Whitepaper on the

Major Computer Science Challenges at Exascale[1]

February 2009

Al Geist, ORNL and Robert Lucas, ISI

Exascale systems will provide an unprecedented opportunity for science, one that will make it possible to use computation not only as a critical tool along with theory and experiment in understanding the behavior of the fundamental components of nature but also for critical advances for the nation's energy needs and security. To create exascale systems and software that will enable DOE to meet the science goals critical to the nation's energy, ecological sustainability, and global security, we must focus on major architecture, software, algorithm, and data challenges, and build on newly emerging programming environments. Only with this new infrastructure will applications be able to scale up to the required levels of parallelism and integrate technologies into complex coupled systems for real-world multidisciplinary modeling and simulation. Achieving this goal will likely involve a shift from current static approaches for application development and execution to a combination of new software tools, algorithms, and dynamically adaptive methods. Additionally, we must bring together new developments in system software, data management, analysis, and visualization to allow disparate data sources (both simulation and real-world) to be managed in order to guide research and to directly advance science. Achieving this vision will require fostering long-term, sustained, communitywide activity in evolving code suites. Large-scale applications, like large-scale computers themselves, require the support of multiple specialists within a single community. Indeed, the community of computer vendors, application scientists, and computer scientists, together with the hardware and software they both develop and use, form an integrated, interdependent ecosystem.

Several recent studies and workshops [1-10] have identified the high level problems facing the HPC community as it moves towards exascale over the next decade. This paper compiles and organizes the major software challenges into four categories:

- Problems caused by the growing scale and complexity of computer architectures
- Problems caused by the growing complexity of science applications, including the longstanding problems with debugging and tuning large applications at scale
- Problems are caused by the huge increase in the data produced and consumed by peta and exascale systems.
- Problems of software sustainability such as hardening and long-term support of popular software packages, education of the next generation of HPC specialists, and training the existing users about advanced techniques and tools.

These workshops pointed out that it is critical that work begin today if the DOE's scientific computing community is to be able to exploit exascale systems when the technology to create them matures in the coming decade.

1. Challenges due to scale and complexity of system

[1] Work in process. Based on an analysis of the computer science challenges from the DOE Exascale studies.

For most of the past five decades, the growing computational power of supercomputers has come primarily from a doubling of clock frequency every 18 months. In the last two decades, this has been augmented by an increase in the number of processors. Over this time period, the clock rate increased by six orders of magnitude, while the number of processors increased by three orders of magnitude. Due to constraints on heat and the power requirements of today's microprocessors, the last frequency doubling occurred about five years ago and has remained effectively constant ever since. Vendors have shifted to putting multiple processors (cores) on a chip; first two, then four, then eight. The number of cores per chip is expected to continue to increase exponentially over the next decade. Today's supercomputer vendors see the only way to continue increasing the computational power of their systems is through increasing the number of processors and hence the scale and complexity of their systems. In the last five years supercomputer architectures have gone from 1000 processors to 100,000 processors and the next generation systems are going to have over a million processors. The rate of growth of parallelism is in fact accelerating, and will likely exceed one hundred million when exascale systems appear. Some estimates even predict that the need for multiple threads to cover main memory and communication latency means that scientific codes will contain billions of threads.

The change of shifting from using faster processors to using multi-core processors is as disruptive to scientific software as the shift from vector to distributed memory supercomputers fifteen years ago. That change required complete restructuring of scientific application codes, which took years of effort. Some application communities still haven't transitioned to even a thousand-way parallelism. The shift to multi-core exascale systems will require applications to exploit million-way parallelism and overcome significant reductions in the bandwidth and volume of memory available to each CPU. This "scalability challenge" driven by the exponential increase in the amount of parallelism in the system affects all aspects of the use of high performance computing. It makes all the existing problems harder, such as getting performance from the applications, managing the system, debugging, etc.. It also creates new challenges such as fault tolerance, the need for new programming models, and verification of results.

There is another looming shift in the complexity of the node architectures that will be as big a challenge to software development as the exponential growth in processors. This is the potential shift to heterogeneous node architectures. Today most supercomputers are of huge scale but they are homogeneous. Over the next decade it is expected that the multi-core processors will include several different types of cores on each node, for example, a computation accelerator, a graphics processor, a communication processor, an IO processor, etc. An early example of a heterogeneous system is the Roadrunner supercomputer at LANL.

The major challenges caused by the increasing scale and complexity HPC systems are cross cutting of the entire software stack. The software challenges include the rapid increase in parallelism, the memory wall, system heterogeneity and fault tolerance. For each of these challenges computer science research is needed across the entire stack not just at one level. For example, making an application fault tolerant is not sufficient if the system software is not also fault tolerant. Making the system software fault tolerant is not sufficient if the data can be corrupted by faults in the data management software. To be able to use these systems to solve the nation's problems, DOE, as the pioneer in HPC, must improve all parts of the software stack and influence the architecture design to meet the scientific needs. The challenges impact both the developers and users of the system software, the applications, the runtime, communication, IO, and data management, including analyzing the results.

1.1 Increasing Parallelism

The increase of system concurrency from hundreds of thousands to hundreds of millions will be a tremendous challenge for system software to manage and for applications to get good performance at this level of parallelism. Almost all of today's large-scale applications use the message-passing programming model (MPI) together with traditional sequential languages (C, Fortran, C++), but new architectures with many cores per chip and parallelism in the millions are expected to make this programming model more problematic and less productive in the future. Thus new approaches are needed. For example, a hybrid programming model such as MPI with some global view techniques such as Unified Parallel C (UPC) or Co-Array Fortran (CAF). In order to facilitate the utilization of the extreme scale resources, new programming models and High Productivity Computer Systems (HPCS) languages must be explored.

1.2 Memory Wall

The memory wall traditionally refers to the challenge that the bandwidth and latency to memory continues to grow at a slower rate than the processor power. The transition from frequency-based scaling to core-based scaling will make the memory wall both higher and broader. It is higher in that bandwidth and latency continue to get worse as memory gets farther away from CPU operations (at least in terms of clocks). The memory wall is going to get broader in that the overall memory capacity per core must decrease. It will be harder and harder to maintain the desired byte-to-flop ratio—in absolute capacity (flops/s per byte) and bandwidth terms (flops per byte). Hence, applications will have to be redesigned to make better user of limited memory. Additionally, applications will have to deal with increasing hierarchies of memory (and indeed storage). There are now often five levels of direct access memory (register sets, three levels of cache, and main memory). In the future there may be more levels and more (or less) sharing of these levels within a shared memory node, as well as a new level of persistent FLASH to augment the DRAM main memory.

1.3 Influencing Architecture Design

DOE scientists have been pioneering users of high-end systems for over five decades. While the systems themselves are usually manufactured and deployed by computer system vendors, architecture research conducted by DOE scientists, often in collaboration with the vendors, allows DOE to develop the specifications for the systems. To maximize the utility of the computer hardware, DOE computer scientists often contribute everything from system software to programming environments and debugging tools. Recent examples abound, including BlueGene/L, Red Storm, and Roadrunner. As we look forward to exascale, high-end systems will become increasingly specialized, and DOE scientists will have to take an even more active role in designing of both the software and the hardware of such systems to assure their utility for the scientific problems that face the nation.

1.4 Heterogeneity

Heterogeneity exists at many different levels in modern supercomputers. The systems have several different node types: compute, IO, login; several different operating systems; and several different interconnection networks: RAS network, command network, one or more communication networks. Despite this heterogeneity, these systems are usually considered homogeneous because the fundamental compute node is homogeneous and replicated tens of thousands of times across the system. A heterogeneous system is one where there are regions of different compute nodes across the system. For example the proposed Japan 10 PF system is designed to be a mix of three different types of architectures: vector, cluster, and specialized (GRAPE). Another form of heterogeneous system is where the compute nodes are heterogeneous. An example is the “Roadrunner” system where each compute node has a traditional AMD multi-core processor plus an IBM Cell processor, originally designed for the Sony Playstation. The major chip vendors have all started exploring creating heterogeneous multi-core chips that combine light-weight, high compute density processor units (e.g., GPUs) and traditional computational units (CPUs) in order to increase the computational power on a single chip. It is expected that over the next decade most supercomputers will be constructed using such heterogeneous multi-core processors.

Heterogeneity is also appearing at the system level, as computer centers adopt a “crop rotation” model, whereby systems are partially updated on a regular basis. A recent example occurred at the ORNL

Leadership Computing Facility, when two generations of Cray XT systems were simultaneously deployed.

Heterogeneity is a radical shift from today's environment. System management, job scheduling, efficient resource utilization, and load balancing all become much more complex. Today's code development assumes a homogenous run-time environment, with parallelization being done manually by each code developer. At the scale where applications need to make use of millions of heterogeneous processes, discovering the opportunities for parallelization becomes much more difficult and requires a set of tools that can automate the parallelization of the trivially parallelizable segments of code, and aid the application developer in finding less obvious opportunities. This task is even more daunting when considering future heterogeneous multi-core architectures, since the parallelization algorithms have to take into account the different types of processors and the interactions between them. Compiler research will be needed to understand how to exploit heterogeneous hardware, automating as much of this as possible and providing code-restructuring assistance where automation is not possible.

1.4 Fault Tolerance

Modern PCs may run for weeks without rebooting and more data servers are expected to run for years. However, because of their scale and complexity, today's supercomputers run for only a few days before rebooting. Exascale systems will be even more complex and have millions of processors in them. The major challenge in fault tolerance is that faults in extreme scale systems will be continuous rather than an exceptional event. This requires a major shift from today's software infrastructure. Every part of the exascale software ecosystem has to be able to cope with frequent faults; otherwise applications will not be able to run to completion. The system software must be designed to detect and adapt to frequent failure of hardware and software components. On today's supercomputers every failure, even ones that get reconfigured around, kills the application running on the affected resources. These applications have to be restarted from the beginning or from their last checkpoint. The checkpoint/restart technique will not be an effective way to utilize exascale systems, because checkpointing stresses the I/O system and restarting kills 999,999 running tasks because 1 fails in a million task application. With the potential that exascale systems will be having constant failures somewhere across the system, application software isn't going to be able to rely on checkpointing to cope with faults. A new fault will occur before the application could be restarted, causing the application to get stuck in a state of constantly being restarted. For exascale systems, new fault tolerance paradigms will need to be developed and integrated into both existing and new applications.

To complicate matters even more, the GPU accelerators that are being considered for heterogeneous systems often do not have any error checking on the processors or in their memories. This is because there is no market force to require error checking since a few incorrect pixels on the frame of an animation is not noticeable. But if GPUs become common in peta and exascale systems then undetected errors from GPUs or other sources could dramatically increase the rate of faults in large systems.

Research in the reliability and robustness of exascale systems for running large simulations is critical to the effective use of these systems. New paradigms must be developed for handling faults within both the system software and user applications. Equally important are new approaches for integrating detection algorithms in both the hardware and software and new techniques to help simulations adapt to faults.

2. Challenges due to complexity of applications

As computational capabilities have grown, so have the resolution and complexity of the simulation models. The large simulation codes today incorporate multidiscipline, multi-physics, multiple time scale and multiple solution methods. They have taken years to develop by teams of programmers and scientists and can include millions of lines of code. As we make the leap to exascale computation the impact on the cost to update, recode, and incorporate more advanced models into the simulations can be an order of magnitude higher than the cost of the supercomputer hardware. In order to contain these costs, the exascale software ecosystem must support more efficient program development that addresses the following application challenges:

- Scaling limitations of present algorithms
- Innovative algorithms for multi-core, heterogeneous nodes
- Software strategies to mitigate high memory latencies
- Hierarchical algorithms to deal with BW across the memory hierarchy
- Need for automated fault tolerance, performance analysis, and verification
- More complex multi-physics requires large memory per node
- Model coupling for more realistic physical processes
- Dynamic memory access patterns of data intensive applications
- Scalable IO for mining of experimental and simulation data

The user requirements are heavily shaped by the length of the life cycle of the applications. HPC applications have both long development cycles and long periods during which the application is in "production." An important aspect of this life cycle is that code is always in development -- even production code. Thus, the users require assurances of stable support for a programming model, including the development tools that enable its use. Further, "new" applications are almost never entirely new—they almost always take some existing code base to provide key underlying physics or mathematics functionality from an existing application. As a result, users are not open to tools that only target "new" applications or require significant changes to the established workflow of the application team.

Applications are becoming much more multifaceted as teams include a variety of languages, libraries, programming models, data structures, and algorithms in a single application. In fact, application teams are listing scalable tools for debugging, memory correctness, thread correctness, and multimode performance analysis as key factors in their productivity.

Today's tools are limited in scope, capability, and scalability. The overhead associated with current measurement techniques is too intrusive at the petascale and may skew analysis so much as to render any analysis ineffective. Therefore, we need to develop scalable and less intrusive methods of collecting performance data, develop knowledge discovery methods for extracting key performance features, and provide assistance in feeding the results of these analyses back to the code transformation.

2.1 Improving Programmability

Exascale computer architectures will require radical changes to the software used to operate them and the applications that run on them.

New ways of specifying computations: Scientists must be freed from the details of managing data movement among memory systems and synchronizing access to shared memory among threads of control. They will need languages and libraries, in some cases discipline- or even application-specific, which specify results to be obtained with less attention to the details of the computation than is currently necessary. Implementation of such libraries and languages will require lower-level programming models and tools that permit execution on a wide range of hardware and exploit the capabilities of exascale architectures.

Portability: Libraries are the typical software test beds where new programming models and execution models are proved out and this will continue to be the case. Numerical and communication libraries provide a fast vehicle for getting the new concepts into use by the application developers. MPI is the portable programming model today. Any new programming model that is created must be, at a minimum, be as portable across the key HPC systems, clusters, and development platforms of the time to be adopted by software developers. Tools to assist in the code transformations to new models and new algorithms are going to be critical in transitioning the millions of lines of code to a new programming model.

Huge code size: The increasing prevalence of coupled multi-disciplinary codes has combined with the long life cycle of scientific applications and the use of third party libraries to make codes larger and more complex. As a result, tools must handle larger executables. Tool developers are already seeing demand for tools to handle codes of several hundred mega-bytes to giga-bytes of executables. In addition, the rise of component based programming is resulting in applications that have hundreds if not thousands of shared libraries. So tools must be developed that handle both huge binary files as well as large numbers of files.

The memory challenge states that the memory per core in petascale architectures is going to decrease. Developers need tools that will help them understand the scaling behavior of memory allocations and usage as well as detect correct memory semantics. With limited node memory, tools that monitor how much memory is being used in a parallel job over time would also be useful. To be applied at extreme scale, all of these tools must have little overhead, a criteria that many existing memory correctness tools fail to meet.

Tools throughout software life cycle: The tool needs vary with the life cycle stage. Initial code developers need full featured debuggers and performance analysis tools and are willing to work with tools with relatively high overheads, such as some memory correctness tools. Similar functionality is also needed for code being maintained. In addition, support for version tracking, code coverage and regression testing (both correctness and performance) are useful at this stage. Supporting code ready to run at large scale requires yet different tools. Lightweight debugging functionality is essential at these scales, as are low overhead mechanisms for performance profiling and analysis. Codes in production need workflow tools to interact with applications and large scale systems. Finally, tools to support fault tolerance, with a focus on data integrity, are expected to become even more important during this life cycle stage as faults become continuous.

2.2 Building New Applications

The vision for the next decade is to have a totally integrated approach to how applications are built, modified, updated, and used in other applications. In such a development environment the tools will interoperate with each other and assist the scientists in writing, debugging, tuning, and maintaining their codes. This will be facilitated through:

Rapid, modular construction of new applications from existing suites of interoperable components. Scientific software components with well-defined interfaces have the potential to greatly increase code reuse, thus shortening development times and increasing software reliability.

Coupling of multiple applications into ever-larger applications through automated workflows. Single large runs remain an important class of large-scale computations, but many applications need parameter studies consisting of large numbers of coordinated sets of runs, each perhaps consistent of a pipeline of computation and analyses. High-level, standard languages for coordinating such families of executions will enable scientists to focus on science rather than “run management.”

Debugging Tools. An integral part of application development includes verifying that code runs as expected. Current debuggers are not able to handle even a few thousand tasks much less the 100,000 tasks on today’s supercomputers. Application developers for today’s large systems fall back to the very inefficient method of debugging—dumping user inserted debug code to output files. With the vast increase of process count going to exascale systems, searching manually for a single anomalous process among the millions of running processes and threads is not tenable.

Application teams need tools for managing application builds and configurations, mixed language support, dynamic linking, program configurations, remote access, compiler infrastructures for application-specific analysis and transformations, and integrated development environments. Application teams specifically request lightweight tools to diagnose memory, threading, and message passing errors that are easy to use and scale from the desktop system to the petaflop platform. Furthermore, the architectures and system software must make the necessary performance and reliability information available to these tools so that they can perform root-cause analysis with greater accuracy.

For performance and correctness tools the availability of scalable tools is particularly critical. These tools require a scalable infrastructure to provide tool communication, data management, binary manipulation of application executables, execution management for batch schedulers and operating systems, and a variety of other capabilities. Tool infrastructures must be efficient, modular, fault tolerant, and flexible.

2.3 Execution Environment

Managing a system with a million processors and faults occurring almost continuously produces new challenges for system software. Efficient scheduling and resource management become significantly harder with a dynamically changing configuration as does upgrading and monitoring. The acceleration in scale puts additional pressure on the scaling of all system software components. In particular, OS scaling has been a historical challenge at each change in scale. Several performance issues are anticipated to become of increasing importance. Perhaps at the top of the list is load balancing. Tools are needed to detect load balance problems and to assist the dynamic load balancing of applications.

In order to guide research, and to directly advance science there must be a more flexible and dynamic resource management capability throughout the computing environment to allow computing, analysis, visualization, and live data to be integrated simultaneously during a simulation. While workflows provide a nice execution interface for the scientist, they will need to evolve to meet the needs of the growing complexity of applications.

a) **Semantic awareness in workflows**: Workflows need intelligence to identify which actors can be linked technically, highlight mismatch of units between actors, enable better control over parameter sweeps, and learn from previous workflows.
b) **Optimization of workflows:** Currently workflows are driven by the need to optimize the scientist's time. In the future they may also need to consider other options such as optimizing power, CPU cycles and data transmission time by dynamically scheduling on appropriate systems. This will require that the workflows be aware of the underlying hardware.
c) **End-to-end software environments to support collaborative data analysis:** As advances in mathematics and computer science make the analysis of larger and more complex data sets feasible, it is also necessary to bring these advances together in an environment that supports the end-to-end process of data analysis from the initial data to the final results. This environment should include support for workflows, provenance, and storage of data.
d) **Incorporation of policies:** Workflows will also need to incorporate any privacy and security policies that may dictate how and what data can be analyzed.

2.4 Validation and Verification

The scale and complexity of the science problems enabled by exascale systems require new techniques for making sure that the calculations are done correctly. It will be increasingly important to validate that new extreme scale algorithms are solving the right problem and to verify that the answer produced is correct and not corrupted by numerical stability or numerical errors from transient non-fatal faults.

The difficulties in drawing scientifically-meaningful conclusions from vast volumes of data is reflected in the greater need for code validation, uncertainty quantification, and the analysis of data across ensembles of simulations. Often, the quality of the data, and the variation in the data, add to the challenges resulting from the massive size of the data, thus increasing the need for robust algorithms that are not sensitive to the settings of parameters.

3. Challenges due to increased data

The data challenges include dealing with the volume, different formats, transfer rates, analysis, and visualization of massive (potentially distributed) data sets. Exascale applications running on as many as a million processors are likely to generate data at a rate of several terabytes per second (even assuming only a few megabytes per processor). It is not practical to store raw data generated at such a rate. Dynamic reduction of the data by summarization, subset selection, and more sophisticated dynamic pattern identification methods will be necessary to reduce the volume of data. And the reduced data volume will have to be stored at the same rate as it is generated, in order for the exascale computation to progress without interruption.

This requirement presents new challenges of orchestrating data movement from the supercomputer to the local and remote storage systems. Data distribution will have to be integrated into the data generation phase. Managing the dataflow using well-coordinated workflow engines will be required as part of the software infrastructure that runs the simulations.

The issue of large-scale data movement will become more acute as very large datasets or subsets are shared by large scientific communities. This situation will require large volumes of data to be replicated or moved between production and analysis machines, often across the wide area. While networking technology is greatly improving with the introduction of optical connectivity, the transmission of large volumes of data will inevitably encounter transient failures, and automatic recovery tools will be necessary.

Another fundamental requirement is the automatic allocation, use, and release of storage space. Replicated data cannot be left in storage devices unchecked, or storage systems will fill and become clogged. A new paradigm of attaching a lifetime to replicated datasets, and the automatic management of data whose lifetime expires, will be essential.

3.1 Parallel File Systems

Parallel file systems such as Lustre and PVFS2, and I/O software stacks including MPI-IO and high-level I/O libraries (e.g. HDF5, Parallel netCDF) are in extensive use in HPC by a wide variety of applications. Current deployments typically use vendor file systems and enterprise hardware and are providing adequate storage performance, capacity, and reliability for current systems. For the next decade the key challenges to Parallel File Systems are scaling, performance, and fault tolerance. Overall, we need storage systems at HPC centers that provide scalable bandwidth and tolerate non-catastrophic failures without data loss.

3.2 Data Management

Scientists are facing the burden of managing the data generated by large-scale simulations and experiments. They need to deal with multiple steps of moving the data between software modules, extracting subsets of the data, summarizing the data, generating images or movies, and moving the data to archival storage. Such tasks are extremely time consuming, and require expertise that is irrelevant to the scientist, such as transfer protocols, security mechanisms, and idiosyncrasies of archival systems.

In order to support exascale data generation, data storage will fundamentally change. Users will need tools that manage the movement of data automatically across a storage hierarchy. Data that is used often will be moved to highly parallel dynamic storage, while archived data will reside in powered down storage or passive storage devices. Furthermore, algorithms to automatically track and remove unused data from the dynamic storage will be essential to minimize storage costs. Collections of datasets will be organized as directories. Such abstraction will fundamentally change the way the I/O is expressed by applications and will involve a storage management layer that maps datasets into physical devices without effecting the applications.

Keeping track of the data generated is already a daunting task. The meaning of the data, referred to as metadata, requires precise annotation of how the data was generated, and the scientific interpretation of each data item. Furthermore, many scientific datasets are generated from other datasets, or perhaps a combination of datasets. This requires the capability of tracking the history, or provenance, of the data. Today, such tools are provided in ad hoc manner; some metadata is collected in various forms of notebooks, some in databases, and some embedded as headers of files. In the exascale regime the automation of this task is essential because of the sheer volume of the data and the accelerated rate of their production. Standard metadata models and tools will have to be developed, as well as tools to automatically capture the metadata as the datasets are generated. Furthermore, the data models need to support standard ontology for each scientific domain and allow for dynamic evolution of such standards.

3.3 Turning Data into Scientific Discoveries

One of the challenges in contemporary science is the process of discovering knowledge and testing hypotheses in the presence of a growing deluge of data. A recurring theme in this document—that existing methods will not scale to meet the challenges of exascale systems and data—holds true in the area of knowledge discovery. Existing approaches for knowledge discovery do not scale to the exascale. Failure to address the issues of knowledge discovery in the exascale ecosystem will have a profound and adverse impact on all science programs.

A number of different, yet complementary, approaches to address these problems will require exploration:

- Ability to visualize and analyze results at coarse and fine resolutions depending to support the natural investigatory process that relies on context/focus interaction;
- Better visual data analysis algorithms for characterizing and presenting uncertainty;
- Integration of visual data presentation and data analysis techniques (e.g., clustering, classification, statistical analysis and representation) to aid in accelerating knowledge discovery;
- Greater emphasis on the human-computer interface to increase the efficacy of visual presentation motifs and interactive knowledge discovery interaction models;
- Context-centric interfaces to simplify use of complex software infrastructure;
- Rethinking design and implementation of fundamental knowledge discovery algorithms and software infrastructure to be capable of effectively leveraging exascale platforms.

As the size of simulation, observational, and experimental datasets grow into the petascale range, many of the existing technologies do not scale to be practical for both on-line and off-line data analysis and knowledge discovery processes. These additional challenges need to take advantage of acceleration, parallel processing, and smart navigation, summarization, and manipulations of the massive datasets. New methods for achieving better efficiency of searching, processing, exploring, and displaying information are needed. Finally, scalable and flexible data formats for storing, processing, provenance, and sharing results of data analysis are required.

3.4 Efficient Searching

Searching for key pieces of information in data is becoming challenging due to several factors. With data reaching the petabyte scale, there is a need for better indexing technology to support multiple tasks such as database search, sub-graph extraction, and text searches with ranking. The data being searched is also becoming more complex, with simple row/column tables being replaced by graphs, data with associated uncertainty, collections of data such as a sequence of interactions in a graph, and so on. Users are also making more complex queries and may require an estimate of the time it would take to obtain an answer to the query. To address these issues, we need advances in several different areas including, but not restricted to, indexing, sampling, query estimation, and approximate query answering. The characteristics of modern datasets, as well as the hardware on which the analysis software is executed, suggest the need to re-think existing algorithms or develop new ones due to:

***Scalability of the analysis techniques**:* We need advances in both mathematical algorithms and computer science issues to ensure that our analysis techniques will scale with both the size of the data and the number of processors available to run the analysis algorithms. This would require new parallel algorithms, scalable data structures, techniques for re-organizing the data to be more suitable for multiple processors, automatic compiler-driven parallelization, etc. Since data maybe inherently distributed and streamlined, algorithms need to be adapted to these physical properties of the data.

***Modifying algorithms for new architectures**:* With the paradigm shift from single processors to multi-core architectures, GPUs, and FPGAs, we need research to determine how scientific data analysis tasks can be re-designed to be highly multi-threaded to take advantage of these architectures. In particular, I/O bottlenecks often encountered by data intensive applications can be circumvented with in-memory data operations and specialized indexing techniques such as Quaterrnary Triangular Mesh (for geoprocessing), and space filling curves (for increasing locality in multidimensional spaces). The new architectures can be particularly well suited for some of the newer types of data. For example, algorithms for fast quantile and frequency estimation in data streams can benefit from the use of GPUs. Likewise, significant amount of processing tasks may be accelerated using FPGAs, e.g., kernel computations, key statistics, pattern recognition using templates etc.

Analysis within storage: An approach to minimizing the time taken to move data from storage to where it is being analyzed is to analyze the data where it is in storage. This is referred to as Active Storage. Research is needed to understand the data structures necessary for such analysis and the approaches including programming models, software libraries used to embed analysis functions within storage, and the storage infrastructure enhancements necessary to make this possible.

Exploiting modern programming models and constructs: MapReduce, Bigtable, and have been successfully used in various applications on several different architectures for the analysis of large datasets. However, it is unclear if such programming models can be directly used in the context of scientific data. Research is needed to determine how such models can be extended to implement scientific data analysis algorithms and meet the requirements of fault tolerance and scalability, while supporting the fine granularity and frequent synchronization needs of scientific applications.

4. Software Sustainability

Creating an exascale software ecosystem entails more than just solving the technical challenges. It includes educating scientists on how to use the solutions, both new tools and new approaches, and demonstrating why using these solutions is to their advantage. It includes making sure that the solutions are hardened to production quality so that they can be integrated into the software suites of the nation's supercomputer centers. It includes making pieces available as they are completed, rather than waiting until everything is done. And it includes helping users integrate these pieces into existing codes so that science teams can benefit in the near term and build up trust in the solutions being provided for the exascale software ecosystem.

Sustaining and hardening software to production quality: Academic and laboratory researchers and developers rarely possess either the software engineering skills or the desire to transition research ideas to production code, with concomitant support. The pathway from research prototype to a software tool that is widely available, production quality and actively supported is not clear. In most cases, the funding researchers receive is targeted toward specific research goals, and not necessarily to provide tool porting, testing, documentation, standardization, or user support. A new model of software tool support is needed if we are to address current and future needs.

Engagement with applications and domain experts: All too often, software tools are developed in the absence of detailed understanding of the user and application needs. Conversely, users are often unaware of the technical difficulties underlying tool design and support. Bridging this gap with a collaborative software development and extension process, where promising ideas are identified and tested early, then enhanced and supported across the application development and support cycle, would ameliorate the expectations gap.
User training: Software development tools can be very flexible and powerful in their own right. The developers of these tools should make it a priority to train the user community on tool capabilities and usage. Furthermore, usability should be a major requirement included in any funding focusing on transition to production software.
Education and workforce: As is the case in other areas of HPC and computer science, there is a specific need to educate new students and workers in order to ensure a sufficiently large and capable workforce.

6. References

1. Final report from Exascale townhall meetings- Breakout Group Seven "Software Challenges". June 2007

2. Workshop on Software Development Tools for Petascale Computing final report. August 2007

3. Workshop on Visual Analysis and Data Exploration at Extreme Scale final report, October 2007,

4. Scalable Systems Software Summary Report ASCR PI meeting, April 2008

5. Data Management and Analysis Summary Report ASCR PI meeting, April 2008

6. Workshop on Mathematics for Analysis of Petascale Data final report, June 2008

7. Whitepaper "The Scientific Data Analysis Process at the Petascale" Editors: Chandrika Kamath, Arie Shoshani, August 2008

8. Workshop on CS/Math Institutes and High Risk/High Payoff Technologies for Applications preliminary report, October 2008

9. DARPA "ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems", Kogge, et.al. (September 2008) http://www.cse.nd.edu/Reports/2008/TR-2008-13.pdf

10. DARPA "Exascale Software study", Sarkar, et.al., (In preparation) http://www.lbl.gov/CS/html/SC08ExascalePowerWorkshop/Sarkar-SC08-Exascale-Workshop-v2.pdf