Science at the University of New Mexico.

Neil joined Sandia Labs two years ago in March'1999. Prior to that his major professional associations include: 16 years with Digital Equipment Corp (now Compaq) as a Group Engineering Manager, and 6 years with Jet Propulsion Laboratory of Caltech where he headed the Orbit Insertion Operations for the Viking Missions to Mars.

His personal technical contribution has been in the guidance and control of space vehicles. He has provided technical guidance to a number of leading edge software R&D projects throughout his career.

Neil received his bachelor's, master's, and Ph.D. degrees, all in Electrical Engineering, respectively from Bihar Institute of Technology in India (1961), Texas A&M University (1966), and Auburn University (1969).

Neil is a recipient of NASA/TRW Lunar Landing Award for the first moon landing in 1969, and NASA/JPL/Caltech Mars Landing Award for the first Mars landing in 1976, and also US Air Force Association's Theodore von Karman award (highest honor in science and engineering) as a member of the Viking Flight Team. His technical direction has resulted in 30 software patents, and 2 international awards for innovative applications.

Among his recent affiliations with the supercomputing activities, Neil was the Technical Chair for the 4th Workshop on Distributed Supercomputing in 2000, and a Panel Chair for Petaflops Computing at SC2000.

Ewing L. Lusk
http://www.mcs.anl.gov/~lusk

Biographical Sketch:
Ewing “Rusty” Lusk received his B.A. from the University of Notre Dame (1965), and his M.S. and Ph.D. from the University of Maryland (1969 and 1970), all in mathematics. He began his career as

an assistant professor of mathematics at Northern Illinois University, but moved into the eventually into Computer Science Department, where he eventually became full professor and Acting Chairman of the Department. In 1982, Lusk joined Argonne National Laboratory (ANL) as a computer scientist, and was named to his current position of senior computer scientist in 1989. Along with Paul Messina and Jack Dongarra, he was instrumental in founding Argonne's Advanced Computing Research Facility, which signaled the beginning of Argonne's commitment to parallel computing research.

Lusk's research interests are in the areas of parallel computing, program visualization, automated theorem proving, logic programming, and database technology. His current projects include research into programming models for parallel architectures, parallel performance analysis tools, and an implementation of the MPI Message-Passing Standard. He was an active member of the original MPI Forum and chairman of the MPI-2 Forum. Together with William Gropp he is an implementor of MPICH, a widely-used, high-performance, portable implementation of MPI. He is the developer of the Jumpshot program visualization system distributed with MPICH.

Lusk is a leading member of teams that have produced the Argonne theorem-proving systems, the Aurora parallel Prolog system, the p4 parallel programming environment, and the MPICH implementation of the MPI Standard. He is the co-author of three books in automated reasoning and parallel computing, and has authored more than 75 research articles in mathematics, automated deduction, and parallel computing.

Selected Publications

William Gropp, Ewing Lusk, and Anthony Skjellum, *Using MPI: Portable Programming with the Message-Passing Interface,* MIT Press, Cambridge, MA (1994).

"A Scalable Process Management Environment for Parallel Programs," (with Ralph Butler and William Gropp), *in Recent Advances in Parallel Virtual Machine and Message Passing Interface*, Jack Dongarra, Peter Kacsuk, and Norbert Podhorszke, eds., *Springer Lecture Notes in Computer Science #1908*, September, 2000, pp. 168-175.

William Gropp, Steven Huss-Lederman, Andrew Lumsdaine, Ewing Lusk, Bill Nitzberg, William Saphir, and Marc Snir, *MPI – The Complete Reference: Volume 2: The MPI-2 Extensions*, MIT Press, Cambridge, MA (1998).

William Gropp, Ewing Lusk, Nathan Doss, and Anthony Skjellum, "A high-performance, portable implementation of the MPI message passing interface standard," Parallel Computing, 22(6) 789-828, September, 1996.

William Gropp and Ewing Lusk, "A high-performance MPI implementation on a shared-memory vector supercomputer," Parallel Computing 22(11) 1513-1526, January, 1997.

William Gropp and Ewing Lusk, "Sowing MPICH: A case study in the dissemination of a portable environment for parallel scientific computing," International Journal of Supercomputer Applications and High Performance Computing, 11(2) 103-114, Summer, 1997.

Ralph Butler, William Gropp, and Ewing Lusk, “Components and interfaces of a process management system for parallel programs”, *Workshop on Clusters and Computational Grids for Scientific Computing*, Lyon, France, September 2000.

Ewing Lusk and William McCune, “ACL2 for parallel systems software, in *ACL2 Workshop 2000 Proceedings*, Matt Kaufmann and J Strother Moore, eds.

William Gropp and Ewing Lusk, “Scalable Unix tools on parallel processors,”, in *Proceedings of the Scalable High-Performance Computing Conference*, IEEE Society Press, 1994, pp. 56-62.

William Gropp, Ewing Lusk, Nathan Doss, and Anthony Skjellum, “A high-performance, portable implementation of the MPI Message-Passing Interface standard, *Parallel Computing*, vol. 22, n. 6, (1996), pp 789-828.

Brett M. Bode
http://www.scl.ameslab.gov/scl/Personnel/brett

Biographical Sketch

Brett Bode received his B.S. degree from Illinois State University (1993) in Chemistry and Physics. In 1998 he received a Ph.D. in Physical Chemistry from Iowa State University. His thesis in Theoretical Chemistry was focused on enhancing computational chemistry via enhanced methods such as a new effective core potential code allowing analytic second derivatives and through a better human interface to the GAMESS code through the creation of a program called MacMolPlt. In December 1998 he joined the staff in the Scalable Computing laboratory at Ames laboratory as an Assistant Scientist and was promoted to his current rank of Associate Scientist in 2000.

Bode’s research interests are broadly focused on delivering more usable compute cycles to scientific applications. In particular his current work examines two significant problems on today's parallel computing systems. The first is improving communication performance, especially communication latency. This research is focusing on OS-bypass techniques such as M-VIA and GM to improve communication latency and then assisting application developers to make use of the new interfaces. The second research area is management software for parallel computer systems including node configuration and batch management software.

Selected Publications

Bode, B. M.; Gordon, M. S. MacMolPlt: A Graphical User Interface for GAMESS. *J. Mol. Graphics Mod.* 1999, *16*, 133-138.

D. M. Halstead, B. Bode, D. Turner, and V. Lewis, Giga-Plant Scalable Cluster. *Proceeding of the USENIX Extreme Linux Technical Conference*, Monterey, California, June 1999, pp 10-15.

Fletcher, G. D.; Schmidt, M. W.; Bode, B. M.; Gordon, M. S. The Distributed Data Interface in GAMESS. *Comput. Phys. Commun.,*,128 (2000) 190–200.

B. Bode, D. M. Halstead, R. Kendall, Zhou Lei, and D. Jackson, The Portable batch scheduler and the Maui Scheduler on Linux Clusters. *Proceeding of the USENIX Extreme Linux Technical Conference*, Atlanta, Georgia, October 2000, pp 217-224.

Robert L. Pennington, Associate Director, Computing and Communication Division NCSA
robp@ncsa.uiuc.edu

Biographical Sketch

Rob Pennington currently heads the Computing and Communications Division at the National Center for Supercomputing Applications. In this role, he is responsible for the major computing systems and supporting infrastructure at the NCSA, which include a large SGI Origin cluster, Linux and Windows clusters, the mass storage system and networking. These resources include production systems that are available to the national community as well as research and development systems, with a strong emphasis on clusters. Prior to this, he was the head of the NT Cluster Group at the NCSA, which has responsibility for the NT Supercluster. The group is focused on supporting high performance computing on a large NT cluster and making it a production system for NCSA users. This work includes software infrastructure development and deployment, evaluating systems, interconnects and storage area networks, and applications porting and testing. In addition to experience with clusters, he has worked on mass storage system performance analysis and usage patterns as part of the High Performance Data Management group at the NCSA and oversaw the transfer across the vBNS of over 2 TB of user's data to NCSA from the Pittsburgh Supercomputing Center for the PACI program. Prior to the NCSA, he led the Advanced Systems Group at the PSC, and was working with software interfaces and performance analysis for mass storage systems. Rob received a Ph.D. in Astronomy (1985) from Rice University.

David B. Jackson
http://supercluster.org/jacksond

Biographical Sketch

David Jackson is a Senior Systems Developer at the Pacific Northwest National Laboratory (PNNL). His research and development activities are focused in the design and implementation of systems middleware for high performance supercomputers and clusters. He designed, developed, and supports the Maui Scheduler, an advance reservation based optimizing back-fill scheduler, which is now in production use at many of the country's largest supercomputer and cluster systems including a high percentage of the largest IBM SP and Linux clusters. Mr. Jackson has also developed scaleable cluster resource management and meta-scheduling software including the Wiki resource manager and Silver metascheduler to support the upcoming needs of the computational grid community. His research interests lie in the areas of optimizing resource utilization under real-world system constraints and developing highly robust, scaleable system management software architectures.

Mr. Jackson received his BS in Electrical and Computer Engineering as well as his BS and MS in Computer Science at Brigham Young University and is currently pursuing a Ph.D. in Computer Science. He has worked for numerous high performance computing centers providing resource management and scheduling services including PNNL, Lawrence Livermore National Laboratory, San

Diego Supercomputer Center, NCSA, MHPCC, and the Center for High Performance Computing. He has also worked as a consultant at IBM's AIX System Center. He is active in the Grid Forum scheduling working group and is the lead of the SP-XXL scheduling and resource management working group. He has written multiple papers in the areas of scheduling and resource management for high performance computing systems and presented at various high performance computing conferences including IPDPS, IEEE Sigmetrics, SPWorld, and Usenix/Extreme Linux.

Selected Publications

Q. Snell, M. Clement, D. Jackson, and C. Gregory, The Performance Impact of Advance Reservation Metascheduling, Job Scheduling Strategies for Parallel Processing, Editors: Dror G. Feitelson and Larry Rudolph, Springer Verlag, LNCS Vol 1911, 2001

D. Jackson, H. Jackson, and Q. Snell, Simulation Based HPC Workload Analysis, Proceedings of the International Parallel and Distributed Processing Symposium, 2001

D. Jackson, and Q. Snell, Integrating Fairness and Resource Optimizations on HPC Systems with the Maui Scheduler, Proceedings of IEEE Sigmetrics, 2001.

Scott M. Jackson

Biographical Sketch

Scott Jackson is a senior systems programmer in the Molecular Science Computing Facility at the Pacific Northwest National Laboratory. He is project manager of the PNNL Linux Cluster Administration task overseeing the establishment of a 194-processor Linux cluster into a production-level HPC resource. He received his BS degree in Electrical/Computer Engineering at Brigham Young University. Prior to his work at PNNL, Scott was a Senior System Analyst at the Maui High Performance Computing Center, which was preceded by a position working for IBM as an AIX technical specialist. He has specialized in the development and integration of software in the areas of cluster administration, accounting, resource management, scheduling, meta-scheduling and security. Mr. Jackson is the architect and developer of QBank, a dynamic resource allocation management software system used at PNNL, MHPCC, University of Utah, and other sites to allocate and manage computing system resources. He holds several advanced AIX certifications and has worked frequently with IBM Software Development in improving functionality and reliability of high performance systems software. He has been highly instrumental in improving PNNL's MPP1 system to register continued utilization improvements of 10% per year since its inception. Mr. Jackson is an active participant of the SP-XXL - an organization of technical computing sites with large IBM SPs whose charter is to work with IBM and each other to tackle the challenges faced by the HPC community. He is also an active participant in the HPC Open Source Working Group as well as in GridForum (scheduling and accounting working groups) and in developing GRID resource management infrastructure.

Stephen Scott
http://www.epm.ornl.gov/~sscott

Biographical Sketch
Stephen Scott is a research scientist in the Distributed Computing Research Group of the Computer Science and Mathematics Division of Oak Ridge National Laboratory (ORNL) – USA. He has received the Ph.D. and M.S. in computer science from Kent State University, Kent, Ohio – USA. He also received a B.A. from Thiel College, Greenville, PA – USA. At Oak Ridge National Laboratory, Stephen's responsibilities include research and development efforts in high performance scalable cluster computing. Primary research interest is in experimental systems with a focus on high performance, scalable, distributed, heterogeneous, and parallel computing. Stephen is a founding member of The Open Cluster Group (http://www.OpenClusterGroup.org) – dedicated to bringing current "best practices" in cluster computing to all users via a self-installing cluster-on-a-CD suite. He is also a contributor to the Parallel Virtual Machine (PVM) and HARNESS research efforts at ORNL (http://www.epm.ornl.gov/pvm and http://www.epm.ornl.gov/harness). He is also a member of ACM, IEEE Computer, and the IEEE Task Force on Cluster Computing. Prior to attending graduate school, Stephen worked various industry positions including one as principal with a business-to-business chemical database startup company.

Selected Publications

M. Brim and S. Scott, "OSCAR: Open Source Cluster Application Resources," 2001 Linux Symposium, July 25-28, 2001, Ottawa, Canada.

M. Brim, G.A. Geist, B. Luethke, S. Scott, and J. Schwidder, "M3C: Managing and Monitoring Multiple Clusters," CCGrid 2001 – IEEE International Symposium on Cluster Computing and the Grid, May 15-18, 2001, Brisbane, Australia.

P.A. Farrell, H. Ong, and S. Scott, "Enabling High Performance Data Transfer on Cluster Architecture," CLUSTER 2000 – IEEE International Conference on Cluster Computing, November 28 – December 1, 2000, Technische Universitat, Chemnitz, Saxony, Germany.

R. Flanery, G.A. Geist, B. Luethke, and S. Scott "Cluster Command & Control (C3) Tools Suite," 3rd Distributed and Parallel Systems (DAPSYS 2000), September 10-13, 2000, Balatonfüred, Lake Balaton, Hungary.

G.A. Geist, S. Scott, and J. Schwidder "M3C – An Architecture for Monitoring and Managing Multiple Clusters," 13th International Conference on Parallel and Distributed Computing Systems (PDCS 2000), August 8-10, 2000, Las Vegas, Nevada, USA.

J. Dongarra, G.A. Geist, J.A. Kohl, P.M. Papadopoulos, and S. Scott, "HARNESS: Heterogeneous Adaptable Reconfigurable NEtworked SystemS," Conference on High Performance Distributed Computing, Chicago, IL, July 1998.

Paul Hargrove
http://www-sccm.stanford.edu/~hargrove

Biographical Sketch

Paul Hargrove received his Bachelor of Arts degree from Cornell University (1994), completing a triple major in Physics (magna cum laude), Math and Computer Science. Since autumn of 1994 Paul has been a Ph.D. student in Stanford University's Scientific Computation and Computational Mathematics program. His thesis research is on kinetic Monte Carlo simulation of annealing and diffusion in semiconductor materials, under the supervision of professor James Plummer of the Electrical Engineering department. He continues to work on his dissertation while employed full-time at Lawrence Berkeley National Laboratory.

Since 1992, Paul has been a user and developer of the Linux operating system. He wrote the Linux implementations of POSIX FIFO files and the ETXTBSY error code, both of which have been included in the Linux kernel distribution since before the 1.0 release. Paul's most significant contribution to Linux is the Macintosh (HFS) file system module, amounting to over 13,000 lines of C code in the kernel code, which has been present in the Linux kernel distribution since version 2.2.0.

Paul worked part-time in the Future Technologies Group (FTG) at Lawrence Berkeley National Laboratory during the Summer of 1999 on the M-VIA project. During that time Paul helped to complete the 1.0 release of M-VIA. In June 2000 he returned to FTG half-time as the lead developer on the M-VIA project. Since September 2000 Paul has been full-time at LBNL and continues to work toward completion of his Ph.D.

Ron Minnich

Biographical Sketch

Ron Minnich is a Technical Staff Member in the Advanced Computing Laboratory at Los Alamos National Laboratory. Ron has been building and delivering clusters since 1991, starting with workstation clusters and moving to PC-based clusters in 1994. Ron's research in clustering has included work in the operating systems area, working closely with SGI, Sun and the Linux and FreeBSD core teams on operating systems enhancements for clusters; the compiler area, building runtime support for parallel compilers for clusters; the applications area, building scalable applications for government and commercial applications; and in the hardware area, in the design of the Memory Integrated Network Interface (MINI), which pioneered many of the concepts commonly in use today in user-level network interfaces. Ron is the inventor of the LinuxBIOS, a project he started 18 months ago at LANL. The LinuxBIOS is a complete replacement for the normal PC BIOS, and boots the PC into Linux in 1-2 seconds. This work has found immediate application in embedded environments, including robotics, network testing equipment, set-top boxes, and, most importantly, in clusters. Currently, two commercial cluster companies have adopted the LinuxBIOS as the foundation of the cluster computing architecture: LinuxLabs and Linuxnetworx. More are on the way.

Ron's most recent work at LANL includes Supermon, a scalable monitoring tool for clusters that reduces monitoring overhead by a factor of 100; the Register Network Interface, a prototype network interface which runs at the same clock rate as the processor; a scalable, secure startup system which embeds SSH client sessions in C++ objects; LOBOS, the first software to show that Linux could boot Linux; a second implementation of private names spaces for Linux (PNS and memdev, available at www.sourceforge.net); and the LinuxBIOS.

Ron has been at LANL since March 1999. Ron worked at the Sarnoff Corporation from 1994 to 1999, and at the Supercomputing Research Center from 1988 to 1994. Ron received his Ph.D. in Computer Science from the University of Pennsylvania in 1991, and his MsEE and BsEE from University of Delaware in 1982 and 1979.

Selected Publications

"The Linux BIOS", Ron Minnich, James Hendricks, and Dale Webster, Los Alamos National Labs, Atlanta Linux Showcase, Oct. 2000

"Interfacing interpreted and compiled languages to support applications on a massively parallel network of workstations (MP-NOW)", Jeremy Kepner, Maya Gokhale, Ron Minnich, Aaron Marks and John DeGood, Cluster Computing: the Journal of Networks, Software Tools and Application, Vol. 3, 35-44

"A private name space system for Unix and its uses for distributed and cluster computing", R. Minnich, First French Conference on Operating Systems, 1999

"The Memory Integrated Network Interface", Ron Minnich, Dan Burns, and Frank Hady, IEEE Micro, Jan. 1995.

"A Radiative Heat Transfer Simulation on a SPARCStation Farm", Ron Minnich and Dan Pryor, HPDC-1, 1992

"Reducing Host Load, Network Load, and Latency in a Distributed Shared Memory", Ronald G. Minnich and David J. Farber, ICDCS-10, 1991

Remy Evard
http://www.mcs.anl.gov/~evard

Biographical Sketch:
Rémy Evard received a B.A. in Mathematics and a B.S. in Computer Science from Andrews University in 1990, and a M.S. in Computer Science from the University of Oregon. During the latter years of his college career and throughout graduate school, he maintained a research relationship with Argonne National Laboratory, working in parallel algorithm development, visualization, climate modeling, and systems administration. In 1992, he took a position as the Manager of Computing Systems for the College of Computer Science of Northeastern University in Boston. While at Northeastern, he completely redesigned and rebuilt the computing and networking infrastructure for the College. As a part of this effort, Evard developed several widely-used open source software packages, published a number of papers, and began to pursue active research into the underlying principles of large-scale systems administration. He moved on to become the College's Director of Technology and started a DOE-funded joint research project with Argonne National Laboratory called "LabSpace" that investigated how small teams could take advantage of persistent collaboration

systems. In 1996, Evard accepted the position of Manager of Advanced Computing and Networking in Argonne's Mathematics and Computer Science Division, taking charge of the division's computing and networking infrastructure and participating in several research and development efforts.

Evard's research interests include scalable system administration, system software, multi-user collaborative systems, and parallel computing. He is one of the leaders of the Argonne Scalable Systems Software Project, and is the principle architect of Chiba City, Argonne's 512-CPU scalable Linux cluster. Chiba City is a highly flexible system designed to support computer science research and scalability testing. His current projects include: "City", an open source release of the tools necessary to manage Chiba City; ANCHOR, the team that manages Argonne's wide-area networking connections; and a multitude of software and infrastructure projects that enable Evard's team of ten engineers to coherently manage 1600 computing systems.

Software Releases Relevant to Proposed Research:
city: a set of utilities for managing large clusters
msys: a set of utilities for managing large computing environments
soft: a simplified approach for large-scale user environment customization

Selected Publications

Rémy Evard, William Gropp, Ewing Lusk, and Rick Stevens, *The Argonne Scalable Systems Software Project*, Argonne Preprint, January 2001.

Peter Beckman, Rémy Evard, and William Saphir, *Production Linux Clusters,* SC99 Tutorials, 1999.

Rémy Evard, *An Analysis of UNIX System Configuration*, in Eleventh USENIX Systems Administration Conference Proceedings, 1997.