Making a Case for Distributed File Systems at Exascale

[1,2]Ioan Raicu, [2,3,4]Ian T. Foster, [2,3,5]Pete Beckman

[1] Department of Computer Science, Illinois Institute of Technology, Chicago, IL, USA
[2] Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, USA
[3] Computation Institute, University of Chicago, Chicago, IL, USA
[4] Department of Computer Science, University of Chicago, Chicago, IL, USA
[5] Exascale Technology and Computing Institute, Argonne National Laboratory, Argonne, IL, USA

iraicu@cs.iit.edu, foster@anl.gov, beckman@anl.gov

ABSTRACT

Exascale computers will enable the unraveling of significant scientific mysteries. Predictions are that 2019 will be the year of exascale, with millions of compute nodes and billions of threads of execution. The current architecture of high-end computing systems is decades-old and has persisted as we scaled from gigascales to petascales. In this architecture, storage is completely segregated from the compute resources and are connected via a network interconnect. This approach will not scale several orders of magnitude in terms of concurrency and throughput, and will thus prevent the move from petascale to exascale. At exascale, basic functionality at high concurrency levels will suffer poor performance, and combined with system mean-time-to-failure in hours, will lead to a performance collapse for large-scale heroic applications. Storage has the potential to be the Achilles heel of exascale systems. We propose that future high-end computing systems be designed with non-volatile memory on every compute node, allowing every compute node to actively participate in the metadata and data management and leveraging many-core processors high bisection bandwidth in torus networks. This position paper discusses this revolutionary new distributed storage architecture that will make exascale computing more tractable, touching virtually all disciplines in high-end computing and fueling scientific discovery.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]; C.5.1 [**Large and Medium (``Mainframe'') Computers**]; D.4.2 [**Storage Management**]; D.4.3 [**File Systems Management**]; D.4.5 [**Reliability**]; D.4.7 [**Organization and Design**]; D.4.8 [**Performance**]; H.3.4 [**Systems and Software**]

General Terms

Management, Performance, Reliability

Keywords

Exascale computing, distributed file systems, storage architecture, many-task computing

LSAP'11, June 8, 2011, San Jose, California, USA.

1. INTRODUCTION

Today's science is generating datasets that are increasing exponentially in both complexity and volume, making their analysis, archival, and sharing one of the grand challenges of the 21st century. Seymour Cray once said – "*a supercomputer is a device for turning compute-bound problems into I/O-bound problems*" – which addresses the fundamental shift in bottlenecks as supercomputers gain more parallelism at exponential rates [1], the storage infrastructure performance is increasing at a significantly lower rate. This implies that the data management and data flow between the storage and compute resources is becoming the new bottleneck for large-scale applications. The support for data intensive computing [2] is critical to advancing modern science as storage systems have experienced a gap between capacity and bandwidth that increased more than 10-fold over the last decade. There is an emerging need for advanced techniques to manipulate, visualize and interpret large datasets. Many domains (e.g. astronomy, medicine, bioinformatics) share these data management challenges, strengthening the potential impact from generic solutions.

Exascale computers (e.g. capable of 10^{18} ops/sec) [3], with a processing capability similar to that of the human brain, will enable the unraveling of significant scientific mysteries and present new challenges and opportunities. The US President made the building of exascale systems a top national priority, stating [4] that it will "*dramatically increasing our ability to understand the world around us through simulation and slashing the time needed to design complex products such as therapeutics, advanced materials, and highly-efficient autos and aircraft*". Major scientific opportunities arise in many fields (such as weather modeling, understanding global warming, national security, drug discovery, and economics) and may rely on revolutionary advances that will enable exascale computing. [1] Many experts predict [3] that exascale computing will be common by 2019: millions of nodes, billions of threads of execution, hundreds of petabytes of memory, and exabytes of persistent storage.

The current architecture of high-end computing (HEC) systems is decades-old and has persisted as we scaled from gigascales to petascales. In this architecture, storage is completely segregated from the compute resources and are connected via a network interconnect (e.g. parallel file systems running on network attached storage, such as GPFS [5], PVFS [6], and Lustre [7]). This approach will not scale several orders of magnitude in terms of concurrency and throughput, and will thus prevent the move from petascales to exascale. Unless significant research is invested to revolutionize the storage hardware architecture and parallel/distributed file system implementations, we will not be able to build *capability* exascale systems. Supercomputers are

generally designed for either *capability* or *capacity* computing, as Irving Wladawsky-Berger wrote in 2010.

> "*Capability supercomputers dedicate the whole machine to solve a very large problem in the shortest amount of time. Capacity supercomputers, on the other hand, support large numbers of users solving different kinds of problems simultaneously. While both kinds of supercomputing are very important, initiatives designed to push the envelope, like DOE's exascale project, tend to focus on the development of capability machines to address Grand Challenge problems that could not be solved in any other way. Capability computing has been primarily applied to what is sometimes referred to as heroic computations, where just about the whole machine is applied to a single task.*" [8]

2. ACHILLES HEEL

We believe storage systems in future exascale systems will be its Achilles heel [9]), unless storage systems are re-architected to ensure scalability to millions of nodes and potentially billions of concurrent I/O requests. We believe that the system mean-time-to-failure (MTTF) at exascale will be of the order of a few hours. However, based on current trends, even basic functionality (e.g. booting - Figure **1**, metadata operations - Figure 2, read/write operations - Figure 3 and Figure 4, loading applications - Figure 3, and check-pointing - Figure 5) at significant concurrency levels are expected to take more time than the expected exascale machine's MTTF, which will result in complete performance collapse for grand challenge applications.

Booting: For example, booting the Blue Gene/P (see Figure 1) is an expensive operation at 0.5 petaflops, as measured on the real machine. On 256 processors, it takes 85 seconds to boot the allocation; on the full 160K processors, it takes 1090 seconds. Unfortunately, it appears that the machine boot time grows linearly with the number of nodes, translating to potentially over 25K seconds (7+ hours) boot-time at 1M node scales. [10]

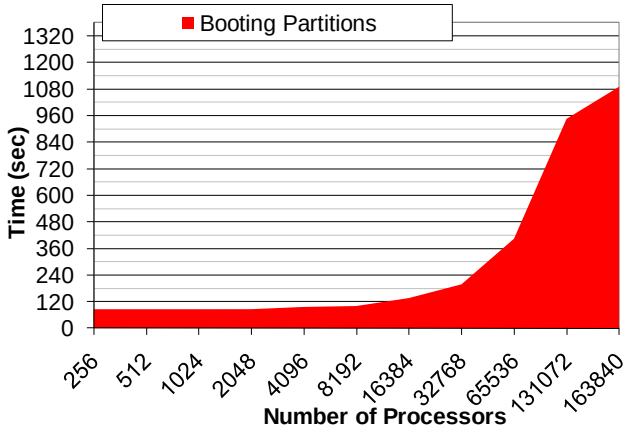

Figure 1: Booting a BlueGene/P; 160K-cores [10]

Metadata operations on parallel file systems can be inefficient at large scale. Early experiments on the BlueGene/P system at 16K-core scales (see Figure 2) shows the various costs (wall-clock time measured at remote processor) for file/directory create on GPFS. Ideal performance would be to have a flat line. If care is not taken to avoid lock contention, performance degrades rapidly, with operations (e.g. create directory) that took milliseconds on a single core, taking over 1000 seconds at 16K-core scales. [10, 11]

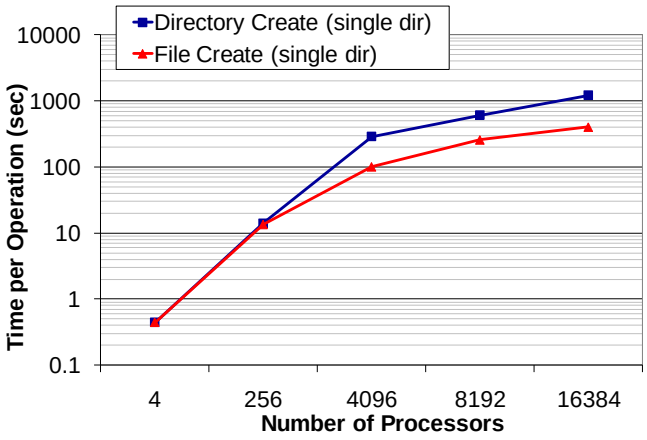

Figure 2: File/dir create time on GPFS; 16K-cores [10]

Read/Write: Reading performance of common datasets (e.g. application binaries, read-only databases) is challenging. The experiment in Figure 3 shows the distribution of data from the GPFS file system to the compute nodes with two approaches: 1) pushing out the data over a spanning tree (CIO), and 2) pulling the data from each compute node independently (GPFS). At the relatively modest scale of 4k-cores, the CIO approach outperforms GPFS by a factor of five, due to better utilization of the bi-section bandwidth of the torus network. Writing from many compute nodes directly to parallel file systems is also challenging; Figure 4 shows the poor efficiency achieved (15%-70%) with 16 second tasks producing 1KB to 1MB output. [10, 11]

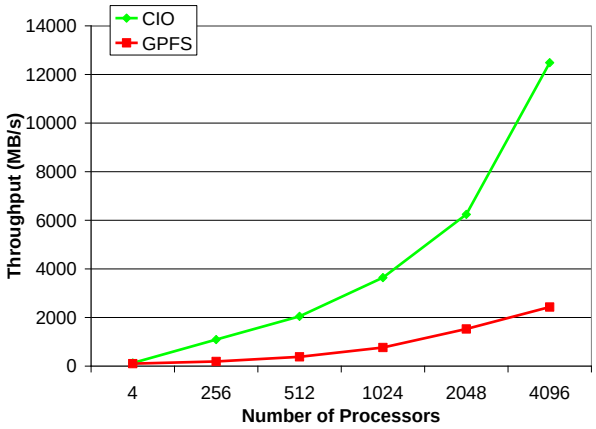

Figure 3: Data read; GPFS vs. spanning tree (CIO) [11]

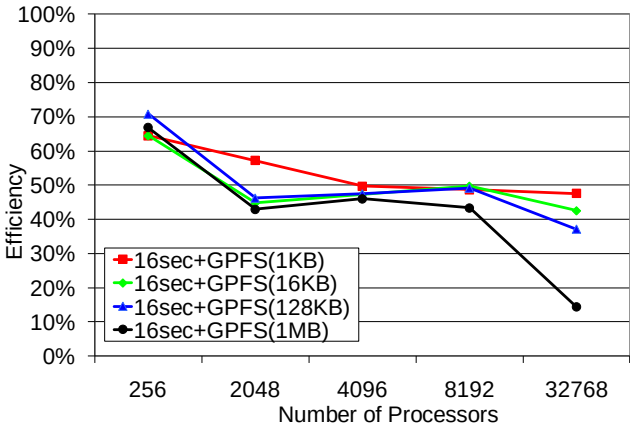

Figure 4: GPFS write efficiency; 16sec tasks, 32K-cores [11]

Checkpointing is the state-of-the-art technique for delivering fault tolerance for high-performance computing (HPC) on large-scale systems. It has been shown that writing the state of a process to persistent storage is the largest contributor to the performance overhead of checkpointing [12]. Let's assume that parallel file systems are expected to continue to improve linearly in throughput with the growing number of nodes (an optimistic assumption); however, note that per node memory (the amount of state that is to be saved) will likely continue to grow exponentially. Assume the MTTF is modeled after the BlueGene with a claimed 1000 years MTTF per node (another optimistic assumption). Figure 5 shows the expected MTTF of about 7 days for a 0.8 petaflop IBM BlueGene/P with 64K nodes, while the checkpointing overhead is about 34 minutes; at 1M nodes (~1 exaflop), the MTTF would be only 8.4 hours, while the checkpointing overhead would be over 9 hours.

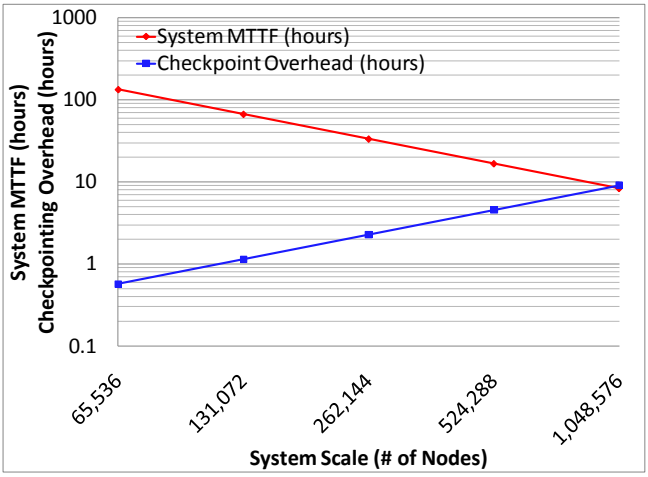

Figure 5: Expected checkpointing cost and MTTF towards exascale

Simulations results (see Figure 6) from 1K nodes to 2M nodes (simulating HEC from 2000 to 2019) show the application uptime collapse for *capability* HEC systems. Today, 20% or more of the computing capacity in a large HPC system is wasted due to failures and recoveries [12], which is confirmed by the simulation results.

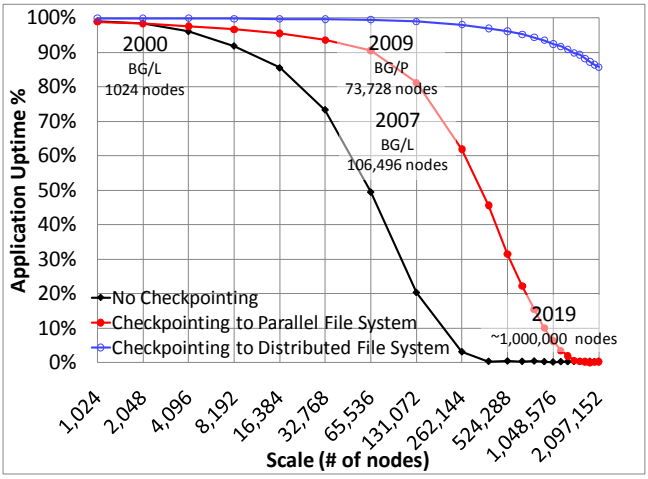

Figure 6: Simulation application uptime towards exascale

The distributed file system shown in the simulation results is a hypothetical file system that could scale near linearly by preserving the data locality in checkpointing. The application requirements are modeled to include 7 days of computing on the entire HEC system. By 1M nodes, checkpointing with parallel file systems (red line with circle symbol) has a complete performance collapse (as was predicted from Figure 5). Note that the hypothetical distributed file system (blue line with hollow circle symbol) could still have 90%+ uptime even at exascale levels.

3. RADICAL NEW VISION

We believe the HEC community needs to develop both the theoretical and practical aspects of building efficient and scalable distributed storage for HEC systems that will scale four orders of magnitude in concurrency. We believe that emerging distributed file systems could co-exist with existing parallel file systems, and could be optimized to support a subset of workloads critical for HPC and MTC workloads at exascale. There have been other distributed file systems proposed in the past, such as Google's GFS [13] and Yahoo's HDFS [14]; however these have not been widely adopted in HEC due to the workloads, data access patterns, and supported interfaces (POSIX [15]) being quite different. Current file systems lack scalable distributed metadata management, and well defined interfaces to expose data locality for general computing frameworks that are not constrained by the map-reduce model to allow data-aware job scheduling with batch schedulers (e.g. PBS [16], SGE [17], Condor [18], Falkon [19]).

We believe that future HEC systems should be designed with non-volatile memory (e.g. solid state memory [20], phase change memory [21]) on every compute node (Figure 7); every compute node would actively participate in the metadata and data management, leveraging the abundance of computational power many-core processors will have and the many orders of magnitude higher bisection bandwidth in multi-dimensional torus networks as compared to available cost effective bandwidth into remote network persistent storage. This shift in large-scale storage architecture design will lead to improving application performance and scalability for some of the most demanding applications. This shift in design is controversial as it requires storage architectures in HEC to be redefined from their traditional architectures of the past several decades. This new approach was not feasible up to recently due to the unreliable spinning disk technology [22] that has dominated the persistent storage space since the dawn of supercomputing. However, the advancements in solid-state memory (with MTTF of over two million hours [23]) have opened up opportunities to rethink storage systems for HEC, distributing storage on every compute node, without sacrificing node reliability or power consumption. The benefits of this new architecture lies in its enabling of some workloads to scale near-linearly with systems scales by leveraging data locality and the full network bisection bandwidth.

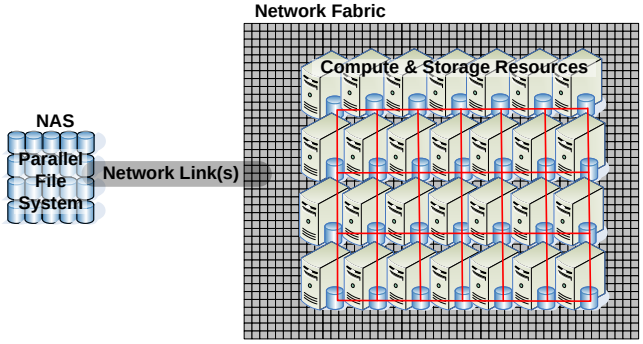

Figure 7: Proposed storage architecture with persistent local storage on every compute node

Future storage systems should support distributed metadata management, leveraging distributed data-structure tailored for HEC. The distributed data-structures share some characteristics with structured distributed hash tables [24], having resilience in face of failures with high availability; however, they should support constant time inserts/lookups/removes delivering low latencies typically found in centralized metadata management. The data should be partitioned into chunks and be spread out over many storage nodes, based on the data access patterns to maximize data locality. Replication [25] would be used to ensure data availability, and cooperative caching [26] would deliver high aggregate throughput. Data would be indexed, by including descriptive, provenance, and system metadata with each file. Various data access semantic should be used, from POSIX-like interfaces for generality (e.g FUSE [27]), to relaxed semantics (e.g. eventual consistency on data modifications [28], write-once read-many [29]) to avoid consistency issues and increase scalability.

We are working on delivering exactly such a distributed file system, name FusionFS [30]. FusionFS is a new distributed filesystem that will co-exist with current parallel filesystems in High-End Computing, and it will be optimized for both a subset of HPC and Many-Task Computing workloads. FusionFS is a user-level filesystem with a POSIX-like interface through FUSE [27] that runs on the compute resource infrastructure, and enables every compute node to actively participate in the metadata and data management. Distributed metadata management is implemented using ZHT [31], a zero-hop distributed hashtable. ZHT has been tuned for the specific requirements of high-end computing. The data is partitioned and spread out over many nodes based on the data access patterns. Replication and cooperative caching are used to ensure data availability and high throughput. Both FusionFS and ZHT are active projects attempting to implement the radical new vision sought after by this position paper.

4. A SURVEY OF THE LITERATURE

There has been a significant amount of research work over the last several decades on how to build scalable storage systems, namely on distributed hash tables, shared/parallel/distributed file systems, and support for data-intensive distributed computing.

Distributed Hash Tables: There have been many distributed hash table (DHT) algorithms and implementations proposed over the years. We discuss DHTs in this section due to their important role in building support for scalable metadata across extreme scale systems. Some of the DHTs from the literature are Kademlia [32], CAN [33], Chord [34], Pastry [35], Tapestry [36], Memcached [37], Dynamo [38], Cycloid [39], Ketama [40], RIAK [41], Maidsafe-dht [42], and C-MPI [43]. Most of these DHTs scale logarithmically with system scales, but some (e.g. Cycloid) go as far as reducing the number of operations to O(c) where c is a constant related to the maximum size of the network (instead of the actual size of the network), which in practice still results to c ~ log(n). It is important to point out that several key features of traditional DHTs are not necessary in HEC, and that if some simplifying assumptions could potentially reduce the number of operations needed per insert/lookup down to a small constant (1 on average). *Some of the emphasized key features of HEC are: trustworthy/reliable hardware, fast network interconnects, non-existent node "churn", the requirement for low latencies, and scientific computing data-access patterns.* Most HEC environments are batch oriented, which implies that a system that is configured at run time, generally has information about the compute and storage resources that will be available. This means that the amount of resources (e.g. number of nodes) would not increase or decrease dynamically, and the only reason to decrease the allocation is either to handle failed nodes, or to terminate the allocation. By making dynamic membership optional, the complexity of the system can be reduced and a low average number of hops per operation can be achieved. Furthermore, nodes in HEC are generally reliable and have predicable uptime (nodes start on allocation, and nodes shut down on de-allocation). This implies that node "churn" in HEC is virtually non-existent, a property of HEC that should be leveraged to create a more optimized distributed data structure. It is also important to point out that nodes in a HEC system are generally trust-worthy, and that stringent requirements to encrypt communication and/or data would simply be adding overheads. HEC systems are generally locked down from the outside world, behind login nodes and firewalls, and although authentication and authorization is still needed, full communication encryption is wasteful for a large class of scientific computing applications that run on many HEC systems. Most parallel file systems used in HEC communicate between the client nodes and storage servers without any encryption. There has been some uptake recently in using traditional DHTs in HEC, namely the C-MPI [43] project, in which the Kademlia DHT has been implemented and shown to run well on 1000 nodes on a BlueGene/P supercomputer. C-MPI is used to perform data management operations for the Swift project [44], but it does not operate at the filesystem level and does not attempt to optimize Kademlia for HEC. Another recent project using DHTs on a HEC is DataSpaces [45], which deploys a DHT on a Cray XT5 to coordinate in-memory data management for simulation workflows. DataSpaces fails to optimize the DHT for HEC, does not decouple the metadata from the data causing poor data locality, and does not expose the work as a file system.

Shared, Parallel, and Distributed File Systems: There have been many shared and parallel file systems proposed since the 1980s, such as the Network File System (NFS) [46], Andrew File System (AFS) [47], General Purpose File System (GPFS) [13], Parallel Virtual File System (PVFS) [6], Lustre [7], Panases [48], Microsoft's Distributed File System (DFS) [49], GlusterFS [50], OneFS [51], POHMELFS [52], and XtreemFS [53]. While the majority of these file systems expose a POSIX-like interface providing a global namespace, and many have been adopted in cluster computing, grid computing, and even supercomputing, the biggest critique of these file systems is their vision that compute resources should be completely agnostic of the data locality on the underlying storage system. All of these file systems assume that the storage nodes/servers are significantly fewer than the client/compute nodes that will access the file system, resulting in an unbalanced architecture for data-intensive workloads. A variety of distributed file systems have been developed to address this unbalance from parallel file systems to support data-intensive computing, such as GFS [13], HDFS [14], Sector [54], CloudStore [55], Ceph [56], GFarm [57, 58], MooseFS [59], Chirp [60], MosaStore [61], PAST [62], Circle [63], and RAMCloud [64]. However, many of these file systems are tightly coupled with execution frameworks (e.g. MapReduce [65], Hadoop [14]), which means that scientific applications not using these frameworks must be modified to use these underlying non-POSIX-compliant file systems. For those that offer a POSIX-like interface, they lack distributed metadata management. And for those few (e.g. Ceph, PAST, Circle) that also have distributed metadata management, they fail to decouple data and metadata management making maximizing data locality difficult. The

majority of these systems also fail to expose the data locality information for general computational frameworks (e.g. batch schedulers) to harness the data locality through data-aware scheduling. Also, with the exception of RAMCloud, none of the filesystems were designed and optimized for solid state memory. It is worth noting that the majority of these distributed file systems were not designed specifically for HEC and scientific computing workloads, and the scales that HEC are anticipating in the coming years and are at a significant disadvantage.

Storage Systems for Data Intensive Computing: Over the past decade, considerable work has been done on data management in cluster and grid computing [19, 66 - 79]. All these projects explore a similar space of how to support data-intensive distributed scientific computing applications, and many draw similar conclusions that data locality is critical to obtaining good performance. However, none of this work was done at the file system level, and certainly none of the work addressed the scales this work addresses.

NSF Funded Projects: There has also been a large number of NSF funded projects whose focus closely aligns with the proposed work, ranging from CAREER to HECURA grants, with many of these projects falling under HEC FSIO [80]. Some of these proposals [81, 82] addressed petascale and exascale systems, but didn't focus on the storage challenges. Others projects [83 - 87] focused on supporting data-intensive distributed computing, but didn't necessarily expose the data management with POSIX interfaces, did not expose data locality for general consumption by scheduling systems, did not address distributed metadata management, and did not aim their solutions to exascale. More promising work was in active storage [88 - 90] where data locality is emphasized, however the work did not aim for HEC at exascale. More traditional work in parallel I/O and filesystems was proposed in various NSF HECURA proposals [91 - 95], but these approaches have the inherent limitations outlined for parallel file systems, namely the segregation of compute resource from storage resources. Others focused on metadata management [96, 97], but emphasized more on the organization and search-ability of metadata, rather than the scalable construction and maintainability of the metadata at scales of millions of nodes. One of the projects [98] specifically explored the use of non-volatile memory to improve the I/O performance of HEC, supporting the PI's claims that non-volatile memory is a critical next step in the evolution of future HEC. Another project [99] identified non-volatile memory to be critical in scaling distributed databases for scientific computing in astronomy workloads. Two projects [100, 101] focused on optimizing for small I/O access patterns in parallel file systems, an access pattern that this work also addresses. Other work [102] focused on storage redundancy and reliability, while another [103] focused on programming models for data intensive computing. There are likely more relevant projects, but in the end, they all distil to a simple summary, extracted from the latest HEC FSIO 2008 Workshop Report [80] (the following italics texts are excerpts from the report):

> *Data path today is the same as the data path 20 years ago.* There is a need for *new technologies that will offer greater scalability for file system metadata. Until recently, I/O stacks and architectures have been static forcing developers to adopt awkward solutions in order to achieve target I/O rates. Studies into methods to deal with small, unaligned I/O and mixed-size I/O workloads as well as collaborative caching are also needed. Novel approaches to I/O and file systems also need to be explored including redistribution of intelligence, user space file systems, data-aware file systems, and the use of novel storage devices. Active disks have been an active research topic, but there has been no good interface and set of semantic rules has come along that would make it generally useful.* Most, if not all progress to date in parallel file systems in HEC has been *evolutionary; what is lacking is revolutionary research; no fundamental solutions are being proposed*. *Revolutionary I/O technologies developments (both software and hardware) are needed to address the growing technology gap.*

5. LONG TERM IMPACT

The ideas in this position paper are transformative due to their departure from traditional HEC architectures and approaches, while proposing radical storage architecture changes based on distributed file systems to make exascale computing a reality. This paper addresses fundamental technical challenges that will become increasingly harder to address with existing solutions due to a declining MTTF of future HEC systems.

This work will open doors for novel research in programming paradigm shifts (e.g. Many-Task Computing [11, 29, 104 - 106]) needed as we approach exascale. Many-Task Computing aims to bridge the gap between two computing paradigms, high-throughput computing and high-performance computing, generally producing both compute-intensive and data-intensive workloads, and has been shown to contain a large set of scientific computing applications from many domains. Some of the challenges in supporting MTC at scale involving metadata and read/write operations can be seen in Figure 2 - 4. More challenges and solutions have been identified in prior work [10, 11, 19, 44, 72, 79, 107 - 109].

The controversial viewpoints of this paper can make exascale computing more tractable, touching every branch of computing in HEC. They will extend the knowledgebase beyond exascale systems into commodity systems as the fastest supercomputers generally become the mainstream computing system in less than a decade; the solutions proposed here will be relevant to the data centers and cloud [110] infrastructures of tomorrow. These advancements will impact scientific discovery and global economic development. They will also strengthen a wide range of research activities enabling efficient access, processing, storage, and sharing of valuable scientific data from many disciplines (e.g. medicine, astronomy, bioinformatics, chemistry, aeronautics, analytics, economics, and new emerging computational areas in humanities, arts, and education). If these ideas materialize, they will *revolutionize* the storage systems of future HEC systems, and open the door to a much broader class of applications that would have normally not been tractable. Furthermore, the concepts, data-structures, algorithms, and implementations that underpin these ideas in resource management at the largest scales can be applied to new emerging paradigms, such as Cloud Computing.

Our main message is that by combining lessons learned from parallel file systems and distributed file systems, along with new advances in hardware (e.g. solid state memory), we can define a new storage architecture that is optimized for future high-end computing at exascale and has the potential to deliver a viable storage architecture for future extreme scale high-end computing. The position of this paper is revolutionary as it breaks the accepted practice of segregating storage resource from computational resources, and leveraging the abundance of processing power, bisection bandwidth, and local I/O commonly found in future high-end computing systems.

6. ACKNOWLEDGMENTS

This work was supported in part by the U.S. Dept. of Energy under Contract DE-AC02-06CH11357, as well as the National Science Foundation grant NSF-0937060 CIF-72 and NSF-1054974. We want to thank our collaborators for the valuable help, feedback, and insight leading up to this work, namely Mike Wilde, Matei Ripeanu, Arthur Barney Maccabe, Marc Snir, Rob Ross, Kamil Iskra, and Alok Choudhary. We also want to thank the anonymous reviewers whose feedback was invaluable to improving the clarity of the paper.

7. REFERENCES

[1] Top500 Supercomputer Sites, Performance Development, http://www.top500.org/lists/2010/11/performance_development, November 2010

[2] T. Hey, S. Tansley, and K. Tolle. The Fourth Paradigm: Data-Intensive Scientific Discovery, Microsoft Research 2009

[3] V. Sarkar, et al. "ExaScale Software Study: Software Challenges in Extreme Scale Systems", ExaScale Computing Study, DARPA IPTO, 2009

[4] B. Obama. "A Strategy for American Innovation: Driving Towards Sustainable Growth and Quality Jobs", National Economic Council, 2009

[5] F. Schmuck, R. Haskin, "GPFS: A Shared-Disk File System for Large Computing Clusters," FAST 2002

[6] P. H. Carns, W. B. Ligon III, R. B. Ross, R. Thakur. "PVFS: A parallel file system for linux clusters", Proceedings of the 4th Annual Linux Showcase and Conference, 2000

[7] P. Schwan. "Lustre: Building a file system for 1000-node clusters," Proc. of the 2003 Linux Symposium, 2003

[8] I. Wladawsky-Berger. "Opinion - Challenges to exascale computing", International Science Grid this Week, April 2010; http://www.isgtw.org/feature/opinion-challenges-exascale-computing

[9] Wikipedia contributors. "Achilles' heel." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 8 Jun. 2010. Web. 29 Jun. 2010

[10] I. Raicu, Z. Zhang, M. Wilde, I. Foster, P. Beckman, K. Iskra, B. Clifford. "Toward Loosely Coupled Programming on Petascale Systems," IEEE SC 2008

[11] Z. Zhang, A. Espinosa, K. Iskra, I. Raicu, I. Foster, M. Wilde. "Design and Evaluation of a Collective I/O Model for Loosely-coupled Petascale Programming", IEEE MTAGS08, 2008

[12] E.N. Mootaz Elnozahy, et al. "System Resilience at Extreme Scale", Defense Advanced Research Project Agency (DARPA), 2007

[13] S. Ghemawat, H. Gobioff, S.T. Leung. "The Google file system," 19th ACM SOSP, 2003

[14] A. Bialecki, M. Cafarella, D. Cutting, O. O'Malley. "Hadoop: A Framework for Running Applications on Large Clusters Built of Commodity Hardware", 2005

[15] J.S. Quarterman, S. Wilhelm, "UNIX, POSIX, and Open Systems: The Open Standards Puzzle", Addison-Wesley, Reading, MA, 1993

[16] B. Bode, et al. "The Portable Batch Scheduler and the Maui Scheduler on Linux Clusters", Usenix, 4th Annual Linux Showcase & Conference, 2000

[17] W. Gentzsch. "Sun Grid Engine: Towards Creating a Compute Power Grid," IEEE CCGrid, 2001

[18] D. Thain, T. Tannenbaum, M. Livny, "Distributed Computing in Practice: The Condor Experience", Concurrency and Computation: Practice and Experience, 2005

[19] I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, M. Wilde. "Falkon: A Fast and Light-weight tasK executiON Framework," IEEE/ACM SC 2007

[20] A. Birrell, M. Isard, C. Thacker, T. Wobber. "A design for high-performance flash disks", Operating Systems Review, 41(2):88–93, 2007

[21] Intel News Release. "Intel, STMicroelectronics deliver industry's first phase change memory prototypes", http://www.intel.com/pressroom/archive/releases/20080206corp.htm, 2008

[22] W. Jiang, C. Hu, Y. Zhou, A. Kanevsky. "Are disks the dominant contributor for storage failures? A comprehensive study of storage subsystem failure characteristics", In USENIX Conference on File and Storage Technologies (FAST), pages 111–125, 2008

[23] Intel Product Manual. "Intel® X25-E SATA Solid State Drive", http://download.intel.com/design/flash/nand/extreme/319984.pdf, 2009

[24] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, I. Stoica. "Looking up data in P2P systems", Communications of the ACM, 46(2):43–48, 2003

[25] H. Stockinger, A. Samar, B. Allcock, I. Foster, K. Holtman, B. Tierney, "File and object replication in data grids," ACM HPDC-10, 2001

[26] S. Podlipnig, L. Böszörmenyi. "A survey of Web cache replacement strategies", ACM Computing Surveys (CSUR), Volume 35 , Issue 4, Pages: 374 – 398, 2003

[27] "Filesystem in Userspace", http://fuse.sourceforge.net/, 2011

[28] W. Vogels, "Eventually consistent," ACM Queue, 2008.

[29] I. Raicu, I. Foster, Y. Zhao. "Many-Task Computing for Grids and Supercomputers", IEEE MTAGS08, 2008

[30] FusionFS: Fusion distributed File System, http://datasys.cs.iit.edu/projects/FusionFS/index.html, 2011

[31] ZHT: Zero-Hop Distributed Hash Table for High-End Computing, http://datasys.cs.iit.edu/projects/ZHT/index.html, 2011

[32] P. Maymounkov, D. Mazieres. "Kademlia: A Peer-to-peer Information System Based on the XOR Metric", In Proceedings of IPTPS, 2002

[33] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Schenker, "A scalable content-addressable network," in Proceedings of SIGCOMM, pp. 161–172, 2001

[34] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup

Service for Internet Applications", ACM SIGCOMM, pp. 149-160, 2001

[35] A. Rowstron and P. Druschel, “Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems,” in Proceedings of Middleware, pp. 329–350, 2001

[36] B.Y. Zhao, L. Huang, J. Stribling, S.C. Rhea, A.D. Joseph, and J.D. Kubiatowicz. "Tapestry: A Resilient Global-Scale Overlay for Service Deployment", IEEE Journal on Selected Areas in Communication, VOL. 22, NO. 1, 2004

[37] B. Fitzpatrick. “Distributed caching with memcached.” Linux Journal, 2004(124):5, 2004

[38] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, W. Vogels. “Dynamo: Amazon’s Highly Available Key-Value Store.” SIGOPS Operating Systems Review, 2007

[39] H. Shen, C. Xu, and G. Chen. Cycloid: A Scalable Constant-Degree P2P Overlay Network. Performance Evaluation, 63(3):195-216, 2006

[40] Ketama, http://www.audioscrobbler.net/development/ketama/, 2011

[41] Riak, https://wiki.basho.com/display/RIAK/Riak, 2011

[42] Maidsafe-DHT, http://code.google.com/p/maidsafe-dht/, 2011

[43] C-MPI, http://c-mpi.sourceforge.net/, 2011

[44] M. Wilde, I. Raicu, A. Espinosa, Z. Zhang, B. Clifford, M. Hategan, K. Iskra, P. Beckman, I. Foster. “Extreme-scale scripting: Opportunities for large task-parallel applications on petascale computers”, SciDAC09, 2009

[45] C. Docan, M. Parashar, S. Klasky. "DataSpaces: An Interaction and Coordination Framework for Coupled Simulation Workflows", ACM HPDC 2010

[46] H. Stern. "Managing NFS and NIS". O’Reilly & Associates, Inc., 1991

[47] P.J. Braam. "The Coda distributed file system", Linux Journal, #50, 1998

[48] D. Nagle, D. Serenyi, A. Matthews. "The panasas activescale storage cluster: Delivering scalable high bandwidth storage". In SC ’04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing, 2004

[49] Microsoft Inc. "Distributed File System", http://www.microsoft.com/windowsserversystem/dfs/default.mspx, 2011

[50] GlusterFS, http://www.gluster.com/, 2011

[51] Isilon Systems. "OneFS", http://www.isilon.com/, 2011

[52] "POHMELFS: Parallel Optimized Host Message Exchange Layered File System", http://www.ioremap.net/projects/pohmelfs/, 2011

[53] F. Hupfeld, T. Cortes, B. Kolbeck, E. Focht, M. Hess, J. Malo, J. Marti, J. Stender, E. Cesario. "XtreemFS - a case for object-based storage in Grid data management". VLDB Workshop on Data Management in Grids, 2007

[54] Y. Gu, R. Grossman, A. Szalay, A. Thakar. “Distributing the Sloan Digital Sky Survey Using UDT and Sector,” e-Science 2006

[55] CloudStore, http://kosmosfs.sourceforge.net/, 2011

[56] S.A. Weil, S.A. Brandt, E.L. Miller, D.D.E. Long, C. Maltzahn. "Ceph: A scalable, highperformance distributed file system". In Proceedings of the 7th OSDI, 2006

[57] W. Xiaohui, W.W. Li, O. Tatebe, X. Gaochao, H. Liang, J. Jiubin. “Implementing Data Aware Scheduling in Gfarm Using LSF Scheduler Plugin Mechanism,” GCA05, 2005

[58] X. Wei, Li Wilfred W., T. Osamu, G. Xu, L. Hu, J. Ju. “Integrating Local Job Scheduler – LSF with Gfarm,” ISPA05, vol. 3758/2005, 2005

[59] MooseFS, http://www.moosefs.org/, 2011

[60] D. Thain, C. Moretti, J. Hemmes. “Chirp: A Practical Global Filesystem for Cluster and Grid Computing,” JGC, Springer, 2008

[61] S. Al-Kiswany, A. Gharaibeh, M. Ripeanu. "The Case for a Versatile Storage System", Workshop on Hot Topics in Storage and File Systems (HotStorage’09), 2009

[62] P. Druschel, A. Rowstron. "Past: Persistent and anonymous storage in a peer-to-peer networking environment". In Proceedings of the 8th IEEE Workshop on Hot Topics in Operating Systems (HotOS), 2001

[63] Circle, http://savannah.nongnu.org/projects/circle/, 2011

[64] J. Ousterhout, et al. “The case for RAMclouds: Scalable high-performance storage entirely in DRAM”. In Operating system review, 2009

[65] J. Dean, S. Ghemawat. “MapReduce: Simplified Data Processing on Large Clusters,” OSDI 2004

[66] A.L. Chervenak, N. Palavalli, S. Bharathi, C. Kesselman, R. Schwartzkopf, “The Replica Location Service”, IEEE HPDC, 2004

[67] A. Chervenak, R. Schuler. “The Data Replication Service”, Technical Report, USC ISI, 2006

[68] K. Ranganathan I. Foster, “Simulation Studies of Computation and Data Scheduling Algorithms for Data Grids”, Journal of Grid Computing, V1(1) 2003

[69] T. Kosar. “A New Paradigm in Data Intensive Computing: Stork and the Data-Aware Schedulers,” IEEE CLADE 2006

[70] D.E. Culler, A. Arpaci-Dusseau, R. Arpaci-Dusseau, B. Chun, S. Lumetta, A. Mainwaring, R. Martin, C. Yoshikawa, F. Wong. “Parallel computing on the berkeley now”, Symposium on Parallel Processing, 1997

[71] R. Arpaci-Dusseau. “Run-time adaptation in river”, ACM Transactions on Computer Systems, 21(1):36–86, 2003

[72] Y. Zhao, I. Raicu, I. Foster, M. Hategan, V. Nefedova, M. Wilde. “Realizing Fast, Scalable and Reliable Scientific Computations in Grid Environments”, Grid Computing Research Progress, Nova Publisher 2008

[73] M. Branco, “DonQuijote - Data Management for the ATLAS Automatic Production System”, Computing in High Energy and Nuclear Physics (CHEP04), 2004

[74] D.L. Adams, K. Harrison, C.L. Tan. “DIAL: Distributed Interactive Analysis of Large Datasets”, Computing in High Energy and Nuclear Physics (CHEP 06), 2006

[75] A. Chervenak, et al. “High-Performance Remote Access to Climate Simulation Data: A Challenge Problem for Data Grid Technologies”, Parallel Computing, Special issue on

High performance computing with geographical data, 2003

[76] M. Beynon, T.M. Kurc, U.V. Catalyurek, C. Chang, A. Sussman, J.H. Saltz. “Distributed Processing of Very Large Datasets with DataCutter”, Parallel Computing, Vol. 27, No. 11, pp. 1457-1478, 2001

[77] D.T. Liu, M.J. Franklin. “The Design of GridDB: A Data-Centric Overlay for the Scientific Grid”, VLDB04, pp. 600-611, 2004

[78] H. Andrade, T. Kurc, A. Sussman, J. Saltz. "Active Semantic Caching to Optimize Multidimensional Data Analysis in Parallel and Distributed Environments", Parallel Computing Journal, 2007

[79] Y. Zhao, M. Hategan, B. Clifford, I. Foster, G. von Laszewski, I. Raicu, T. Stef-Praun, M. Wilde. “Swift: Fast, Reliable, Loosely Coupled Parallel Computation,” IEEE Workshop on Scientific Workflows 2007

[80] M. Bancroft, J. Bent, E. Felix, G. Grider, J. Nunez, S. Poole, R. Ross, E. Salmon, L. Ward. "HEC FSIO 2008 Workshop Report", High End Computing Interagency Working Group (HECIWG), Sponsored File Systems and I/O Workshop HEC FSIO, 2009

[81] T. Hacker. "CAREER: Aon - An Integrative Approach to Petascale Fault Tolerance", NSF CCF CAREER Award, 2010

[82] R. Vuduc. "CAREER: Autotuning Foundations for Exascale Computing", NSF CCF CAREER Award, 2010

[83] J. Wang. "CAREER: Data-Intensive HPC Analytics: A Systems Approach Through Extended Interfaces, Data Restructuring and Data-centric Scheduling", NSF CCF CAREER Award, 2010

[84] A. Butt. "CAREER: A Scalable Hierarchical Framework for High-Performance Data Storage", NSF CCF CAREER Award, 2008

[85] K. Shen. "CAREER: System Support for Data-Intensive Online Applications", NSF CCF CAREER Award, 2005

[86] A. Sirer. "CAREER: Building Robust, High-Performance Infrastructure Services Through Informed Resource Tradeoffs", NSF CNS CAREER Award, 2006

[87] T. Kosar. "CAREER: Data-aware Distributed Computing for Enabling Large-scale Collaborative Science", NSF CNS CAREER Award, 2009

[88] D. Thain. "CAREER: Data Intensive Grid Computing on Active Storage Clusters", NSF CNS CAREER Award, 2007

[89] J. A. Chandy, "Active Storage Networks for High End Computing", NSF HECURA Award, 2006

[90] J. Chandy. "Active Object Storage to Enable Scalable and Reliable Parallel File System", NSF HECURA Award, 2009

[91] A. Maccabe, K. Schwann. "Petascale I/O for High End Computing", NSF HECURA Award, 2006

[92] W. Ligon. "Improving Scalability in Parallel File Systems for High End Computing", NSF HECURA Award, 2006

[93] A. Choudhary, M. Kandemir. "Scalable I/O Middleware and File System Optimizations for High-Performance Computing", NSF HECURA Award, 2006

[94] X. Ma, A. Sivasubramaniam, Y. Zhou. "Application-adaptive I/O Stack for Data-intensive Scientific Computing", NSF HECURA Award, 2006

[95] K. Shen. "Concurrent I/O Management for Cluster-based Parallel Storages", NSF HECURA Award, 2006

[96] H. Jiang, Yifeng Zhu. "SAM^2 Toolkit: Scalable and Adaptive Metadata Management for High-End Computing", NSF HECURA Award, 2006

[97] Y. Zhu, J. Hong. "A New Semantic-Aware Metadata Organization for Improved File-System Performance and Functionality in High-End Computing", NSF HECURA Award, 2009

[98] T. Li, X. He, T. Zhang. "Cross-Layer Exploration of Non-Volatile Solid-State Memories to Achieve Effective I/O Stack for High-Performance Computing Systems", NSF HECURA Award, 2009

[99] A. Szalay, H. Huang. "Balanced Scalable Architectures for Data-Intensive Supercomputing", NSF HECURA Award, 2009

[100] C. Leiserson. "Microdata Storage Systems for High-End Computing", NSF HECURA Award, 2006

[101] G.R. Gao. "A High Throughput Massive I/O Storage Hierarchy for PETA-scale High-end Architectures", NSF CPA Award 2007

[102] R. Arpaci-Dusseau. "HaRD: The Wisconsin Hierarchically-Redundant, Decoupled Storage Project", NSF HECURA Award, 2009

[103] V. Sarkar, J. Dennis, G. Gao. "Programming Models and Storage System for High Performance Computation with Many-Core Processors", NSF HECURA Award, 2009

[104] I. Raicu. "Many-Task Computing: Bridging the Gap between High Throughput Computing and High Performance Computing", Doctorate Dissertation, University of Chicago, 2009

[105] I. Raicu. “Many-Task Computing: Bridging the Gap between High Throughput Computing and High Performance Computing”, VDM Verlag Dr. Muller Publisher, 2009

[106] I. Raicu, et al. "Middleware Support for Many-Task Computing", Cluster Computing, The Journal of Networks, Software Tools and Applications, 2010

[107] M. Wilde, I. Foster, K. Iskra, P. Beckman, Z. Zhang, A. Espinosa, M. Hategan, B. Clifford, I. Raicu. "Parallel Scripting for Applications at the Petascale and Beyond", IEEE Computer Nov. 2009 Special Issue on Extreme Scale Computing, 2009

[108] I. Raicu, I. Foster, Y. Zhao, P. Little, C. Moretti, A. Chaudhary, D. Thain. “The Quest for Scalable Support of Data Intensive Workloads in Distributed Systems”, ACM HPDC, 2009

[109] I. Raicu, I. Foster, A. Szalay, G. Turcu. "AstroPortal: A Science Gateway for Large-scale Astronomy Data Analysis", TeraGrid Conference, 2006

[110] I. Foster, Y. Zhao, I. Raicu, S. Lu. “Cloud Computing and Grid Computing 360-Degree Compared”, IEEE GCE, 2008