LBNL-59999

Software Roadmap to Plug and Play Petaflop/s

Editor: Bill Kramer (wtkramer@lbl.gov)

Contributions from:
Jonathan Carter, David Skinner, Lenny Oliker, Parry Husbands, Paul Hargrove,
John Shalf, Osni Marques, Esmond Ng, Tony Drummond, Kathy Yelick, Bill Kramer

NERSC Center and Computational Research Divisions
Ernest Orlando Lawrence Berkeley National Laboratory
University of California
Berkeley, California 94720

July 2006

This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

Software Roadmap to Plug and Play Petaflop/s

In the next five years, the DOE expects to build systems that approach a petaflop in scale. In the near term (two years), DOE will have several "near-petaflops" systems that are 10% to 25% of a peraflop-scale system. A common feature of these precursors to petaflop systems (such as the Cray XT3 or the IBM BlueGene/L) is that they rely on an unprecedented degree of concurrency, which puts stress on every aspect of HPC system design. Such complex systems will likely break current "best practices" for fault resilience, I/O scaling, and debugging, and even raise fundamental questions about languages and application programming models. It is important that potential problems are anticipated far enough in advance that they can be addressed in time to prepare the way for petaflop-scale systems. DOE asked us to formulate our response by considering the following four questions:

1. What software is on a critical path to make the systems work?
2. What are the strengths/weaknesses of the vendors and of existing vendor solutions?
3. What are the local strengths at the labs? (list of areas of expertise and names of people)
4. Who are other key players who will play a role and can help? (other labs, e.g., efforts at Sandia for Red Storm)

Our response is organized as follows.

Section 1 provides a high-level answer to question #1, *"What software is on the critical path to make the systems work?"* We broadened the response to include both hardware and software issues because the two are so intricately entwined on systems of this scale. We also differentiate near-term (2007) challenges from those that we anticipate in the long term (2008 and beyond).

Section 2 addresses question #2, *"Describe the strengths and weaknesses of the vendors and existing vendor solutions,"* using data collected from the NERSC-5 procurement.

Section 3 addresses question #3, *"What are the local strengths at the labs? (list of areas of expertise and names of people)"* by describing the local strengths at LBNL and NERSC for responding to the challenges of petascale computing described in the earlier sections.

Section 4 addresses question #4, *"Who are other key players who will play a role and can help? (other labs, e.g., efforts at Sandia for Red Storm)"* by identifying key players at other institutions who can be considered key partners for addressing the problems posed in earlier sections.

Section 5 provides supplemental information regarding NERSC's effort to use non-invasive workload profiling to identify application requirements for future systems. The data collected by NERSC may be valuable for proactively identifying bottlenecks in current systems and anticipating future application requirements.

Section 6 describes a set of codes that provide good representation of the application requirements of the broader DOE scientific community. The success of these codes is a bellwether for the overall success of these computing platforms for DOE scientific applications.

Section 7 is a comprehensive production software requirements checklist that was derived from the experience of the NERSC-3, NERSC-4, and NERSC-5 procurement teams. It presents an extremely detailed view of the software requirements for a fully functional petaflop-scale system environment. It also includes an assessment of how emerging near-petaflop systems (XT3, BG/L, Power SP) conform or fail to conform to these requirements.

1. Software Needs for Petascale and Near Petascale Computing

In the sections below we characterize the potential deficiencies in system and application software that will exist in the 2007 timeframe for the three high performance computing (HPC) architectures of interest to the Office of Science. Section 1.1 summarizes the major issues. Section 1.2 looks at longer-term issues that must be addressed before petaflop systems can reach a usable state.

1.1. Critical Issues for Near Petascale (circa 2007)

The following issues lie on the critical path for success of near-petaflops systems in the near term.

- **Fault Tolerance:** The emerging crop of near-petascale HPC systems contain an order of magnitude more components than current-generation systems. Current software and hardware approaches are inadequate to the task of managing and containing the failure modes that are likely to exist in such complex systems. Fault tolerance is primarily limited by the fragile software environment. For example, once an XT3 router chip fails, the torus cannot dynamically change routing; but more importantly, the router, once fixed, cannot be returned to service without a full reboot.[1] The BG/L has similar issues. Hence, the system resources slowly decay until enough nodes are offline that system managers shut down and reboot.

  To address this issue, software reliability has to be improved. The majority of system-wide outages at NERSC are due to software failures. *A fundamental review of system software is needed with the goal of reducing complexity and increasing reliability. Ways to proactively judge software reliability will be critical to deciding where resources should be deployed. Attention to fault tolerance in future software/hardware design must rise to a top priority in order to ensure the success of systems that are 10x larger than today's implementations.*

- **MPI and Support for Legacy Programming Models:** Efficient MPI implementations0 are necessary at increased size. The workloads at NERSC and other Office of Science labs show that most codes can work effectively at the scale of 512 to 4096 processors using MPI. The systems in 2007 will have four times the number of CPUs. The current investment in science codes has to be preserved, so MPI will have to operate effectively at the scale of 10,000–40,000 tasks. *Support for tools and optimizations of legacy programming models must continue even on petaflop-scale systems. Even if all new HPC software development were to shift exclusively to advanced parallel programming languages, it will still take more than a decade to shift the existing HPC software infrastructure over to the new programming methods.*

- **Public Interfaces for Lightweight Communication:** As the number of processors grows and relative cost of the network increases, applications that are communication-intensive must be optimized to make effective use of network resources. Hardware support such as remote direct memory access (RDMA) offers opportunities to overlap communication with computation using lightweight one-sided communication, but interfaces below MPI are often kept proprietary. Hardware issues, such as lack of cache-coherence between network processors and compute processors within a node, also interfere with the ability to use RDMA. The use of unmodified commodity operating systems like Linux can also add overhead due to virtual memory management and other services. Benchmarking efforts (e.g., HPC Challenge GUPS [giga updates per second] benchmark) on the BlueGene/L machine have utilized non-public network interfaces. The LBNL/UC Berkeley Unified Parallel C (UPC) group and the PNNL Aggregate Remote Memory Copy Interface (ARMCI) group have demonstrated the value of RDMA-based benchmarks in both microbenchmarks and application-level algorithms.[2] *Public lightweight communication interfaces*

[1] Note that the XT3 cannot do hot swaps either, so an entire rack needs to be powered down, losing 96 nodes.

[2] See http://gasnet.cs.berkeley.edu/performance for microbenchmark numbers and "Optimizing Bandwidth Limited Problems Using One-Sided Communication and Overlap," C. Bell, D. Bonachea, R. Nishtala, K. Yelick, 20th International Parallel and Distributed Processing Symposium (IPDPS), 2006.

are needed to optimize public domain implementations of MPI as well as global address space models (below) in order to expose the best available network performance on each machine.

- **Global Address Space Models:** While we believe support for legacy programming models such as MPI and OpenMP is important, efficient implementations of current-generation global address space models like UPC, Co-Array Fortran (CAF), and Global Arrays (GA) provide a user-level programming abstraction for the efficient one-sided communication described above. The language-based models offer an advantage over libraries, because they expose parallel constructs to automated optimization and reorganization by the compiler. Otherwise, the semantic meanings of application program interface (API) calls (such as MPI) are opaque to the compiler's optimizer and therefore cannot be optimized in any fashion by the compiler. *Global memory languages are poised to play an important role **now** (in the near-petascale time range) for enabling effective use of petaflop-scale systems. UPC compilers are available on every major HPC system architecture, from PC clusters to the X1e, including prototypes for the XT3 and BG/L. LBNL is partnering with Cray and Intrepid to provide optimized UPC support for the XT3; and IBM has an internal research compiler for UPC on BG/L. CAF has a portable open-source compiler from Rice, and GA is available from PNNL. These languages have nearly achieved the level of ubiquity necessary to change the software ecosystem.*

- **Stable Parallel Filesystems:** Current global, parallel filesystems are demonstrated to run effectively at the scale of 2,000 clients. There are no other practical solutions in the near term aside from GPFS and Lustre. Lustre has shown significant scaling and reliability issues. GPFS, while showing good scaling, excellent performance, and excellent reliability for concurrencies up to 2,000 clients, is just now starting to move onto some of the near-petascale architectures. However, GPFS performance and reliability remains relatively unexplored at concurrencies in excess of 2,000 clients. If the current implementations are just scaled to the number of compute nodes in the pre-petaflop systems, it is likely that significant scaling issues will be encountered, even show-stopping problems. Several solutions need to be explored, including redesign of the filesystems and implementation of lightweight interfaces on compute nodes. *The filesystem problems identified in this section cannot be solved by the DOE Labs, even if the software is open-sourced. Any credible solution requires close collaboration between the labs and the two primary vendors who have developed scalable parallel I/O system technologies — collaboration that is supported by funding to proactively target these issues.*

- **Numerical Libraries:** Additional focus is required to ensure the scalability of parallel libraries. Much of the current activity in tuning performance focuses on single processor performance, which will continue to be important; but higher-level performance issues such as communication performance, load imbalance, and synchronization overhead need to be addressed. The level of sophistication required for parallel libraries on petascale systems will be dramatically larger than on current lower-parallelism and flat interconnect topologies. Funding of advanced library development that *leads* rather than lags the deployment of these systems will be essential to ensure the systems are used effectively for scientific applications once they come online. HPC centers will need to work with library and tool developers to assemble software testbeds to test new platforms, document the results, and systematically accumulate a list of requirements and expertise on the software and platforms where it runs. Library and tool developers should be among the community of early pre-production users on the near-petascale systems, along with application-focused early users, and should aggressively use this access to port and test their libraries and tools and to identify problems at the level of the tools (such as numerical tool implementations, precision, etc.) and applications codes. Early detection and resolution of problems through consistent validations can make the science community more confident in petascale systems. *Both users and library developers have underscored the need for robust software testbeds that allow them to proactively port their software to new platforms using smaller-scale early-release versions of candidate HPC system hardware. A program that enables centers to purchase non-production early/prototype hardware could play an important role in establishing advanced software testbeds to prepare our software infrastructure for eventual full-scale implementations based on the same system architecture.*

- **Debugging:** There is no debugging solution for petascale systems. Totalview does not work for users above 1,000 tasks, and only works on one near-HPC system beyond 1,000 nodes. Furthermore, the Etnus cost model is extremely expensive at scale. Continuing in the current direction for development for this software technology is untenable. *In the short term, it will be necessary to push vendors such as Etnus to extend the capabilities of Totalview to support larger-scale platforms, but the costs may be impractical. Ultimately this issue requires renewed exploration of alternative approaches to debugging at massive concurrency. Targeted funding is required to reinvigorate the research pipeline in order to spawn new ideas in this area. This point is explored in more depth in section 1.2 as a long-term issue.*

- **Checkpoint/Restart:** Operationally, C/R provides many advantages that make large systems more effective. By implementing C/R on both the Cray T3E and the IBM SP, NERSC increased its ability to run the right job at the right time, allowing long-running jobs while still providing fair throughput and reducing the impact of system shutdowns. C/R is feasible at scale on all the systems, but the necessary software is only available for the IBM Power architecture at this time. Linux C/R, developed through the DOE FastOS effort, may find its way into the XT3 and BG/L compute nodes if and when they move from microkernels to stripped-down Linux kernels. Meiosys is another possible implementation of C/R being considered by at least one vendor. *While this move presents an opportunity for the Linux C/R efforts, it will require software integration and support to ensure production-quality services at this scale. It will require renewed support for vendor/laboratory collaborations to bring this work to fruition.*

The following problems are lower priority. They will greatly affect the usability and efficiency of the system, but will not necessarily be show-stoppers to making systems work for production-quality science. Systems without these functions merely waste DOE effort invested in these architectures.

- **Changing Memory Balance:** Memory is one of the most expensive and power-hungry components of systems. For the past 15 years, systems have consistently hovered around a byte-per-flop (B/F) ratio of 0.5 byte per peak flop delivered by the system, despite the lack of firm scientific grounding for this ratio. Historically it has been associated with Amdahl's heuristic design point for mainframes that called for one byte of memory per instruction/sec[3] processing performance. As systems move to having thousands to tens of thousands of nodes, it may not be practical to continue the current memory balances on future systems and still have enough funding to reach the computational goals. It is unclear to what extent this design trade-off would impact the design of future applications and numerical algorithms. *The scientific computing community needs to develop the appropriate framework to address this issue in a rational and scientific manner. The process can be informed by the kind of in-depth analysis of algorithms being performed by groups such as Oliker's benchmarking team at LBNL; Worley, Dunigan and Vetter at ORNL; and the SciDAC PERC collaboration. These efforts can provide the ultimate answer to this question if it is identified as an important issue, but must be chartered explicitly to investigate the B/F ratio for the Office of Science workload.*

- **Efficient Job Migration and Torus Packing:** In order to reduce performance variability on architectures with a 3D torus topology, jobs must be given topologically contiguous sections of the torus. As jobs are scheduled and retired, gaps develop in the torus that must be removed by migrating running jobs to repack the torus. Such migrations require many of the same capabilities as OS-initiated checkpoint/restart, but the job state need not be written to and from disk. Integrating efficient job migration and job packing algorithms with the batch subsystem will be important for efficient utilization of these system architectures. NERSC implemented such capabilities for the T3E/900 system (mcurie). *Recreating this capability requires much of the C/R capabilities as a minimum requirement, and must be complemented by an open/modular resource management system that can be modified to direct the job suspension and process migration facilities to effect job migration in response to current job scheduling conditions. The XT3 will present a problem in this regard due to lack of explicit control of job layout in the current system software implementation.*

[3] This is a non-floating-point measure.

- **Job Mapping:** BG/L and the XT3 require very complex job mappings for anything except the simplest of problems.[4] Currently users have little information to select job mappings other than the canonical mapping. There are few tools to assist with selecting a better mapping, much less automatically generating map files for the user (who is left to write scripts to generate the map files, if possible). Currently available monitoring tools do not provide useful advice on how to improve the job mapping or are not practical for full-scale utilization. *Funding to develop lightweight, non-invasive systems for collecting communication and performance data from the jobs while they run at production scale will be essential to make more efficient use of these resources. Such performance monitoring frameworks must be non-invasive enough that they can be used for* ***all*** *jobs run on the system rather than as an offline performance optimization/tuning step (a key example is described in section 5 of this document). Combining these capabilities with the facilities to re-pack jobs on the interconnection network will greatly improve resource utilization efficiency.*

- **Parallel I/O:** MPI I/O may be insufficient at this scale unless POSIX[5] semantics can be relaxed. Some form of asynchronous I/O or other support for relaxed POSIX compliance will be critical to scalable I/O performance. This capability must be exposed efficiently via mainstream I/O libraries like Parallel NetCDF or parallelHDF5. This is not a show-stopper because users will continue to follow current practice of writing one file per processor. The result of such user behavior is overblown storage system performance requirements (such as the 30k file creates/sec required by DARPA HPCS) that will otherwise increase costs and reduce the efficiency of archival storage systems such as HPSS. *DOE labs can help by continuing R&D of parallel file formats and libraries such as MPI-I/O, more efficient parallel-I/O alternatives such as server-directed I/O, expanding the effectiveness and portability of parallel global filesystems that allow parallel writes to single files in an efficient manner, and refining a strategy for relaxed POSIX I/O. The latter requires close collaboration with industry and some organized effort to promote standardization of the approach across multiple platforms. Refinements in the lower-level parallel I/O strategy must be complemented by comparable funding for the incorporation of those advances into high-level file storage organization such as pHDF5 and Parallel NetCDF.*

- **Visualization and Data Analysis Infrastructure:** The ASC VIEWS program has provided a good set of baseline tools to attack this problem, such as LLNL VisIT. However, scientists require constant evolution in these baseline tools, tailoring their capabilities to meet the specialized visualization and data analysis requirements of their scientific domain. Tool development for data analysis and visualization is a continuous optimization process, requiring close collaboration with the domain scientists. VisIT offers a substrate to build upon that meets baseline needs, but such tools do not fix or extend themselves. Failure to consider the requirements of data analysis (both hardware and software resources) **together** with the large-scale system procurement will render any such system ineffective. *DOE needs to preserve the investment in data analysis software infrastructure by continuing to fund collaborations between the providers of tools such as VisIT and Paraview, and the application stakeholders.*

1.2. Critical Software Issues for Petascale (circa 2010)

1.2.1. Fault Tolerance

The emerging set of near-petascale systems present a two-orders-of-magnitude leap in scale over typical systems today, in terms of hardware components (total number of disks, processors, and DRAM sticks),

[4] Almasi, Bhanot, Gara, Gupta, Sexton, Walkup, Bulatov, Cook, de Supinski, Glosli, Grennough, Gygi, Kubota, Louis, Spelce, Streitz, Williams, Yates, Archer, Moreira, Rendelman, "Scaling physics and material science applications on a massively parallel Blue Gene/L system," Proceedings of the 19th Annual International Conference on Supercomputing, ICS 2005, Cambridge, Massachusetts, June 20–22, 2005, pp. 246–252.

[5] Portable Operating System Interface for UNIX: a set of standards from IEEE and ISO that define how programs and operating systems interface with each other.

which is causing a critical gap to open up in fault management of these systems. Most of the systems have hardware designs that include many features that improve fault tolerance, but critical failure modes persist. Currently, systems software components for large-scale machines remain largely independent in their fault awareness and notification strategies. Software implementations remain fragile and do not support many of the hardware RAS (Reliability Availability Serviceability) features, so operationally, there is limited benefit for the hardware features. Faults can arise not just from the hardware but also from the OS, middleware, and application levels. Failures typically get reported using rudimentary error conditions such as "job failed," that provide little indication of the root cause of the failure. Given the overwhelming complexity of emerging petascale HPC applications, such opaque failure modes may well render the system unusable. Moreover, the multiple layers of software between application and computer have little to no opportunity to report, avoid, or correct the issue.

As mentioned above, the work in hardware fault tolerance is being limited by the fragile software. For example, once an XT3 router chip fails, the torus cannot dynamically change routing; but more importantly, the router, once fixed, cannot be returned to service without a full reboot.[6] The BG/L has similar issues. Hence, the system resources slowly decay until enough nodes are offline that system managers shut down and reboot.

To address this, software reliability has to be improved. The majority of system-wide outages at NERSC are due to software failures. A fundamental review of system software is needed with the goal of reducing complexity and increasing reliability. Ways to proactively judge software reliability will be critical to deciding where resources should be deployed.

Checkpoint/restart is a primary approach for fault-tolerance. It helps preserve application work and also helps system effectiveness when system shutdowns are needed. System-wide checkpoint becomes increasingly important because any solution that attempts to improve on fault-tolerance over C/R is likely to creep into the programming model. For instance, even in the most likely case of moving to application-initiated C/R, there is a need to annotate the minimum amount of state that must be preserved. Entire system C/R is feasible at scale.

However, the time to do a checkpoint is dominated by the size of memory and the I/O bandwidth. Since bandwidth increases more slowly than memory capacity, the time for checkpoints will continue to increase. Hence, it will become important to have incremental/journaled checkpoints. This is analogous to full and incremental checkpoints. You only do a full checkpoint for an application occasionally, and you do incremental checkpoints that reduce the amount of I/O needed. Restart would be longer, having to read in both the full and incremental checkpoints, but that may be more effective.

It is necessary for the batch system to handle the restart, but it is not possible to explain to a batch system in a concise manner where to locate the restart files and how to invoke the application to perform the restart unless there are some common guidelines to enforce uniformity. If you expect one processor failure every 10 minutes, it is a non-solution if the user must intervene in the restart process. Uniform software frameworks must be developed to support uniform application-level C/R if this approach is ever to be workable.

The problems with extending C/R to future systems suggest a need to build an infrastructure that enables systems to adapt to faults in a *holistic* manner. Such a system would provide common uniform event handling and notification mechanisms for fault-aware libraries and middleware. Applications would need to interface with these capabilities in a more seamless manner.

Attention to fault tolerance in future software/hardware design must rise to a top priority in order to ensure the success of systems that are 10x larger than today's implementations. The focus should be on detecting deviant behavior in addition to catastrophic failure. Whereas current system software error detection mechanisms remain largely independent of the error handling at system and hardware level, a more integrated approach to the detection, propagation, notification, recovery, and even prediction of errors conditions is necessary. Given the realities of component failure rates, applications and underlying system services will increasingly need to adapt to fault conditions and take defensive action to recover from faults

[6] Note that the XT3 cannot do hot swaps either, so an entire rack needs to be powered down, losing 96 nodes.

rather than die with a cryptic error when any fault is detected. Methods like statistical learning theory should be explored to provide very early warning indicators from user behavior and application results.

More work on alternatives to C/R needs to be launched in the research community. Developing methods to proactively measure and predict reliability, particularly for software, is another research area. A holistic view of reliability demands that fault detection, notification, and recovery mechanisms be integrated across both the OS and application domain in a uniform manner in order to enable a more holistic and comprehensive approach to fault resilience.

1.2.2. Application Coupling and Multiphysics Applications

Petascale computing platforms enable physical processes to be simulated with much higher fidelity. The structure of such computations will likely increase the number of applications requiring complex couplers to join simulation components that employ incompatible domain decompositions or underlying numerical algorithms to simulate a broader array of physical processes. There are many aspects of this problem that are now of interest. As the coupling of models becomes more complex, it becomes more prohibitive to simply combine two different codes into one; but rather the trend has been to develop software paradigms to make them interoperate (like MCT, DCT, CCA or other more specific software frameworks). However, the execution control part of the problem has not been optimally addressed by vendors or the middleware, given the complexity of the problem. *So, we need to develop more robust versions of existing coupling toolkits and ways that one can control the launching of multi-executables that need to exchange model variables, fields, or data online, ensuring that these coupled models run scalably and reliably on massively concurrent systems. The capabilities of such systems are likely to make multi-physics couplers more common in the future.*

1.2.3. Analytics, Visualization, and Data Management

As we move towards larger-scale systems, the issue of data management (data storage and movement to different data storage resources) will become intimately entwined with every aspect of the data analysis and visualization process. There is a tendency to separate the concerns of data analysis from those of the core computing system, but this approach cannot be sustained on the largest-scale systems. The ASC program took pains to ensure that visualization and analysis requirements were intimately linked to each new system by suggesting the data analysis resources be provisioned as at least 10% of the scale of the primary computing system. Anecdotal evidence from the scientists served by the ASC program indicates this metric has served them well. The ASC metric for scaling data analysis resources as a fraction of the size of the primary computing platform should be examined to determine if the ratio makes sense for petaflop-scale systems. Likewise, investments in advanced and scalable visualization technology must be preserved through funding of joint development efforts between science groups and visualization technology providers. Otherwise, the current rewards system of these respective groups makes the natural emergence of effective data analysis solutions unlikely.

1.2.4. Massively Parallel Programming Models

Most applications remain unprepared for the level of concurrency exposed in petascale systems and their precursors such as the XT3 and BG/L. Advanced programming languages will likely play an essential role in exposing/expressing enough concurrency to take advantage of the myriad of processors. The path to scalability is increasing the number of user-visible threads, because hardware techniques such as pipelining and instruction level parallelism have reached their practical limits.

Some significant issues are:

- **Load Imbalance:** Problems with any degree of irregularity or load imbalance such as unstructured grids (e.g., UMT2K), block-structured AMR (e.g., Chombo and SAMRAI), and sparse linear algebra (e.g., SuperLU, MUMPS) exhibit lower scalability with current methods. Application-level load imbalance will be exacerbated on petascale systems, as all processes must wait for the slowest to arrive at synchronization points, and the dynamic load balancing required for some applications require asynchronous background communication or other techniques to hide their cost. Advanced dynamic partitioning software such as Zoltan and graph partitioners such as Metis can play a role in a software strategy to detect and react to load imbalance

- **Small Messages:** Increasing the number of processors in a system, especially in an environment with small memory per processor, will lead to smaller messages. The 3D FFT is an example where message sizes get smaller (send nprocs messages to each processor in 1/nprocs) as concurrency scales up. To do better at spreading communication over a longer period of time to reduce bisection bandwidth requirements, one must employ even smaller messages. The overhead associated with sending each message has a significant impact for small messages that are required to spread out the communication costs, as described in the Chen/Iancu/Yelick paper on fine-grained communication optimizations for UPC.[7] Current best practice is to aggregate messages into larger buffers, but that approach increases software complexity and cannot be extrapolated to systems with such massive concurrency. *Hardware that provides fast communication with low CPU overhead will allow for more effective use of networking hardware. In addition, these facilities must be exposed to user-level software, ideally in a common communication interface.*
- **Topology:** Codes with large global communication, like 3D FFT and particle-in-cell methods, require extra attention to communication locality and the topology of the underlying interconnection network. Experience with BG/L has already highlighted the important of topology optimizing communication performance, and this is likely to become more significant as machines scale. MPI libraries need to optimize for topology, and machines should expose information to allow application programmers to optimize as well.
- **Multithreading:** The primary path to exploiting hardware threading currently involves mixed-mode OpenMP + MPI programming (OpenMP to exploit threading on each node and MPI to span the distributed memory domains). However, OpenMP offers a rather limited approach to expressing threading concepts that are completely disjoint from the considerations of the message-passing model. Few programmers are willing to take the additional step of employing two-level parallelism in this manner. Languages that offer a more natural/readable approach to exposing parallelism are required. Vendors can implement Unified Parallel C (UPC) and Co-Array Fortran (CAF) in this time frame, but they offer only an interim solution to exposing the necessary degree of parallelism required to make petaflop-scale systems even marginally useable.

Recommendation: Continued development of advanced programming models in close collaboration with vendors and application scientists is essential to ensure that the hardware/software architecture for messaging on future systems will meet the needs of advanced programming models, and that the programming models meet the needs of the users. Cross-agency and vendor coordination is also essential to ensure ubiquity of emerging programming models.

1.2.5. Libraries

Efficient libraries are essential for large-scale systems. There is too much software complexity for each individual application group to take on alone (unless their application is exceedingly simple). The community must leverage the numerical algorithm development and performance tuning work that goes in to library solutions.

The increasing sophistication of computer models together with the complexity of algorithms and computer architectures greatly inhibits progress towards optimized version of an application. In addition, a development team may already be faced with software intricacies that cannot be undertaken by the team alone. Together, all these issues may be even more challenging on large-scale systems.

Libraries can provide a convenient way for incorporating state-of-the-art computational technology into development efforts, with the additional benefit of alleviating many aspects of the development process and enabling testing of different techniques or implementations.

[7] Wei-Yu Chen, Costin Iancu, Katherine A. Yelick, "Communication Optimizations for Fine-Grained UPC Applications," IEEE PACT 2005: 267–278.

While some users have been able to exploit serial libraries, e.g., calling non-parallel libraries such as FFTW within parallel applications, the degree of sophistication required to achieve scalable performance on mesh interconnects (such as the torus networks on BG/L and the XT3) far exceeds current requirements. This will necessitate more focus on the largest scale.

While library autotuning might be viable in some cases, overall it is likely to be impractical on large-scale systems since it may require increased validation and testing to make sure the libraries are functioning properly. The search space for autotuning is greatly expanded if parallelism is admitted into the search space on the same level with scalar optimizations. This may require a hierarchical approach to autotuning of parallel applications or it may require limiting the range of autotuning.

To identify the main libraries used in simulation codes, provide the resources for early porting of such libraries to emerging large-scale architectures; identify eventual bottlenecks in the libraries and recommend alternative solutions.

We reiterate the importance of providing targeted funding for massively concurrent system libraries and programming abstractions. The development of such libraries must anticipate the arrival of the systems (using smaller-scale proxies or early prototypes to assist in development) rather than spin up after such systems have been delivered. The lead time required for software development is considerable, so a close interaction between the research, evaluation, and prototype program and developers of advanced software technology is essential for the success of future systems.

1.2.6. Debugging

Although some tools like Umpire[8] and MARMOT[9] have been developed to detect faulty use of MPI functionalities, it is fair to state that there is no practical debugging solution for problems scaled beyond thousands of processors. Although users do want to employ debugging functionalities on more than 512 processors, existing solutions such as Totalview are inadequate. To the best of our knowledge, there is currently is no effort underway to replace Totalview and there is little motivation for Etnus to invent and test for higher scales. At the same time, there are no responsive solutions in the pipeline from the research community. An alternative approach to debugging large-scale codes may be higher-level toolkits that go beyond stepping through code with interactive debuggers, print statements, and time stamps that are cumbersome to look at when running on hundreds of processors. There is a need for tools that can replicate scenarios and program failures with user-friendly interfaces.

1.2.7. Operating System

OS interference is a well documented problem on systems such as ASCI White and NERSC-3. The microkernel approach on the XT3 and BG/L mitigates OS interference problems. However, XT3 and possibly BG/L will be moving to stripped-down Linux kernels. This has ramifications on reliability/uptime and possibly OS interference on large-scale systems, and requires additional planning and attention as we lead up to production system deployment. OS interference is potentially *very* dangerous, with show-stopping consequences if we do not proceed with deliberate caution and preparation.

While considered solid for desktop systems, the reliability of Linux on a massive scale remains questionable. Furthermore, the complex web of hardware/driver dependencies makes any OS upgrade or modification challenging for systems that use a Linux or UNIX base. The planned move of the XT3 and BG/L compute nodes from microkernels to Linux implementations presents a high degree of risk, particularly if the Linux community refuses (as they have done many times in the past) to adopt extensions that might improve the reliability and effectiveness of Linux on large-scale systems.

[8] J. S. Vetter and B. R. de Supinski, "Dynamic Software Testing of MPI Applications with Umpire," Proc. SC2000: High Performance Networking and Computing Conf. (electronic publication), ACM/IEEE, 2000.

[9] B. Krammer, M. S. Muller and M. M. Resch, "Runtime Checking of MPI Applications with MARMOT," ParCo 2005, minisymposium on Tools Support for Parallel Programming, Malaga, Spain, Sep. 12–16, 2005.

1.2.8. Profiling and Performance Analysis

There are tools that have been used successfully to analyze performance data on thousands of processors. In particular, TAU has been used to generate and analyze profiling information of a CFD code on 16 K nodes on LLNL's BG/L. That same data was later replicated four times in order to simulate the profiling of the application on 64 K processors and test the scalability of TAU. The last version of Paraprof, TAU's visualizer, offers 3D view functionalities (such as triangle mesh plot and scatter plot) that can help users look into performance data obtained on a large number of processors.

Another solution for performance analysis is IPM (Integrated Performance Monitoring), which is discussed in detail in section 5. The primary contribution of IPM is that it presents a lightweight approach to application monitoring. The overheads (< 3% typically) are low enough to enable continuous monitoring and data collection for a system workload. This has enormous benefits for better characterizing the workload at a given center in order to feed back into the design requirements for future systems.

It remains to be investigated, however, how tools such as these behave on a wide set of large applications, the overhead resulting from their use, their performance, and their capability in helping the user understand enormous collections of profiling and tracing data.

1.2.9. Parallel I/O

Parallel I/O for petascale systems raises several software issues:

- **Bandwidth:** In hardware, bandwidth is a factor of the number of I/O servers, the number and capabilities of the controllers, the number of storage units, and the number of connections from disk devices. Bandwidth also depends on the ability of the software to manage the control of operations. The software control is the primary area to attack, since the other constraints are a matter of buying more hardware.
- **Reliability of the Underlying Storage Medium:** High performance disk systems have a price point well above commodity disks. Much of the cost is in the controller, which has embedded software. Trying to use disks, such as SCSI, that have lower duty cycle design points raises reliability issues. Current filesystem software stacks will be unable to cope with the move to less reliable storage devices, so fault recovery and resilience pathways must be reexamined in the light of this impending hardware constraint.
- **Metadata Performance:** Standard operations such as simply running a cron job that purges old files (clean scratch) will be impractical on a filesystem with hundreds of millions of files.
- **File Create Performance:** HPCS 30 K file creates/sec is considered unreasonable, but may be a practical requirement for these systems unless concurrent file I/O performance problems are addressed.

Facility-Wide File Systems (FWFS)[10] provide consolidated storage for online user data, replacing traditional system-local parallel filesystems for home directories, scratch storage, and project storage. While an FWFS is external to all computational systems, it is mounted natively and at high performance. FWFS grow and evolve over time, serving several generations of computational systems.

The benefits of FWFS to scientific productivity are manifold. By providing a single unified namespace, FWFS will make it easier for users to manage their data across multiple systems. Users will no longer need to keep track of multiple copies of programs and data; they will no longer need to copy data between NERSC systems for pre- and post-processing. Storage utilization will become more efficient through decreased fragmentation. Computational resource utilization will become more efficient as users can more easily run jobs on an appropriate resource. Storage allocations (quotas) will become larger, because they will no longer be fragmented among several systems. FWFS will also provide improved methods of backing up user data that can mitigate disturbance to users when block backups are run.

[10] In FY2005, NERSC began deployment of a Facility-Wide File System (FWFS)

Anticipated developments in filesystem technology will provide further benefits. It is expected that FWFS will provide Hierarchical Storage Management (HSM) functionality, in which data can be automatically migrated to and retrieved from tertiary storage. This will further improve scientific productivity by enabling the user perception of extremely large online disk (very large quotas) and making it unnecessary for users to manually transfer data to and from HPSS. Other possible technology enhancements include enhanced security, wide area access, and tighter integration with grid technologies.

GPFS is the leading candidate because it has performance, scalability, reliability, better security, can be geographically distributed, and is interfaced to two different archive systems. Unfortunately, GPFS is not ported to all potential petascale systems. Lustre is an evolving candidate that currently has no plans for archive integration and presents security concerns and reliability issues.

In the petascale domain, there will be tens to hundreds of terabytes of data files. Moving them between systems in a facility will be difficult and time consuming. Indeed, some project teams currently devote 0.5 to 1 FTE just to move files to the right place. *Hence it is essential that petascale vendors support filesystems that integrate well, and at high performance, into facility-wide filesystems. However, it is important to encourage some level of competition in this area as well, so DOE should fund work in at least two global, parallel filesystems.*

Alternatives to MPI/IO such as Server Directed I/O that were developed for the database community should be reconsidered. These approaches were more effective than the imperative approach of MPI-I/O because the data was committed to disk using fences that offer a greater opportunities to overlap computation with communication. It is unfortunate that the techniques were overlooked by the HPC community.

1.2.10. Network I/O

Computational switches are based on very large frame (packet) sizes, typically 64 KB, and suffer performance degradation at lower sizes. Local-area networks based on Gigabit Ethernet can run up to 9 KB jumbo frames reliably for onsite mass storage and backups. The 9 KB frame size increases performance compared to standard Ethernet by reducing overhead and CPU load; but compared to 64 KB frames, it has lower performance and higher latency than computational switches. Wide-area networks typically are restricted to 1,500-byte frames because of the need to support millions of simultaneous connections, even though the main protocols — 10 Gigabit Ethernet, SONET, and ATM — will pass frames up to 9 KB.

This mismatch in maximum transfer unit (MTU) sizes, and more importantly the performance degradation that is associated with running small MTUs, makes it difficult to effectively integrate a massively parallel computer system into any networked environment. Furthermore, internal interconnects do not run IP but instead use switch-specific protocols such as LAPI and Portals. The two major approaches used to date have been to add an external interface into each node or to turn one or more computational nodes into gateway routers and have external traffic flow across the internal switch to these gateways.

Both of these approaches have major weaknesses. Petascale computing will require thousands to tens of thousands of nodes.[11] Adding an external gigabit interface to every compute node for external connectivity to data storage systems and other computational resources is not practical, nor will it allow single stream performance to increase above current levels, even though this has the best chance for meeting large aggregate bandwidth requirements. Using a compute node that has a connection to the internal switch fabric as well as multiple bonded Gigabit Ethernet interfaces or 10 Gb/s Ethernet as a router is logistically easier and potentially has better single stream performance, but aggregate performance will suffer, especially coupled with small MTU traffic on the computational switch. Further, not all systems have nodes that can drive 10 Gigabits at near optimal rates. Also, a compute node that has to run packets through its IP stack to divide the traffic into packets, generate packet headers, and perform flow and congestion control

[11] C. William McCurdy, Rick Stevens, Horst Simon, et al., "Creating Science-Driven Computer Architecture: A New Path to Scientific Leadership," Lawrence Berkeley National Laboratory report LBNL/PUB-5483 (2002), http://www.nersc.gov/news/ArchDevProposal.5.01.pdf.

will never be able to keep up with the fastest switches and routers that just store and forward with all decisions in application-specific integrated circuit (ASIC).

One possible solution would be a Layer-7 router with a computational switch interface that could do the bridging, MTU repackaging, and load balancing. This would be directly connected onto the vendor's internal interconnect. No potential petascale vendor has this in plan. A more workable solution would be to modify a compute node to act like a high performance router. Most of the hardware components for modifying a compute node already exist or would be simple to create. A computational switch, such as a Federation SMA3 adapter, and a 10 Gigabit Ethernet card, both with sufficient field-programmable gate array (FPGA) space to offload the IP protocol, would limit the compute nodes duties to setting up remote direct memory access (RDMA) between the two interfaces. Frame fragmentation and coalescing are often done through TCP proxies but should be programmable in a reasonable amount of FPGA space, thus reducing the adverse impact of widely disparate MTUs.

1.2.11. Transport Protocols

TCP's congestion control algorithm aims to fully utilize the network path yet be fair to other traffic. There are two types of losses in the network: random and congestion. If network traffic arrives at a router or network interface and there is not enough capacity left to buffer the packet, then the packet is discarded (congestion loss). In TCP, the receiver acknowledges data as it is received. When a sender receives three duplicate acknowledgements, it assumes that data has been lost, cuts its sending rate in half, and retransmits the data just above the duplicate acknowledgement. It then increases the sending rate by one each round-trip time until the next loss, and the cycle repeats. This algorithm is called additive increase, multiplicative decrease (AIMD). TCP also includes a slow start algorithm, which is used at the beginning of a TCP connection to double the sending rate each round-trip time until the first loss is detected. TCP uses a congestion window to track the sending rate that is allowed. If the buffer size for sockets is not sized appropriately for the network connection, the congestion window might be artificially limited or the receiver might be overrun. Ideally the buffer size is continually tuned to the optimal size. The Net100 project has created a workaround daemon to perform this dynamic tuning[12].

Network gateway nodes are often the limiting factor in network performance from large systems. It is tempting to assume that the host throughput to and from the network is simply the slower of the I/O bus speed or the memory bus speed, but the reality is more complex. The real throughput that can be achieved in transferring data from the user memory on the machine to the network interface card is defined by adding (1) the time to copy data from user memory to the kernel memory across the memory bus and (2) the time to copy from the kernel memory to the network interface card. Typically it takes two memory bus cycles to copy data from the user memory to the kernel memory, and one I/O bus cycle to copy from the kernel memory to the network interface card. The number of memory bus cycles required to transfer a word from the kernel memory to the I/O bus is determined by dividing the memory bus speed by the I/O bus speed. So, the typical throughput between the user memory and the network interface of a typical PC is defined by the following equation:

$$\mathrm{Throughput} = \frac{\mathrm{MemoryBandwidth}}{2 + \frac{\mathrm{MemoryClock}}{\mathrm{IOBusClock}}} \quad (1)$$

The analysis can be found in [[13]] but the bottom line is that most x86 systems with the fastest available memory and PCI bus are not able to provide enough I/O power to drive 10 Gb NIC at full speed. PCI Express systems will do better, but will still be insufficient in the petascale domain.

[12] T. Dunigan, M. Mathis and B. Tierney, "A TCP Tuning Daemon," *Proceedings of SC2002,* November 2002; http://www-didc.lbl.gov/papers/net100.sc02.final.pdf.

[13] William Kramer, Deborah A Agarwal, Arie Shoshani, Brent Draney, Guojin Jin, Gregory Butler and John Hules, "Deep Scientific Computing Requires Deep Data," IBM Journal on Research and Development, Volume 48, Number 2, March 2004.

Several mechanisms have been created to allow applications requiring high bandwidth to have priority access to the bandwidth. The most aggressive of these is to create a dedicated path for the traffic by reserving a dedicated link/circuit/channel. The best-known mechanisms for this include virtual circuits and RSVP. Another mechanism is to mark the traffic as priority; then each router in the path expedites the traffic. The standard method for this is to use a bandwidth broker to arbitrate the access to the bandwidth on a pair-wise basis. If the intervening routers agree to the priority path, then the packets get marked as priority as they enter the network, and each router in the path forward gives the packets preference over all other traffic. An alternative approach that has been proposed is use of priority ratings on traffic or reserving virtual circuits. Although prioritizing traffic has received a lot of attention, it has generally been impractical. It is likely that traffic requiring guaranteed bandwidth will need to reserve virtual circuits through the network. This does not, however, remove the need to find a transport protocol that can effectively make use of the dedicated bandwidth.

These mechanisms are not supported on the software roadmaps of petascale systems.

2. What Are the Strengths and Weaknesses of the Vendors and of Existing Vendor Solutions?

2.1. Cray XT3

Issue	Mitigation
The XT3 interconnect shows quite high latency differences for very large installations. This is a potential problem for parallel applications writers.	It is not clear how much can be done with software or communication libraries. It would seem ANL, with their experience in MPI, might be best suited to address this. NERSC/LBNL are best posed to offer programming and algorithm expertise to science projects.
The XT3 interconnect is quite fragile, with little ability for software recovery. Once a node/router goes out of service, it is not possible to return to service without a full system boot.	This takes low-level system and interconnect software work to improve. LBNL and ANL have expertise in these areas
Lustre may have issues with stability, performance, scaling, and interoperability as compared with other high performance global parallel filesystem.	NERSC/LBNL have several years of experience with Lustre, GPFS, and other filesystems due to our Global Unified Parallel File System (GUPFS) and NERSC Global Filesystem (NGF) efforts. While Lustre will be made reliable eventually, GPFS appears to be a very feasible option and Cray is open to it.
Cray lags behind in compliance for the latest Fortran standard.	This is more complicated now that Pathscale has been purchased.
Interconnect latency lags behind commodity interconnect in this timeframe. While this is software, it is very low-level software.	It is not clear how much can be done with software or communication libraries. It would seem ANL, with their experience in MPI, might be best suited to address this. NERSC/LBNL are best posed to offer programming and algorithm expertise to science projects. LBNL's Berkeley Institute for Performance Studies (BIPS) effort would be crucial to understanding the limitations.
Memory available per MPI process is low compared with other architectures. This may cause some applications to have to be redone, and certainly puts pressure on any system-level efforts.	This is basically a programming and testing effort.
The successor to Catamount, Compute Node Linux (CNL), has an almost non-existent roadmap at this time. In particular, there are no requirements regarding memory and cpu cycle consumption.	Cray has only general goals and plans for CNL. LBNL, ANL, and ORNL all have expertise that could help with implementing CNL. NERSC has expertise in identifying and dealing with memory and CPU intrusion of operating systems.
Basic user and job resource management controls are very limited (quota, resource limits, etc.). Advanced workload management features such as checkpoint/restart, job migration, and preemption have no clear roadmap.	NERSC implemented the first highly parallel checkpoint/restart, and was the first site to use it on IBM systems at scale, as well as implementing excellent workload management methods. This experience, combined with LBNL's Berkeley Lab C/R (Hargrove and Duell), makes LBNL/NERSC the best place to address this.

Tools for large-scale debugging, analyzing, and tuning are lacking. E.g., Totalview only scales to 512 CPUs at this time and will at best be at 1024 in the 2007 timeframe, ~5–10% of the maximum job size.	Large-scale debugging has to be rethought from scratch. Totalview is a very expensive monopoly that is not really effective. ANL, ORNL, and LBNL have expertise here

2.2. IBM Blue Gene

Note: BG/L is the system that NERSC/LBNL knows the least about. There are probably a lot more issues than listed here.

Issue	**Mitigation**
With a choice of NPFS, Lustre, and GPFS, there is a variety of high performance global parallel filesystems available. The very large number of processors could cause scaling issues.	NERSC/LBNL has several years of experience with Lustre, GPFS, and other filesystems due to our GUPFS and NGF effort.
Specialized chip architecture requires compiler tuning not relevant to mainstream products.	ANL and ORNL have expertise here
Workload management software immature for providing high utilization and throughput under a workload with a wide range of concurrencies.	NERSC implemented the first highly parallel checkpoint/restart, and was the first site to use it on IBM systems at scale, as well as implementing excellent workload management methods. This experience, combined with LBNL’s Berkeley Lab C/R (Hargrove and Duell), makes LBNL/NERSC the best place to address this.
Memory available per MPI process is low compared with other architectures.	This is basically a programming and testing effort.
Tools for large scale debugging, analyzing, and tuning are lacking.	Large-scale debugging has to be rethought from scratch. Totalview is a very expensive monopoly that is not really effective. ANL, ORNL, and LBNL have expertise here.

2.3. IBM Power

Issue	**Mitigation**
Power5 architecture has high power, cooling, and space requirements compared with other systems. Power6 will probably be an improvement, but will likely lag behind other systems.	This is hardware design and not really addressable in this time frame. ORNL and LBNL have expertise designing advanced buildings for computers.
IBM Federation interconnect will be at the end of its lifetime in this timeframe. Its fat-tree topology also has higher costs than networks with a 3D torus connectivity. The latency of the Federation interconnect will not be any better than commodity in this timeframe.	Not much to do here.
AIX is known to be a heavyweight operating system both in terms of memory and cpu cycles consumed.	ANL’s OS experience might help to implement a lightweight AIX.

Intrinsic barriers to application scaling exist in the AIX operating system, such as system daemons, lack of synchronization in OS images.	LBNL, ANL, and ORNL all have expertise that could help with implementing CNL. NERSC has expertise in identifying and dealing with memory and CPU intrusion of operating systems.
Tools for large scale debugging, analyzing and tuning are lacking.	Large scale debugging has to be rethought from scratch. Totalview is a very expensive monopoly that is not really effective. ANL, ORNL, and LBNL have expertise here

2.4. General Issues

Issue	**Mitigation**
System administration and resource management.	NERSC and ORNL have experience in making large systems work with effective system administration.
High performance MPI collective operations and support for overlapped computation and communication activity.	ANL would be a national place to work on this. LBNL's BIPS also has expertise that could be applied.
Security: None of the systems provide anything new in security beyond standard Linux/UNIX. AIX has some advanced features, but some are not compatible with open standards (e.g., OpenLDAP). BG/L and the XT3 have given little effort to security.	LBNL and NERSC have expertise in both providing open access to systems without advanced security features, as well as working to implement new features.
Advanced languages: Cray and IBM are interested in UPC but not willing to completely support it.	LBNL has extensive expertise in UPC.

3. What Are the Local Strengths at the Labs?

3.1. LBNL Strengths

3.1.1. Large-Scale System Management

Nick Cardo, Tina Butler, Jim Craw, Bill Kramer: NERSC has demonstrated the ability manage large-scale systems to achieve a variety of goals, from high utilization to high levels of time going to large jobs. NERSC has a history of fielding early production systems and making them highly effective for science.

3.1.2. Large-Scale Data Management

Greg Butler, Brent Draney, Cary Whitney, Will Baird, Damian Hazen, Howard Walter: NERSC is one of the development sites for HPSS and operates one of the largest data repositories. NERSC is also the first site to implement a facility-wide global filesystem across multiple architectures. This is complemented by NERSC's ability for end-to-end network tuning.

3.1.3. Large-Scale Application Improvement

Jonathan Carter, David Skinner, Richard Gerber, Francesca Verdier: NERSC's expertise in scalability and optimization has enabled a number of applications to reach and exceed the 1,000-processor scale.

3.1.4. Languages

Kathy Yelick, Parry Husbands, Costin Iancu: UPC and Titanium language development. Evaluation of advanced HPC languages (with Rusty Lusk at ANL) as part of the High Performance Languages effort.

3.1.5. Analytics and Data Management

Wes Bethel: Leads visualization and analytics for NERSC and LBNL.

Arie Shoshani: Leads the SciDAC Scientific Data Management Center and focuses on data management for the scientific community running on the largest-scale computing systems.

3.1.6. Interconnect Architecture and Software

Jason Duell, Michael Welcome, Parry Husbands, etc.: GASNet. Experience with developing efficient runtime layers for single-sided messaging for UPC on a wide array of systems. Development of very efficient runtime layer and software for one-sided communication.

David Skinner, Leonid Oliker, John Shalf, Ali Pinar: Analysis of HPC interconnect performance on large-scale systems using Skinner's IPM. HFAST interconnect architecture.

3.1.7. Libraries

Esmond Ng, Sherry Li, Osni Marques, Tony Drummond, James Demmel, Kathy Yelick: Sparse linear algebra. OSKI/SPARSITY auto-tuners. SuperLU.

Osni Marques, Tony Drummond: The DOE ACTS Collection — a collection of robust and scalable software libraries and toolkits that implement a variety of numerical algorithms, and facilitate code development, tuning and portability.

3.1.8. Algorithms

John Bell, Phil Colella: Adaptive multiscale methods (AMR). Chombo.

Sherry Li, Osni Marques, Juan Meza, Esmond Ng, Ali Pinar, and Chao Yang: High performance algorithms on sparse matrices, combinatorial problems, and optimization.

Andrew Canning, Lin-Wang Wang: Materials science algorithms and applications. PARATEC, PeSCAN (density functional theory). Efficient parallel 3D FFTs.

Julian Borrill and Peter Nugent: Algorithms in the area of cosmology.

Chris Ding, Ali Pinar: Algorithms in data mining, graph theory, and bioinformatics.

3.1.9. Benchmarking, Performance Modeling, and Analysis

Leonid Oliker: Benchmarking and analysis of ultrascale applications on the largest systems in the world. Broad application experience and contacts with many software groups.

David Skinner: Performance analysis tools for parallel applications (IPM and related tools for visualizing performance data).

Costin Iancu: Communications analysis for UPC and global memory languages.

Erich Strohmaier: Benchmarking and performance modeling for large-scale systems. (APEX Map).

David Bailey: NAS Parallel Benchmarks, leader of Performance Engineering Research Center (PERC).

Bill Kramer: The Sustained System Performance (SSP) and Effective System Performance (ESP) Tests.

Greg Butler, Rei Lee, David Skinner, Hongzhang Shan : I/O benchmarks for large scale I/O.

3.1.10. Operating Systems

Paul Hargrove, Jason Duell, Tom Davis: Linux checkpoint/restart. MVIA, FastOS.

3.1.11. Computer Architecture

Kathy Yelick, Leonid Oliker, John Shalf: Collaborations with David Patterson and Krste Asanovic at UC Berkeley and with Christos Kozyrakis at Stanford. Research into opportunities for emerging architectures such as Cell and ViVA. Development of new architectures such as IRAM and VIRAM (processor in memory architecture).

3.1.12. Distributed and Grid Computing

Deb Agarwal, Bill Johnston, Keith Jackson, Steve Chan, Bill Kramer: LBNL is a leader in grid computing and currently is supporting a dozen virtual organizations and grids in production.

3.1.13. Network Research

Deb Agarwal, Bill Johnston, Mary Thompson, Howard Walter, Brent Draney: LBNL is the home of ESnet, an organization that takes a global, end-to-end view of networking requirements. LBNL’s history in network research goes back to the early days of the Internet. It currently focuses on understanding network bottlenecks.

3.1.14. CyberSecurity

Vern Paxson, Brent Draney, Scott Campbell, Nick Cardo, Howard Walter, Bill Kramer: Bro is a flexible and proven instruction detection system. NERSC has been a leader in large-scale cyber security, both with its track record, and through conferences, IT system administration practices, tutorials, and improving software.

3.1.15. Computer Science

Strong connections to computer science and computational science disciplines. There include reliable and adaptive distributed systems (Dave Patterson), sensor networks and reconfigurable systems (David Culler), math algorithms (James Demmel), Center for Information Technology Research in the Interest of Society (CITRIS) (James Demmel), storage (John Kubitowiz), theory, security (David Wagner), etc.

4. Who Are Other Key Players Who Will Play a Role and Can Help?

4.1. Argonne National Laboratory

- Experience in MPI, might be best suited to address the high interconnect latency.
- Experience in operating systems that could be lightweight OS for these systems.
- Large-scale debugging.
- BG/L and fault-tolerant software: Pete Beckman, Susan Coughlan.

4.2. Lawrence Livermore National Laboratory

- Experience in MPI, might be best suited to address the high interconnect latency.
- Experience in operating systems that could be lightweight OS for these systems.
- Large-scale debugging.
- Large-scale visualization and data analysis: Hank Childs's VisIT software infrastructure is an important substrate for visualization of massive datasets. Valerio Pascucci's space filling curves and data streaming technology is essential for reorganizing data for efficient analysis and retrieval.
- Large-scale system assessment: Mark Seager has a long track record of identifying the weaknesses in vendor roadmaps and architectural plans for the largest-scale ASC systems. LLNL contributes valuable experience on risk assessment and mitigation for complex systems at all levels of the effort.
- Experience in unstructured mesh discretization and refinements, multigrid algorithms for solving linear systems, and algorithms for nonlinear equations: Lori Freitag Diachin, SciDAC Terascale Simulation Tools and Technologies (TSTT) center.
- Experience in high performance modeling and simulations in large scale applications.

4.3. Oak Ridge National Laboratory

- Benchmarking and performance analysis: Patrick Worley and Jeff Vetter have a long track record of benchmark and performance analysis of key DOE applications on large-scale systems.
- Benchmarking tools and technology: Jeff Vetter has a long history in the field of developing the tools necessary for assessing massively parallel computing systems and applying the technologies to real-world problems.
- Early systems evaluation: Worley, Vetter, Tom Dunigan.
- Climate model: John Drake, Pat Worley.
- Scalable System Software (SciDAC): Al Geist.

4.4. Pacific Northwest National Laboratory

- Global arrays / ARMCI

4.5. Sandia National Laboratory

- Low-level communication interfaces – PORTALS

- Computer system design and engineering: Bill Camp and Jim Thompson.
- High performance CFD (and combustion): John Shadid, Jackie Chen.
- High performance combinatorial algorithms: Bruce Hendrickson.
- Optimization: Bill Hart, Paul Boggs, and others.
- Numerical linear algebra: Rich Lehoucq.
- High performance computing in engineering applications (e.g., Salinas project for structural dynamics, which won a Gordon Bell Prize).

4.6. Information Sciences Institute

- Fault tolerance and scalable system software: Bob Lucas.
- Advanced compiler technology: Mary Hall, Bob Lucas.

5. Identifying Requirements Using Scalable Performance Monitoring

Modern parallel systems are highly complex, and using them will continue to be significantly challenging. Systems approaching the petascale will present even more challenges in all areas of software from operating systems to storage to applications. One brief example is described below, based on performance analysis work at NERSC.

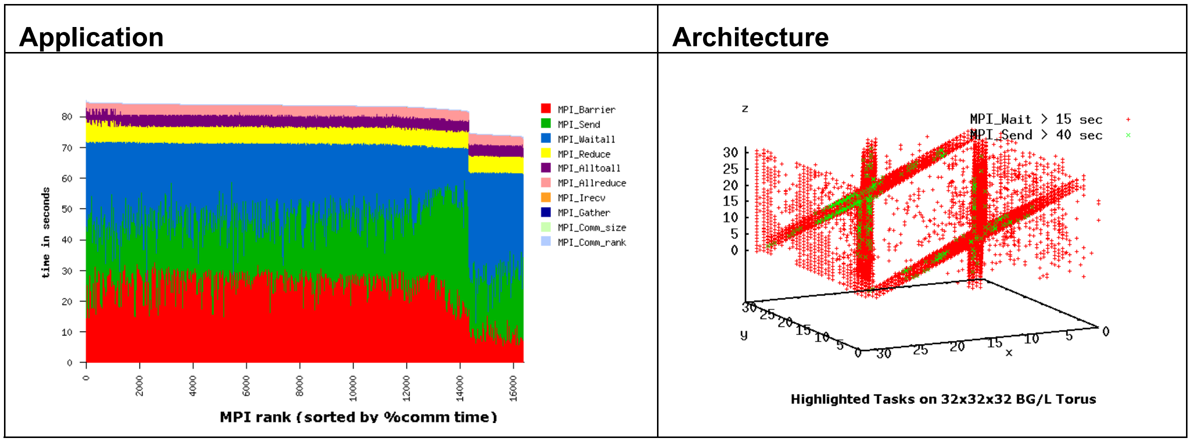
Modern parallel computers consist of unprecedented numbers of components and subsystems. At such scales, both the complexity and consequence of performance shortfalls is magnified as a performance issue, when a single component out of tens of thousands is sufficient to erode the performance and therefore delivered value of the entire system.

Massive concurrency brings with it two significant information management problems for application performance analysis whose solution will require new software:

1. The volume of performance data generated requires specialized low overhead techniques for the collection and aggregation of performance data. A profiling method that adds even modest overhead will appear to a 10K-way application to be a nearly continuous interruption of the user code. Careful attention must be paid to how aggregation and reporting are done.
2. The analysis, either offline or dynamic, of performance data becomes a significant problem for performance tools and their users. In order to realize performance gains, the analysis of the data must be scalable as well. Automated optimization techniques help lessen the overload of data to the user by providing a higher-level performance analysis context.

Without performance monitoring infrastructure designed to be resilient at these scales, the value of petascale systems, in terms of scientific output realized, will be significantly diminished. NERSC has developed an infrastructure for non-invasively gathering performance data from a workload. Such an infrastructure is useful in gaining insight into application performance and improvement, but also for system software improvements.

The starting point for performance analysis is performance measurement. NERSC has developed a scalable application profiling layer, IPM, which is useful at larges scales and has been demonstrated to scale well beyond the concurrency regime of current NERSC machines. IPM has demonstrated scalability on BG/L systems and been effective at identifying performance issues on 32K tasks as shown in Figure 1.


Application

Architecture

Figure 1. Performance is more complex than a single number. Examining how performance varies within a parallel application (in MPI rank space) or within the architectural space of the torus reveals important performance characteristics and makes bottlenecks to performance clear. In application space, we see that most of the tasks spend more time communicating than a particular a small fraction of tasks. This performance discontinuity is a likely indication of load imbalance. By moving to the architectural space of machine coordinates within the torus, the bottlenecks that make up the discontinuity become clear. Given the massive concurrency involved, care must be taken to retrieve and render the relevant data but not all performance data.

Obtaining insight into the behavior of a parallel application or a system in this regime is challenging. Having recognized that massive concurrency is a likely direction for HPC, NERSC is actively involved in the examination and solution of the problems that come with it. Some of the trends in application and architecture of petascale systems make obvious the technology gaps that exist:

- Trends toward using adaptive approaches such as adaptive mesh refinement to address performance problems at large scale serve to exacerbate the problem of obtaining performance insight by increasing the amount of time-varying behavior, including significant yet elusive load imbalance problems. Gunter is using IMP to get "regional" snapshots of communication patterns. [More information can be developed if needed.]
- Mesh and torus interconnects heighten the need for optimization not only of software and algorithms but also of the placement of the parallel tasks in a parallel computer. This presents an extra burden on the users of such a system, as they must not only optimize their code but also seek optimal task placement geometries. The need to automate this optimization is largely unaddressed on current large-scale 3D torus architectures. On BG/L, task placement is an open problem for many codes that do not have obvious embeddings in a 3D grid. This software gap means that such machines are less useful to wide classes of applications. NERSC is active in pursuing this issue through both its Science-Driven System Architecture team and through long-range research into interconnect alternatives.
- As the number of factors influencing application performance expands, the need to be able to take frequent low overhead performance snapshots also expands. The huge number of conditions and components that can lead to performance degradation is best managed by having an easily accessed body of data on the performance health of the machine. NERSC makes heavy use of regular performance monitoring of HPC resources. Even at today's smaller scales, on multiple occasions such performance health maintenance has revealed performance deficiencies that would otherwise have gone unresolved.

6. Bellwether Ultrascale Applications

This section outlines a set of codes that provide good coverage application requirements for the DOE scientific community, both in terms of application areas and key numerical algorithms. If the application codes outlined in Table 1 are able to run scalably and successfully on petaflops precursors such as the XT3 and BG/L, then there is an opportunity to apply the lessons learned to the broader DOE community and thereby ensure the success of peta scale systems when they emerge. More information about these applications can be found in papers written by Leonid Oliker et al. (http://crd.lbl.gov/~oliker).

Table 1. Scientific application suite for evaluation study representing a broad spectrum of numerical methods

Name	Lines of Code	Discipline	Problem and Method	Structure
MADCAP	5000	Cosmology	CMB analysis via Newton-Raphson using ScaLAPACK	Dense matrix
Cactus	84,000	Astrophysics	Einstein's theory of General Relativity via finite differencing	Structured grid
LBMHD	1,500	Plasma Physics	Magnetohydrodynamics via lattice Boltzmann	Lattice
FVCAM	200,000+	Climate Modeling	Atmospheric circulation via finite volume	Structured grid
GTC	5,000	Magnetic Fusion	Vlasov-Poisson equation via particle in cell	Particle/grid (PIC)
SuperLU	42,000	Linear Algebra	Sparse iterative solver using LU decomposition	Sparse matrix
PMEMD	37,000	Life Sciences	Molecular dynamics via particle mesh Ewald	Particle
PARATEC	50,000	Material Science	Material science density functional theory via FFT	Spectral/3D-FFT + dense matrix
SuperNova	200,000+	Combustion	Rayleigh-Taylor via adaptive mesh refinement	SAMR
GAMESS	500,000	Chemistry	Density functional theory	
MILC		Quantum Chromodynamics	Conjugate gradient algorithm for physics on a 4D lattice	Sparse

6.1. Astrophysics: Cactus

One of the most challenging problems in astrophysics is the numerical solution of Einstein's equations following from the Theory of General Relativity (GR): a set of coupled nonlinear hyperbolic and elliptic equations containing thousands of terms when fully expanded. The Cactus Computational ToolKit (CCTK) is designed to evolve Einstein's equations stably in 3D on supercomputers to simulate astrophysical phenomena with high gravitational fluxes, such as the collision of two black holes and the gravitational waves radiating from that event. The standard MPI driver for Cactus solves the PDE on a local grid section and then updates the values at the ghost zones by exchanging data on the faces of its topological neighbors in the domain decomposition. The update pattern employed by Cactus is typical of a broad class of PDE solvers that use stencils on structured grids to iterate towards a solution. Therefore, the lessons learned from optimizing Cactus for the XT3 and BG/L are entirely transferable to many important codes in the DOE community.

6.2. Plasma Physics: LBMHD

Lattice Boltzmann methods (LBM) have proved a good alternative to conventional numerical approaches for simulating fluid flows and modeling physics in fluids. The basic idea of the LBM is to develop a simplified kinetic model that incorporates the essential physics and reproduces correct macroscopic averaged properties. Recently, several groups have applied the LBM to the problem of magneto-hydrodynamics (MHD) with promising results. LBMHD simulates the behavior of a two-dimensional conducting fluid evolving from simple initial conditions and decaying to form current sheets. The computational structure of LBMHD uses a 2D spatial grid coupled to an octagonal streaming lattice and block distributed over a 2D processor grid.

Due to the relatively simplicity of LBMHD's computational structure, combined with the significant disparity between scalar and vector performance demonstrated in our work, this code has been chosen as an application benchmark for the DOD HPCS evaluation effort. We plan to continue working closely with the HPCS community in generating full-scale benchmark applications and disseminating our findings to the HPC community at large.

6.3. Magnetic Fusion: GTC

The Gyrokinetic Toroidal Code (GTC) is a 3D particle-in-cell (PIC) application developed at the Princeton Plasma Physics Laboratory to study turbulent transport in magnetic confinement fusion. GTC is currently the flagship SciDAC fusion microturbulence code. Turbulence is believed to be the main mechanism by which energy and particles are transported away from the hot plasma core in fusion experiments with magnetic toroidal devices. An in-depth understanding of this process is of utmost importance for the design of future experiments since their performance and operation costs are directly linked to energy losses. GTC simulations of plasma microturbulence in a magnetic fusion device show elongated finger-like structures are turbulent eddies in the electrostatic potential that act as energy and particle transport channels. This type of calculation helped to shed light on anomalous energy transport that was observed in real experiments.

This code could potentially be scaled up to very large numbers of processors. We therefore plan to conduct an extensive analysis of the tradeoffs between using a very high level of coarse-grained parallelism (as found in BG/L) and utilizing a relatively small number of processors that leverage fine-grained vector parallelism. A new data-decomposition scheme for GTC may, for the first time, enable a breakthrough of the teraflop barrier and prepare the way for GTC simulations on near-petaflop platforms like the BG/L and larger XT3 systems. We are working to help establish GTC as an HPCS application benchmark due to the importance of this SciDAC fusion application and its potential to utilize ultra-scale computational resources.

6.4. Material Science: PARATEC

PARATEC (PARAllel Total Energy Code) performs *ab initio* quantum-mechanical total energy calculations using pseudopotentials and a plane wave basis set. The pseudopotentials are of the standard norm-conserving variety. Forces can be easily calculated and used to relax the atoms into their equilibrium positions. PARATEC uses an all-band conjugate gradient (CG) approach to solve the Kohn-Sham equations of density functional theory (DFT) and obtain the ground-state electron wavefunctions. PARATEC simulations are used to better understand nuclear magnetic resonance experiments. In solving the Kohn-Sham equations using a plane wave basis, part of the calculation is carried out in real space and the remainder in Fourier space using specialized parallel 3D FFTs to transform the wavefunctions. Due to PARATEC's global communication requirements, architectures with a poor balance between their bisection bandwidth and computational rate will suffer performance degradation at higher concurrencies.

PARATEC is an extremely useful tool in evaluating the balance between computational resources and global interprocessor communication facilities, and will be a critical component of our study in evaluating the tradeoffs of evolving interconnect designs. Furthermore, a recent survey of NERSC ERCAP requests for materials science applications showed that DFT codes similar to PARATEC accounted for nearly 80%

of all HPC cycles delivered to the materials science community. Therefore, PARATEC is an excellent proxy to the application requirements of the entire materials science community.

6.5. Cosmology: MADCAP

The Cosmic Microwave Background (CMB) is a snapshot of the Universe some 400,000 years after the Big Bang. The pattern of anisotropies in the CMB carries a wealth of information about the fundamental parameters of cosmology. Realizing the extraordinary scientific potential of the CMB requires making precise measurements of the microwave sky temperature over a significant fraction of the sky at very high resolution. Such measurements are made by scanning the sky for as long as possible with a cryogenically cooled telescope and as many microwave detectors as possible. The reduction of the resulting datasets — first to a pixelized sky map, and then to an angular power spectrum — is a serious computational challenge, and one which is only getting worse with increasing dataset sizes, as we try to make ever more precise measurements. It is therefore critical to choose the optimal algorithmic approach and supercomputing platform; one approach is the Microwave Anisotropy Dataset Computational Analysis Package (MADCAP), which has been widely used on a variety of HPC platforms.

The full MADCAP spectral estimator includes a large number of special-case features, from preliminary data checking to marginalization over foreground templates, which dramatically increase the size and complexity of the code without altering its basic operational structure. For simplicity, we are therefore developing a stripped-down version, called MADbench, a lightweight version of MADCAP expressly designed for benchmarking, which retains the operational complexity and integrated system requirements of the full application. We plan to publicly distribute MADbench, and use our synthetic benchmark generation methodology as a model for creating full-scale portable codes for community-wide evaluation efforts. Additionally, we are in the process of distributing MADbench to the HPCS community, as it combines the potential for ultra-scale parallelism with the challenges of heavy I/O requirements. Several other DOE-supported groups including the FastOS research community have shown interest in MADbench, and we plan to work closely together to support their research efforts. Finally, utilizing MADbench on the latest generation of HPC architectures will allow us to understand the system balances of various platforms, especially in the context of I/O — a critical issue for future data-intensive analyses.

6.6. Climate Modeling: FVCAM

The Community Atmosphere Model (CAM) is the atmospheric component of the flagship Community Climate System Model (CCSM3.0). Developed at the National Center for Atmospheric Research (NCAR), the CCSM3.0 is extensively used to study climate change. The CAM application is an atmospheric general circulation model (AGCM) and can be run either coupled within CCSM3.0 or in a standalone mode driven by prescribed ocean temperatures and sea ice coverage. AGCMs are key tools for weather prediction and climate research. They also require large computing resources: even the largest current supercomputers cannot keep pace with the desired increases in the resolution of these models.

Lenny Oliker’s group at LBNL will continue the investigation of FVCAM across a broad spectrum of computing platform using very large grid configurations and thousands of processors, with the goal of isolating the performance-limiting architectural features. Due to the limited number of coarse-grained domain decompositions, FVCAM represents an example of a code that may not benefit from ultra-parallel systems, but must instead utilize fine-grained parallelism to extract the necessary degree of concurrency to scale up to large systems. Our study will focus on understanding these tradeoffs for the latest HPC platforms.

6.7. Combustion: AMR

A Type Ia supernova explosion likely begins as a nuclear runaway near the center of a carbon-oxygen white dwarf. The outward propagating flame is unstable to the Landau-Darrieus, Rayleigh-Taylor, and Kelvin-Helmholtz instabilities, which serve to accelerate the flame to a large fraction of the speed of sound. At present, the exact mechanism for the subsequent explosion of the star is unknown, and investigators have turned to numerical simulation. The SuperNova code, developed by researchers in the Center for

Computational Sciences and Engineering at LBNL, is one such code that models the unstable flow processes in reacting, degenerate nuclear matter that occurs in white dwarf stars. SuperNova utilizes adaptive mesh refinement (AMR), a technique for automatically refining regions of the physical domain, concentrating computational resources where interesting physics is occurring.

Massive parallelism exacerbates longstanding problems with balancing the need for communication locality with memory balance and load-balancing requirements for AMR codes. Measuring and analyzing these tradeoffs is a critical step in understanding the potential of effectively using massively parallel architectures for future AMR computations. It is commonly believed that adaptive calculations will become a key component of future high-fidelity multi-scale simulations across of broad spectrum of application domains. However, to date, no AMR performance results at the enormous concurrencies offered at the massive degree of parallelism of sub-petascale systems are currently available to the scientific community. The lessons learned from running AMR codes on such systems will guide system requirements for codes that specify more intelligent and targeted use of HPC resources rather than resorting to brute computational force to meet the needs of multi-scale scientific problems.

6.8. Linear Algebra: SuperLU

An emerging method for scientific computation is the use of sparse matrix methods. We plan to explore this important class of algorithms. Our current candidate is SuperLU, a general-purpose library for the direct solution of large, sparse, nonsymmetric systems of linear equations on high performance machines. The library routines perform an LU decomposition with partial pivoting and triangular system solves through forward and back substitution. The LU factorization routines can handle non-square matrices, but the triangular solves are performed only for square matrices. The matrix columns may be preordered (before factorization) either through library- or user-supplied routines. This preordering for sparsity is completely separate from the factorization. Working precision iterative refinement subroutines are provided for improved backward stability. Routines are also provided to equilibrate the system, estimate the condition number, calculate the relative backward error, and estimate error bounds for the refined solutions. The irregular data access patterns of this code present a performance challenge for superscalar systems, while the complexity of the control flow is expected to inhibit scalability and exacerbate load imbalances on massively parallel systems such as BG/L and the XT3. This class of codes is at odds with traditional cache-based architectures, and will allow us to evaluate the tradeoffs between various classes of memory hierarchies. Like AMR, SuperLU offers insight into the requirements of applications that are able to attack larger problems using elegance rather than brute force. The requirements for such applications must be considered as essential for petascale platforms — lest we end up with overly specialized hardware architectures that force us into brute-force numerical approaches.

6.9. Life Sciences: PMEMD

We are interested in investigating algorithms in the field of molecular dynamics, due to their increasing importance and computational irregularity. One candidate code is PMEMD. PMEMD an application that performs molecular dynamics (MD) simulations and minimizations using particle mesh Ewald molecular dynamics. The force evaluation is performed in an efficiently parallel manner using state-of-the-art numerical and communication methodologies. PMEMD uses a highly asynchronous approach to communication for the purposes of achieving a high degree of parallelism. Variants on the parallel particle mesh calculations in PMEMD are used in several other MD codes. The dynamic nature of this code inhibits data-parallelism and will present a significant challenge for vector architectures. This application may therefore benefit more from ultra-parallel systems than architectures that depend on fine-grained chip-level parallelism. Our study will investigate these competing tradeoffs.

6.10. Chemistry: GAMESS

GAMESS[14] is a program for *ab initio* molecular quantum chemistry. Briefly, GAMESS can compute SCF wavefunctions ranging from RHF, ROHF, UHF, GVB, and MCSCF. Correlation corrections to these SCF wavefunctions include configuration interaction, second order perturbation theory, and coupled-cluster approaches, as well as the density functional theory approximation. Nuclear gradients are available for automatic geometry optimization, transition state searches, or reaction path following. Computation of the energy Hessian permits prediction of vibrational frequencies, with IR or Raman intensities. Solvent effects may be modeled by the discrete effective fragment potentials or continuum models such as the polarizable continuum model. Numerous relativistic computations are available, including third-order Douglas-Kroll scalar corrections and various spin-orbit coupling options. The fragment molecular orbital method permits many of these sophisticated treatments to be used on very large systems by dividing the computation into small fragments.

A variety of molecular properties, ranging from simple dipole moments to frequency-dependent hyperpolarizabilities, may be computed. Many basis sets are stored internally, together with effective core potentials, so all elements up to radon may be included in molecules. Most computations can be performed using direct techniques, or in parallel on appropriate hardware. Graphics programs, particularly the MacMolPlt program for the Macintosh, are available for viewing of the final results.

6.11. QCD: MILC

The MILC code is a set of codes developed by the MIMD Lattice Computation (MILC) collaboration for doing simulations of four dimensional SU(3) lattice gauge theory on MIMD parallel machines. The latest version of this code includes libraries and routines for SU(2) gauge theory as well. The MILC Code is publicly available for research purposes.

[14] "General Atomic and Molecular Electronic Structure System," M. W. Schmidt, K. K. Baldridge, J. A. Boatz, S. T. Elbert, M. S. Gordon, J. H. Jensen, S. Koseki, N. Matsunaga, K. A. Nguyen, S. Su, T. L. Windus, M. Dupuis, J. A. Montgomery, *J. Comput. Chem.*, **14**, 1347–1363 (1993).

7. Software Requirements for Petascale Production Systems

This section provides a detailed enumeration of system software requirements that NERSC requires of its production systems. The requirements below evolved from the NERSC-3, -4, and -5 contracts and should be considered an almost minimal set of requirements to have systems work well for a diverse, capability workload. DOE could use these requirements as formatted below, along with benchmark results for the codes in section 6, to assess systems from vendors or large-scale computing sites.

For each requirement below, a rating could be given in brackets for the Cray XTE, IBM BlueGene/L, and IBM Power SP. For example, [Yes, Maybe, No] indicates our understanding that the Cray XT3 does meet this requirement, the IBM BlueGene/L may meet it, and the IBM Power SP does not meet it. Possible means the vendor is currently considering whether to support the requirement.

7.1. Supported Functionality

All publicly-available APIs provided with this system will have accurate and complete documentation. In particular, any APIs or features of APIs that are not to be used by the site will be explicitly documented as such.

7.2. Operating System Software

7.2.1. 64-Bit Address Space Software Support [______, ______, ______]

The operating system shall support a 64-bit address space for both privileged and user level applications.

7.2.2. The operating system and necessary system software, including subsequent releases, shall reside within no more than 5% of the memory on a node. [______, ______, ______]

The Vendor's supplied operating system including all associated system functions and services shall reside within 5% of physical memory. This limit includes the kernel, interconnect overhead, MPI overhead, workload manager, job launcher, RAM disk, and any system monitoring.

7.2.3. Aggregate User Accessible Memory [______, ______, ______]

Aggregate user accessible memory shall be determined via the following formula:

$$\sum(memory_i)*.9$$

where $memory_i$ represents the memory on a single node.

Aggregate user accessible memory shall be accessible via an unprivileged parallel user application.

7.2.4. Available Compute Cycles [______, ______, ______]

The operating system, subsequent releases, and necessary system software shall consume ≤ 1% of all available compute cycles.

7.2.5. Process Accounting [______, ______, ______]

The operating system shall have the ability to record usage of processes that execute on the system. Accounting records from each source contain the following usage information:

- Command name

- Path to command
- Start timestamp
- End timestamp
- User CPU usage
- System CPU usage
- User ID
- Group ID
- Account ID
- Job/session ID
- Batch job identifier
- TTY name
- Memory usage
- Characters read/write
- Blocks read/write
- Process exit status

7.2.6. Process Limits [______, ______, ______]

Standard UNIX/Linux process limits as supported by a standard distribution of the operating system shall be supported. These include, but are not limited to, hard and soft limits for:

- Time (CPU-seconds)
- File (blocks)
- Core dump (blocks)
- Data (kilobytes)
- Locked memory (kilobytes)
- Memory (kilobytes)
- Number of open files (descriptors)
- Process limit

7.2.7. Co-Scheduling [______, ______, ______]

The operating system shall support the "co-scheduling" functionality. Co-scheduling is the ability to align system processes to be executed nearly simultaneously. This minimizes the impact of system processes on the system.

This functionality further spans to allow parallel jobs on multiple nodes to be co-scheduled on the processors nearly simultaneously.

7.2.8. Operating System Security [______, ______, ______]

File Access: Discretionary access control mechanism restricting file access based on ownership and permissions shall be supported on all filesystems.

Access Control Lists: ACLs shall be supported on all filesystems.

Passwords: Encrypted passwords shall be stored in a secure manner (i.e., only readable by privileged users).

Privileges: A facility shall exist that allows only a subset of “super-user” privileges to be granted to a specific user.

7.2.9. Login Information [______, ______, ______]

Users shall be notified upon successful login of the following information:

- Date and time of last successful login
- Date and time of last unsuccessful login (if appropriate)
- Total number of unsuccessful logins since their last successful login (if appropriate)

7.2.10. Secure Shell [______, ______, ______]

The Vendor shall provide and support the current and new versions of OpenSSH.

7.2.11. Secure Socket Layer [______, ______, ______]

The Vendor shall provide and support the current and new versions of OpenSSL.

7.2.12. Lightweight Directory Access Protocol [______, ______, ______]

The Vendor shall provide and support the current and new versions of OpenLDAP.

7.2.13. Pluggable Authentication Modules [______, ______, ______]

The Vendor shall provide and support PAM. This includes a generalized API that provides authentication services with which new authentication methods can be written. Authentication policies can be modified by editing configuration files.

7.2.14. Linux and/or UNIX Operating Systems [______, ______, ______]

Standards supported by the Vendor:

- IEEE 1003.1-1996 [POSIX System Interfaces]
- IEEE 1003.2-1992 [Shell and Utilities]
- X/Open PG4
- Open Group UNIX95 and UNIX98 branding

In areas where the operating system is not compliant with the standards above, the Vendor shall clearly identify lack of compliance.

7.3. Inter-Node Communication Network

7.3.1. High Availability Configuration [______, ______, ______]

The Interconnect shall be run and managed in a highly available configuration. The Interconnect shall support the capabilities necessary for running and managing a highly available interconnect configuration.

Configuration tables are used for data flow mapping across the interconnection network. They shall be reloaded in the event of a hardware failure, thus providing a means to reconfigure the interconnection network around hardware failures. Routing tables shall be reconfigured and dynamic so a reconfiguration does not require a full system boot.

The system can survive partial interconnect failure with the loss of only a single node.

7.3.1.1. Dynamic Route Management [______, ______, ______]

The Interconnect topology shall be dynamically reconfigurable. All links/planes shall be capable of being dynamically removed from or joined with the interconnect fabric. Tools shall exist to identify the end points of any given link.

7.3.1.2. Automatic Dynamic Rerouting [______, ______, ______]

The Interconnect fabric shall be capable of automatically reconfiguring routes to optimize around failed components. Upon completion of any repair actions, the Interconnect shall detect the repaired component and perform optimization of routing to utilize the returned hardware.

7.3.1.3. Dynamic Hardware Failure Recovery [______, ______, ______]

All components of the Interconnect fabric shall be capable of being dynamically removed or joined to the Interconnect fabric without restarting the entire Interconnect.

7.3.1.4. Dynamic Congestion Management [______, ______, ______]

The Interconnect fabric shall be capable of detecting congested links and dynamically reroute traffic via an alternate path.

7.3.1.5. Node Dynamic Switch Management [______, ______, ______]

The Vendor's system shall support the capability of dynamically removing or dynamically adding a node to the Interconnect fabric.

7.3.1.6. Online Interconnect Diagnostics [______, ______, ______]

The Interconnection shall support the capability of having diagnostics performed while running production workloads. These diagnostics shall be capable of, but not limited to, identifying failed interconnect links, identifying degraded links, and topology verification.

7.3.1.7. Online Adaptor Diagnostics [______, ______, ______]

The Vendor's system shall be capable of performing online diagnostics against the interconnect adapter. These diagnostics shall be capable of, but not limited to, detecting failed adapters, detecting adapters running at degraded performance levels, and point-to-point topology verification.

7.3.1.8. Link Failure [______, ______, ______]

No link failure shall result in a total failure of the interconnect.

7.3.1.9. Router Failure [______, ______, ______]

No router failure shall result in a total failure of the interconnect.

7.4. Workload Management

7.4.1. Batch System

7.4.1.1. Modifiable Limits and Resources [______, ______, ______]

A utility shall exist that can modify the requested limits and resources for a batch job.

7.4.1.2. Queue Reallocation [______, ______, ______]

A utility shall exist that can reallocate a batch job to a new batch queue.

7.4.1.3. SMP Scheduling [______, ______, ______]

The batch system shall perform SMP scheduling, the ability to schedule multiple tasks per node.

7.4.1.4. Packed/Unpacked Scheduling [______, ______, ______]

The batch system shall support the capability to specify that a node be dedicated to a single job or shared with multiple jobs.

7.4.1.5. Multiple Job Queues [______, ______, ______]

The batch system shall be capable of organizing jobs by job queues. Each queue can have different priorities and limits.

7.4.1.6. Queue Complexes [______, ______, ______]

The batch system shall be capable of organizing a set of queues into a queue complex. All limits shall be applicable to a queue complex and shall govern the set of queues within the complex.

Characteristics and priorities may be assigned to complexes and to each queue within a complex independently.

7.4.1.7. User Limits [______, ______, ______]

The batch system shall be capable of setting and enforcing per-job user limits. These limits shall be dynamically configurable and enforceable for queues, jobs, users, and groups.

7.4.1.8. Batch Job Aging [______, ______, ______]

The batch system shall be capable of being dynamically configured to enforce an aging algorithm that takes into account the time the job was queued for the computation of job priority.

7.4.1.9. Scheduler Statistics [______, ______, ______]

The batch systems scheduler shall be capable of producing detailed statistics for analysis of scheduler performance.

7.4.1.10. Scheduler Hints [______, ______, ______]

The batch systems scheduler shall be capable of providing an informative message as to why a job has not been chosen to run. Such examples include but are not limited to a batch queue's run limit being reached.

7.4.1.11. Job Routing [______, ______, ______]

The batch systems scheduler shall be capable of routing jobs to the appropriate queue based on the jobs' requirements. The queues need not be on the same system.

7.4.1.12. Queue Limits [______, ______, ______]

Resource and job limits shall be enforceable at the queue level. Such limits include and are not limited to:

- CPU time
- Wallclock time
- Memory
- Running jobs
- Per user running jobs
- Per group running jobs
- Queued jobs
- Per user queued jobs
- Per group queued jobs

7.4.1.13. Job Limits [______, ______, ______]

All resource limits shall be enforceable for a single job.

7.4.1.14. User Limits [______, ______, ______]

All resource limits shall be enforceable at the user level.

7.4.1.15. Group Limits [______, ______, ______]

All resource limits shall be enforceable at the UNIX group level.

7.4.1.16. Resource Limits [______, ______, ______]

The batch system shall support resource limits. These resources shall include but are not limited to CPUs, memory, disk space, and nodes.

7.4.1.17. Consumable Resource Limits [______, ______, ______]

The batch system shall support consumable resources. These resources shall include but are not limited to CPUs, nodes, memory, licenses , disk space, and site-specific defined consumables. Consumable resources are distributed to multiple batch jobs until the resource is exhausted.

7.4.1.18. Job/User Favoring [______, ______, ______]

The batch system shall be capable of identifying a user or batch job and favoring the job or user to the head of the queue. The action shall also be capable of being undone.

7.4.1.19. Documentation [______, ______, ______]

The Vendor shall supply all documentation for the batch system. Documentation shall consist of all manuals and man pages. All documentation shall be complete and all-inclusive for the version installed.

7.4.1.20. Accounting [______, ______, ______]

The batch system shall have a comprehensive list of data recorded for each batch job. This data shall include but is not limited to:

- Multi-node resource consumption data; individual node resource usage data for each individual node used by the job
- Job ID
- Job name
- Credentials (user/group/account)
- Pathnames
- Environment variables
- Memory: Individual node, job's high water mark, average usage
- I/O rates for disk and networking
- CPU: user, system, and I/O wait time
- Wallclock time consumed
- Submit timestamp
- Dispatch timestamp
- Completion timestamp
- Hardware counters
- Events
- Job requirements

7.4.1.21. Prolog/Epilog [______, ______, ______]

The batch system shall support the capability of executing a custom written prolog and epilog. All details regarding a job's requirements and usage shall be made available to the prolog and epilog.

The batch system shall support two types of prologs/epilogs: one that will run on a single (starter) node, and one that can run on all nodes for the job.

System-wide prologs/epilogs are defined at the system level and shall apply to all jobs and users. Users many also define prologs/epilogs that shall apply to all the jobs submitted by that user. The system-wide prolog shall execute before any user-defined prolog. The system-wide epilog shall execute after any user-defined epilog.

7.4.1.22. Dynamic Configuration Updates [______, ______, ______]

All configuration options for the batch system shall be dynamically settable. The batch system shall not be required to be shut down for changes to take effect.

7.4.1.23. Workload Manager Documentation [______, ______, ______]

The workload manager's documentation shall be clearly written, concise in descriptions, as well as complete for the batch system.

7.4.1.24. Configuration Validation [______, ______, ______]

A utility shall exist with the batch system to validate the settings in the batch system's configuration files. This utility shall identify invalid options as well as syntactical errors.

7.4.1.25. Submit Filter [______, ______, ______]

The batch system shall support the capability of passing all batch jobs through a site-defined filter which can reject a job at the time of submission. All requirements and job details shall be passed to the submit filter.

7.4.1.26. High-Availability Configuration [______, ______, ______]

The batch system shall support high-availability computing environments. No single component of the batch system shall cause the batch system to not function in full production. Automatic failover of batch system components shall also be configurable and supported.

7.4.1.27. Dynamic Default Value Initialization [______, ______, ______]

All configurable settings shall be dynamically settable and initialized to values determined by the site.

7.4.1.28. Job Scheduling Based on Requirements [______, ______, ______]

The batch system shall support the capability of scheduling jobs system-wide based on requested node resources and limits. The batch system shall also be capable of scheduling jobs of varying degrees of concurrency to maximize the utilization of the compute pool.

7.4.1.29. Per-User Per-Class Limits [______, ______, ______]

The batch system shall support the capability of per-user per-class limits. Each class in the batch system shall be capable of independent user limits with respect to other job classes.

7.4.1.30. Process/Job Migration [______, ______, ______]

The batch system shall be capable of interrupting a job and relocating it to a different set of processors. The capability shall exist for any number of tasks up to the entire job.

7.4.1.31. Migration Timing [______, ______, ______]

All migrations shall complete in less than 10 minutes.

7.4.1.32. Checkpoint/Restart [______, ______, ______]

The batch system shall be capable of checkpointing a running batch job and restarting it, from the point it left off, at a later time. Restart on different nodes.

7.4.1.33. Checkpoint Time [______, ______, ______]

Checkpointing the full system running in production shall complete is less than 30 minutes.

Checkpointing a single job utilizing up to 25% of the compute pool shall complete in less than 5 minutes.

Checkpointing a single job utilizing up to 50% of the compute pool shall complete in less than 10 minutes.

Checkpointing a single job utilizing up to 75% of the compute pool shall complete in less than 15 minutes.

7.4.1.34. Restart Time [______, ______, ______]

The time to restart a checkpointed job shall be less than or equal to the time to checkpoint that job.

7.4.1.35. Advance Reservations [______, ______, ______]

The batch system shall be capable of reserving a part or the whole compute partition for a specific time for a given duration. This reservation shall support the ability to specify users that can run on the reserved nodes during the reservation. It shall also be possible to have recurring and persistent reservations.

7.4.1.36. Backfill Scheduler [______, ______, ______]

The batch system shall include a scheduler that utilizes an effective and efficient backfill algorithm with priority override capability.

7.4.1.37. Fair Share Scheduler [______, ______, ______]

The batch system shall include a scheduler that utilizes an effective and efficient fair share algorithm with priority override capability.

7.4.1.38. Batch System Gang Scheduling [______, ______, ______]

The batch system shall support the capability of allowing multiple jobs to be run on a single node. All tasks for any single job shall be scheduled simultaneously across all nodes for that job.

7.4.1.39. Batch System Job Preemption [______, ______, ______]

The batch system shall be capable of preempting a running job in order to make resources available for higher priority work.

7.4.1.40. IEEE 1003.2 Compliance [______, ______, ______]

The batch system shall be in compliance with the IEEE 1003.2 standard for batch systems.

7.4.1.41. Dynamic Scheduling Control [______, ______, ______]

The batch system shall support the ability to start the batch system without scheduling jobs, then dynamically be capable of initiating the scheduling of batch jobs.

7.4.1.42. Capability Computing [______, ______, ______]

The site shall have the ability to apply all batch system defined resources of the system to a single parallel application. This application may be composed of a mixture of baseline programming languages and baseline programming paradigms.

7.4.1.43. Node Specifications [______, ______, ______]

The batch system shall support the capability of specifying a set of nodes for the batch job to run on.

7.4.1.44. Alternate Credential Submission [______, ______, ______]

The batch system shall support the capability of submitting batch jobs with a different set of user credentials. Alternate credentials shall be specified at the time of job submission and validated. These credentials shall be used to initiate and record all job resource consumption. The batch system's accounting records shall indicate the credentials of the submitter as well as the credentials of the job.

7.4.1.45. Environment Variables [______, ______, ______]

The batch system shall support the capability of passing environment variables to the batch job to be utilized by the batch job.

7.4.1.46. Account Name [______, ______, ______]

The batch system shall support the capability of specifying an account name to charge the resource consumption against.

7.4.1.47. Grid Submissions [______, ______, ______]

The batch system shall interface to commonly used Grid software tools such as Globus, Open Grid Services Architecture, and Virtual Data Toolkit.

7.4.2. Job Launcher

7.4.2.1. Effective Job Launcher [______, ______, ______]

The Vendor shall supply and support a utility for effectively and efficiently launching batch jobs onto the computational nodes.

7.4.2.2. Launch Timing [______, ______, ______]

Launching a single job utilizing up to 25% of the compute pool shall complete in less than 15 seconds.

Launching a single job utilizing up to 50% of the compute pool shall complete in less than 30 seconds.

Launching a single job utilizing up to 75% of the compute pool shall complete in less than 45 seconds.

Launching a single job utilizing up to 100% of the compute pool shall complete in less than 60 seconds.

7.4.2.3. Administrative Control [______, ______, ______]

The Vendor's job launcher shall support the capability of being restricted to batch jobs only. This restriction shall require privileged access to change.

7.4.2.4. Capability Computing [______, ______, ______]

The site shall have the ability to launch a job across all batch system defined resources of the system to a parallel application. This application may be composed of a mixture of baseline programming languages and baseline programming paradigms.

7.4.2.5. Node Specifications [______, ______, ______]

The job launcher shall support the ability to specify the nodes to launch the job on. This functionality shall work outside the batch system.

7.4.2.6. I/O Redirection [______, ______, ______]

The job launcher shall support the capability of redirecting STDIO, STDERR, and STDIN.

7.4.2.7. Environment Variables [______, ______, ______]

The job launcher shall support the capability of passing environment variables to the application.

7.4.2.8. Credentials [______, ______, ______]

The job launcher shall be capable of passing user credentials to the computational nodes.

7.4.2.9. Account Name [______, ______, ______]

The job launcher shall support the capability of specifying an account name to charge the resource consumption against.

7.5. Network Software

7.5.1. Monitoring of Network Interfaces [______, ______, ______]

All network interfaces shall be capable of being monitored via the Vendor's supplied SNMP (Simple Network Management Protocol) agent that supports MIB-II (Management Information Base).

7.5.2. Host Resolution [______, ______, ______]

Host resolution shall have the capability of being arbitrarily ordered (DNS, NIS, `/etc/hosts`). Re-ordering can be achieved by specifying a new ordering in `/etc/nsswitch.conf`.

7.5.3. Internet Protocol (IPv4) [______, ______, ______]

IPv4 shall be supported on all network interfaces. The Vendor’s IPv4 implementation shall also conform to RFC 1323, “TCP Extensions for High Performance” (window shift feature), and MTU (Maximum Transmission Unit) discovery.

7.5.4. Internet Protocol (IPv6) [______, ______, ______]

IPv6 shall be supported on all network interfaces. The Vendor’s IPv6 implementation shall also conform to RFC 1323, “TCP Extensions for High Performance” (window shift feature), and MTU (Maximum Transmission Unit) discovery.

7.5.5. Packet Filtering [______, ______, ______]

The operating system shall support the capability of filtering packets on all interfaces. This shall be dynamically configurable.

7.6. Filesystem Software

7.6.1. Network File System (NFS) [Yes, Yes, Yes]

7.6.1.1. Network File System Version 3 [______, ______, ______]

The Vendor shall provide support for NFS version 3 (NFSv3). The Vendor shall also provide both the client and server versions of the software. It shall be supported on all network interfaces.

7.6.1.2. Network File System Version 4 [______, ______, ______]

The Vendor shall provide support for NFS version 4 (NFSv4) (date available to be determined). The Vendor shall also provide both the client and server versions of the software. It shall be supported on all network interfaces.

7.6.1.3. NFS Maximum Filesystem Size [______, ______, ______]

The NFSv3 and NFSv4 architecture shall architecturally support a 500 TB filesystem.

7.6.1.4. NFS Maximum File Size [______, ______, ______]

NFSv3 and NFSv4 shall support a single file size equal to the maximum size of the filesystem that is being exported.

7.6.1.5. POSIX Compliant NFSv3 and NFSv4 [______, ______, ______]

NFSv3 and NFSv4 filesystems shall follow the POSIX standards.

7.6.1.6. NFS File Count [______, ______, ______]

A single NFSv3 or NFSv4 filesystem shall be able to support up to the maximum number of files of the native filesystem which is being exported.

7.6.1.7. NFS File Access Time [______, ______, ______]

NFSv3 and NFSv4 shall be able to access an existing file in a 95% full NFSv3 or NFSv4 filesystem within a 5% time delta of the same file in a 5% full NFSv3 or NFSv4 filesystem.

7.6.2. Parallel Filesystem (GPFS, Lustre, or PNFS)

7.6.2.1. GPFS Open [______, ______, ______]

The Vendor shall supply and support a mechanism for GPFS filesystem access from the system..

7.6.2.2. 1 PB Filesystem [______, ______, ______]

The global parallel filesystems shall support a size of 1 PB.

7.6.2.3. Single File Size Equal to Size of Filesystem [______, ______, ______]

The filesystem shall support a single file equal to the size of the filesystem.

7.6.2.4. File Access Time for Full Filesystems [______, ______, ______]

The global filesystem shall be able to access an existing file in a 95% full filesystem within a 5% time delta of the same file in a 5% full global filesystem under the same system load.

7.6.2.5. Automatic Filesystem Recovery [______, ______, ______]

A SIO node failure shall be detected by the global filesystem or any of its subcomponents and recovery actions launched automatically.

7.6.2.6. Global Filesystem Recovery [______, ______, ______]

Failure of a global filesystem component or service node shall be detected and recovery actions initiated. A user process shall not fail due to a pending request that cannot be serviced during the recovery period.

7.6.2.7. 10 Million Files per TB of Storage [______, ______, ______]

The global filesystem shall support 10 million files per TB of filesystem storage.

7.6.2.8. DMAPI Support [______, ______, ______]

The global filesystems shall support the DMAPI standard.

7.6.2.9. Backup/Restore [______, ______, ______]

The Vendor shall provide and support an efficient mechanism for backing up and restoring global filesystems.

7.6.2.10. System Software Characteristics [______, ______, ______]

The global filesystems shall have the following traits:

- High availability
- High performance
- Zero corruption of data
- Zero loss of data

7.6.2.11. User/Group Disk Quotas [______, ______, ______]

The global filesystems shall provide user and group inode and block quota limits.

7.6.2.12. Quotactl [______, ______, ______]

The global filesystems shall utilize the quotactl system call for manipulating disk quotas.

7.6.2.13. Metadata Update Times [______, ______, ______]

The global filesystems shall update metadata globally within 300 seconds. Updates on the node performing the update operation shall be updated immediately.

7.6.2.14. File Creation Timing [______, ______, ______]

A global filesystem shall be capable of creating a million files in a single directory in less than 1 hour using standard system calls on a single node.

7.6.2.15. Directory Tree File Creation Timing [______, ______, ______]

A global filesystem shall be capable of creating a million files in a directory hierarchy in less than 2 hours using standard system calls on the same node. The directory hierarchy shall be defined and created by a site-defined test.

7.6.2.16. File Listing Timing [______, ______, ______]

A global filesystem shall be capable of performing stat system calls on a million files in a single directory in less than 30 minutes**.**

A global filesystem shall be capable of generating a long listing (e.g., ls -l) of a million files in a directory hierarchy in less than 40 minutes. The directory hierarchy shall be defined and created by a site-defined test.

7.6.2.17. Filesystem Statistics [______, ______, ______]

The Vendor shall provide, support, and document an API for accessing global filesystem usage and performance statistics.

7.6.2.18. 18 GB/s Aggregate Bandwidth [______, ______, ______]

The global filesystem shall be capable of sustaining an aggregate bandwidth of at least 18 GB/s.

7.6.2.19. 1 GB/s Single Stream Bandwidth [______, ______, ______]

The global filesystem shall be capable of sustaining a bandwidth of at least 1 GB/s from a single compute node.

7.6.2.20. 10,000 Aggregate Metadata Operations per Second [______, ______, ______]

The global filesystem shall be capable of sustaining an aggregate of at least 10,000 metadata operations per second.

7.6.2.21. 2,500 Single Node Metadata Operations per Second [______, ______, ______]

The global filesystem shall be capable of sustaining at least 2,500 metadata operations per second from a single node.

7.6.2.22. 1 PB Filesystem [______, ______, ______]

The global parallel filesystem shall support a size of 1 PB.

7.6.2.23. Single File Size Equal to Size of Filesystem [______, ______, ______]

The filesystem shall support a single file equal to the size of the filesystem.

7.6.2.24. File Access Time for Full Filesystems [______, ______, ______]

The global filesystem shall be able to access an existing file in a 95% full filesystem within a 5% time delta of the same file in a 5% full global filesystem under the same system load.

7.6.2.25. Automatic Filesystem Recovery [______, ______, ______]

A service/IO node failure shall be detected by the global filesystem or any of its subcomponents and recovery actions launched automatically.

7.6.3. Parallel I/O [______, ______, ______]

The global filesystem shall support simultaneous file access to one or more files from all nodes in the cluster via multiple I/O paths.

7.6.4. Large Files [______, ______, ______]

The global filesystem shall support files the size of the filesystem containing them.

7.6.5. Storage Network Management [______, ______, ______]

All storage network interfaces shall be capable of being monitored via the Vendor’s supplied SNMP (Simple Network Management Protocol) agent that supports MIB-II (Management Information Base).

7.7. Security

DOE Office of Science sites’ primary security focus is to prevent unauthorized access. Hostile actions, intentional or unintentional, shall not inhibit access by users to DOE resources.

7.7.1. X/Open GSS-API Standard Interface [______, ______, ______]

The system shall support a complete implementation of the X/Open GSS-API standard interface. In areas where the system is not compliant with the standard, the Vendor shall clearly identify lack of compliance.

7.7.2. Documented Port Numbers [______, ______, ______]

The Vendor shall provide documentation identifying all port numbers used by the Vendor-supplied software.

7.8. System Management Facilities

7.8.1. Single Point of Control [______, ______, ______]

The Vendor's system shall be managed from a single point of control. The Vendor's system maintenance and control workstations are able to boot, dump, and monitor components of the system.

The disk subsystem is powered on and shut down separately. Once powered on, the disk subsystem is managed through the system maintenance workstation.

7.8.2. Warm Boot in 15 Minutes [______, ______, ______]

The Vendor's system shall boot from all nodes to full production status at a powered-on but shut-down state in 15 minutes. Production status is reached when users are capable of establishing login sessions, all filesystem space is available across all nodes, and batch jobs are running.

7.8.3. Cold Boot in 1 Hour [______, ______, ______]

The Vendor's system shall boot all nodes from a powered-off state to full production status in 1 hour. Production status is reached when users are capable of establishing login sessions, all filesystem space is available across all nodes, and batch jobs are running.

7.8.4. Internal Service, Diagnostic, and Analysis Tools [______, ______, ______]

Upon request, the Vendor will provide any internal service, diagnostic, or analysis tools that are used by Vendor service personnel. The site agrees to keep such tools confidential.

7.8.5. System Dump Facility [______, ______, ______]

The Vendor will provide a dump tool that analyzes and collects information from a system that is failing or has failed, crashed, or hung. This analysis is performed on, but not limited to, event log data, active heartbeat probing, voltages, temperatures, health faults, portals trace buffers, in-memory console buffers, and high-speed interconnect network errors. When failed components are found, detailed information is gathered from those components.

7.8.6. Node Diagnostics [______, ______, ______]

There are no online diagnostics. The Vendor shall supply the following offline diagnostics:

- *memtest:* Tests memory; detects memory faults and memory-related failures.
- *cpuburn:* Runs a loop of instructions for a specified time (intent is to heat CPUs).
- *adaptor check:* Tests internal functions of and interfaces to the CPU and L0; detects bad and failed interfaces in the path from L0 to adaptor to CPU.

- *link:* Simple HSN ping test between an adaptor and its nearest neighbor adaptor; detects bad adaptor and simple interconnect problems.

7.8.7. System Monitoring [______, ______, ______]

The Vendor shall provide a set of applications, each of which has a graphical user interface (GUI), that enables the site to perform monitoring and system management tasks by directly manipulating icons that represent system objects.

7.8.8. Node Warm Boot [______, ______, ______]

A warm boot of a single compute and service/IO node shall complete in less than 4 minutes.

7.8.9. Node Cold Boot [______, ______, ______]

A cold boot of a single compute and service/IO node shall complete in less than 8 minutes.

7.9. Node-Level Activity Monitoring [______, ______, ______]

The Vendor shall provide and support node-level activity monitoring via `sar`.

7.10. Application Development Environment

7.10.1. Baseline Languages, Parallel Programming Models, and Libraries [______, ______, ______]

Baseline languages include assembler, Fortran, C, UPC, and C++ as described in Programming Languages below. Baseline parallel programming models include MPI and SHMEM. Baseline libraries include SCALAPACK, SuperLU, ACML, and Portals. The baseline languages and parallel programming models shall be extended to include OpenMP and Posix threads as described below.

7.10.2. Optimizing Compilers [______, ______, ______]

Baseline programming language compilers shall be capable of producing highly optimized executables. The compilers shall provide statistics and information regarding the nature of the optimizations performed on the source code in a readily understandable format.

7.10.3. IEEE Conformance [______, ______, ______]

All numerical library routines support ANSI/IEEE Std 754-1985 32- and 64-bit (as appropriate) numerical formats. All transcendental functions in ACML and libm return results shall be accurate to within 1, or in limited cases, 2 ulps. The libm routines used by the PGI and gcc compilers are IEEE754 and C99 compliant. The corresponding routines in the ACML library achieve the same accuracy, but obtain higher speed by not raising exception flags in exceptional cases and in that respect only are not C99 or F2003 compliant. All other numerical routines in ACML, libm, and libSci use numerically sound algorithms and achieve the levels of accuracy that are appropriate and expected for data in ANSI/IEEE Std 754-1985 basic 32- and 64-bit (as appropriate) numerical formats.

7.10.4. Capability Computing [______, ______, ______]

The site shall have the ability to apply all compute resources of the system to a parallel application. This application may be composed of a mixture of baseline programming languages and baseline programming paradigms.

7.10.5. Interoperability [______, ______, ______]

The Vendor shall provide the fully supported capability to build a single parallel program from a mixture of baseline languages and baseline parallel programming models. Inter-language sub-procedure invocation must be supported.

7.10.6. Memory Utilization [______, ______, ______]

An application written using a baseline programming language and using a baseline parallel programming model shall be able to access and utilize all user-available memory on each compute node.

7.10.7. Page Sizes [______, ______, ______]

The Vendor shall support both small (4 KB) and large (2 MB) pages. An application written using a baseline programming language and using a baseline parallel programming model shall be able to exploit large pages for program data.

7.11. Programming Languages

7.11.1. Assembler [______, ______, ______]

The Vendor shall provide and support an assembler.

7.11.2. Standards-Compliant Fortran [______, ______, ______]

The Vendor shall supply and support a standards-compliant Fortran compiler with the system as part of a complete application development environment. The development environment shall be licensed for the entire system. Support is required for Fortran 77 (ANSI X3.9-1978), Fortran 90 (ANSI X3.198-1992), and Fortran 95 (ISO/IEC 1539-1:1997). Support for Fortran 2003 (ISO/IEC 1539-1:2004) shall be required by 1 Jan 2007.

7.11.3. Standards-Compliant C [______, ______, ______]

The Vendor shall supply and support a standards-compliant C compiler with the system as part of a complete program development environment. The development environment shall be licensed for the entire system. Support is required for at least C89 (ANSI/ISO 9899-1989) and C99 (ISO/IEC 9899-1999).

7.11.4. Standards-Compliant C++ [______, ______, ______]

The Vendor shall supply and support a standards-compliant C++ compiler with the system as part of a complete program development environment. The development environment shall be licensed for the entire system. Support is required for ISO/IEC 14882:2003 (core language and C++ standard library).

7.11.5. Standards-Compliant Java [______, ______, ______]

The Vendor shall supply and support a standards-compliant Java compiler with the system as part of a complete program development environment. The Java development and runtime environment shall operate and be licensed for the system partition only. Support is required for J2SE 5.0.

7.12. Programming Models

7.12.1. MPI [______, ______, ______]

The Vendor shall supply and support a library implementing the MPI 2.0 standard (except Section 5.0, Process Creation and Management). This library shall work efficiently over the Portals interface.

7.12.2. SHMEM-Open [______, ______, ______]

The Vendor shall supply and support a library implementing the SHMEM API as described in Section 11 and the intro_shmem man page (1/17/06). This library is modified to work efficiently over the Portals interface.

7.12.3. UPC [______, ______, ______]

The Vendor shall supply and support a library implementing the UPC programming model as described in the UPC Language Specifications V1.2 (May 2005). This shall be delivered by 3rd quarter 2007.

7.12.4. Threads [______, ______, ______]

Threads are not supported

7.12.5. OpenMP [______, ______, ______]

OpenMP is not supported.

7.13. Libraries and Applications

7.13.1. Core Math Library [______, ______, ______]

The Vendor shall supply and support the optimized basic libraries for the specific hardware deployed:

- Level 1, 2, and 3 Basic Linear Algebra Subroutines (BLAS)
- A full suite of linear algebra routines (LAPACK)
- A suite of fast Fourier transform routines (FFT) for single-precision, double-precision, single-precision complex, and double-precision complex data types
- Vector math library support for exp, log, sin, cos, and sincos

7.13.2. ScaLAPACK Library [______, ______, ______]

The Vendor shall supply and support the ScaLAPACK library, a set of linear algebra routines redesigned for use in parallel applications. The library shall comprise all auxiliary libraries, BLACS and PBLAS, and conform to the public releases of ScaLAPACK 1.7, BLACS 1.1, or later versions.

7.13.3. SuperLU Library [______, ______, ______]

The Vendor shall supply and support the SuperLU library, a set of routines that solve large, sparse, nonsymmetrical systems of linear equations. The library is written in C but can be called from programs written in either C or Fortran. The library shall conform to LBNL technical report LBNL-44289.

7.13.4. MPICH2 Library [______, ______, ______]

The Vendor shall supply and support the MPICH2 library, an implementation of the Message Passing Interface (MPI). This library is modified to work efficiently over the Portals interface. The spawn functions available in MPICH2 are not supported at this time, but otherwise the libraries are fully MPI 2.0 compliant. Programmers can use the MPICH2 libraries to write applications in Fortran 90, C, or C++.

7.13.5. Portals 3.3 API [______, ______, ______]

The Vendor shall supply and support the Portals 3.3 application programming interface, a connectionless, low latency, low overhead message passing protocol. Portals provide the interface for message passing between all nodes in the Vendor's system. Application processes communicate with one another by linking libraries that support the Portals 3.3 interface. Programmers shall have the option of using the Portals 3.3 API directly when necessary.

7.13.6. ROMIO Library [______, ______, ______]

The Vendor shall supply and support ROMIO, an implementation of MPI-IO optimized for noncontiguous access patterns based on MPI 2.0 Standard section 9.

7.13.7. Glibc Library [______, ______, ______]

The Vendor shall supply and support the GNU C/C++ glibc, a C language library for the system that has been optimized for the lightweight kernel.

7.14. Development and Performance Tools

Unless specifically noted, all tools described in this section shall be capable of working with applications that utilize all the compute cores and/or all user-accessible memory available on the system.

7.14.1. Graphical User Interface API [______, ______, ______]

The Vendor shall provide the standard version of X11R6 and Motif, or current versions, applications, servers, and API libraries on the SIO nodes.

7.14.2. Debugger for Parallel Applications [______, ______, ______]

The Vendor shall provide a scalable debugger for parallel applications (e.g., TotalView from Etnus Inc.).

7.14.3. Hardware Performance Counter Analysis Tools for Parallel Applications [______, ______, ______]

The Vendor shall provide CPU units with hardware counters. The interface between hardware and user-level tools shall be provided by the kernel extension perfctr version 2.6 or later or equivalent.

7.14.4. Performance API [______, ______, ______]

The Vendor shall supply and support the Performance Application Programming Interface (PAPI) version 3.2 or l.

7.14.5. Modules Utility [______, ______, ______]

The Vendor shall supply and support the Modules utility.

7.14.6. Performance Analyzer Tool [______, ______, ______]

The system shall have the ability to create an instrumented executable on an application build from baseline programming languages and baseline parallel programming models without changing the application code. By means of flags to commands or environment variables, instrumented executables can be used in tracing, profiling, and sampling experiments. Each experiment can record runtime, hardware performance counters, routine arguments, statistics relevant to the parallel programming model (e.g., MPI routine times and achieved bandwidths), or I/O performance. Tools to summarize such experiments by routine source line, routine, process rank, and application run shall be provided.

The system shall have the ability to collect hardware performance counter data including but not limited to: level 1 and level 2 data and instruction cache accesses, reads, hits and misses, data and instruction TLB misses; level 1 and level 2 load misses, fixed and floating point instructions, total cycles, stalled cycles, branch instructions (including those taken and correctly and incorrectly predicted), vector or SIMD instructions, and floating point instructions per second.

Use of the performance analysis tool shall be scalable to the full size of the system with appropriate choice of experiment.

Vendor shall supply and support a tool to visualize the performance data collected. The tool shall present visualizations such as pie charts and event timelines in a readily understandable manner.