Serving Data to the Lunatic Fringe

The Evolution of HPC Storage

JOHN BENT, BRAD SETTLEMYER, AND GARY GRIDER

John Bent. Making supercomputers superer for over a decade. At Seagate Government Solutions. john.bent@seagategov.com.

John Bent.

Brad Settlemyer is a Storage Systems Researcher and Systems Programmer specializing in high performance computing. He works as a research scientist in Los Alamos National Laboratory's Systems Integration group. He has published papers on emerging storage systems, long distance data movement, network modeling, and storage system algorithms. bws@lanl.gov

As Division Leader of the High Performance Computing (HPC) Division at Los Alamos National Laboratory, Gary Grider is responsible for all aspects of high performance computing technologies and deployment at Los Alamos. Gary is also the US Department of Energy Exascale Storage, I/O, and Data Management National Co-Coordinator. Gary has 30 active patents/applications in the data storage area and has been working in HPC and HPC-related storage since 1984. ggrider@lanl.gov

Before the advent of Big Data, the largest storage systems in the world were found almost exclusively within high performance computing centers such as those found at US Department of Energy national laboratories. However, these systems are now dwarfed by large datacenters such as those run by Google and Amazon. Although HPC storage systems are no longer the largest in terms of total capacity, they do exhibit the largest degree of concurrent write access to shared data. In this article, we will explain why HPC applications must necessarily exhibit this degree of concurrency and the unique HPC storage architectures required to support them.

Computing for Scientific Discovery

High performance computing (HPC) has radically altered how the scientific method is used to aid in scientific discovery and has enabled the development of scientific theories that were previously unimaginable. Difficult to observe phenomena, such as galaxy collisions and quantum particle interactions, are now routinely simulated on the world's largest supercomputers, and large-scale scientific simulation has dramatically decreased the time between hypothesis and experimental analysis. As scientists increasingly use simulation for discovery in emerging fields such as climatology and nuclear fusion, demand is driving the growth of HPC platforms capable of supporting ever-increasing levels of fidelity and accuracy. Extreme-scale HPC platforms (i.e., supercomputers), such as Oak Ridge National Laboratory's Titan or Los Alamos National Laboratory's Trinity, incorporate tens of thousands of processors, memory modules, and storage devices into a single system to better support simulation science. Researchers at universities and national laboratories are continuously striving to develop algorithms to fully utilize these increasingly powerful and complex supercomputers.

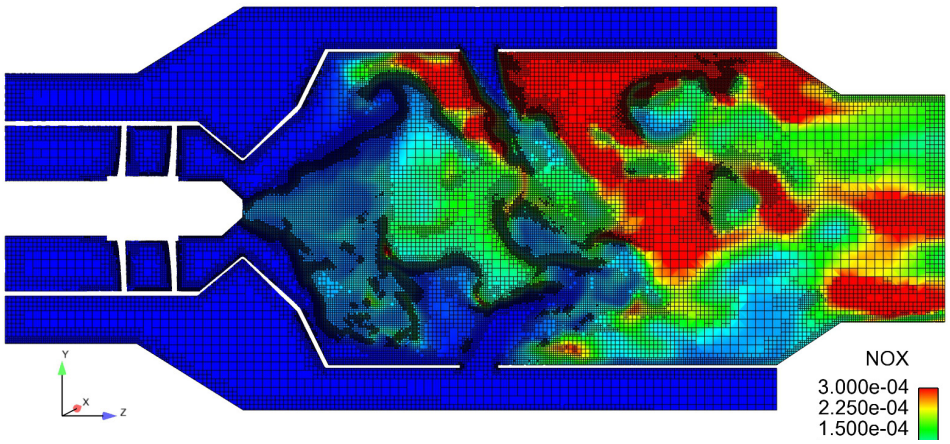


Figure 1: Adaptive Mesh Refinement. An example of adaptive mesh refinement for a two-dimensional grid in which the most turbulent areas of the mesh have been highly refined (from https://convergecfd.com/applications/gas-turbines/).

A simulation is typically performed by decomposing a physical region of interest into a collection of cells called a mesh and then calculating how the properties of the elements within each cell change over time. The mesh cells are distributed across a set of processes running across the many compute nodes within the supercomputer. Contiguous regions of cells are assigned to each process, and processes must frequently communicate to exchange boundary conditions between neighboring cells split across processes. Although replicated cells, called ghost cells, are sometimes used to reduce the frequency of communication, processes still typically exchange messages dozens of times per second.

Additional communication occurs during a *load-leveling* phase. Complex areas of the mesh that contain a large number of different types of elements are more difficult to simulate with high fidelity. Accordingly, simulations will often subdivide these areas into smaller cells as shown in Figure 1. This process of *adaptive mesh refinement* causes work imbalance as some processes within the parallel application will suddenly be responsible for a larger number of cells than their siblings. Therefore the simulation will rebalance the assignment of cells to processes following these refinements.

Due to the frequency of communication, the processes must run in parallel and avoid performance deviations to minimize the time spent waiting on messages. This method, *tightly coupled bulk synchronous computation,* is a primary differentiator between HPC and Big Data analytics.

Another primary differentiator is that the memory of each process is constantly overwritten as the properties of the mesh are updated. The total amount of this distributed memory has grown rapidly. For example, an astrophysics simulation on LANL's Trinity system may require up to 1.5 PB of RAM to represent regions of space with sufficient detail. Memory is one of the most precious resources in a supercomputer; most large-scale simulations expand to use all available memory. The large memory requirements coupled with the large amount of time required to simulate complex physical interactions leads to a problem for the users of large-scale computing systems.

The Need for Checkpointing

How can one ensure the successful completion of a simulation that takes days of calculation using tens of thousands of tightly coupled computers with petabytes of constantly overwritten memory?

The answer to that question has been checkpoint-restart. Storing the program state into a reliable storage system allows a failed simulation to restart from the most recently stored state.

Seemingly a trivial problem in the abstract, checkpoint-restart in practice is highly challenging because the actual writes in a checkpoint are extremely chaotic. One, the amount of data stored by each process is unlikely to match any meaningful block size in the storage system and is thus unaligned. Two, the writes are to shared data sets and thus incur either metadata or data bottlenecks [3, 8]. Three, the writes are *bursty* in that they all occur concurrently during the application checkpoint phase following a large period of storage idleness during the application compute phase. Four, the required bandwidth is very high; supercomputer designers face pressure to ensure 90% efficiency such that the checkpoint-restart of massive amounts of memory must complete quickly enough that no more than 10% of supercomputer lifetime is used.

Although many techniques have been developed to reduce this chaos [4], they are typically not available in practice. Incremental checkpointing reduces the size of the checkpoint but does not help when the memories are constantly overwritten. Uncoordinated checkpointing reduces burstiness but is not amenable to bulk synchronous computation. Two-phase I/O improves performance by reorganizing chaotic writes into larger aligned writes. Checkpointing into neighbor memory improves performance by eliminating media latencies. However, neither of these latter two is possible when all of available memory is used by the application.

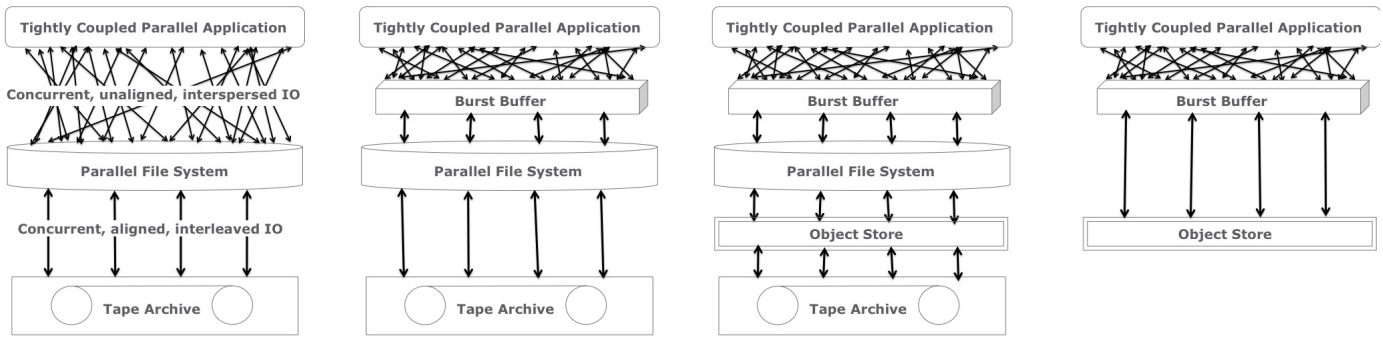
Thus, HPC storage workloads, lacking common ground with read-intensive cloud workloads or IOPS-intensive enterprise workloads, have led to the creation of *parallel file systems*, such as BeeGFS, Ceph, GPFS, Lustre, OrangeFS, and PanFS, designed to handle bursty and chaotic checkpointing.

Storage for Scientific Discovery

From the teraflop to the petaflop era, the basic supercomputer architecture was remarkably consistent, and parallel file systems were the primary building block of its storage architecture. Successive supercomputers were largely copied from the same blueprint because the speed of processors and the capacities of memory and disk all grew proportionally to each other following Moore's Law. Horizontal arrows in Table 1 show how these basic

FLOPS / RAM	⇐⇒
RAM / core	⇓
MTTF per component	⇐⇒
MTTI per application	⇓
Impact of performance deviations	⇑
Drive spindles for capacity	⇐⇒
Drive spindles for bandwidth	⇑
Tape cassettes for capacity	⇐⇒
Tape drives for bandwidth	⇑
Storage clients / servers	⇑

Table 1: Supercomputer Trends Affecting Storage. Horizontal lines do not necessarily indicate no growth in absolute numbers but rather that the trend follows the relative overall growth in the machine.



(a) 2002–2015 (b) 2015–2016 (c) 2016–2020 (d) 2020–

Figure 2: From 2 to 4 and back again. Static for over a decade, the HPC storage stack has now entered a period of rapid change.

architectural elements scaled proportionally. However, recent inflection points, shown with the vertical arrows, require a redesign of both the supercomputer and its storage architecture.

Era: 2002–2015. The storage architecture for the teraflop and petaflop eras is shown in Figure 2a. Large tightly coupled applications ran on large clusters of mostly homogeneous components. Forward progress, despite inevitable application interruptions, was ensured via checkpoint-restart. Tape-based archives stored cold data permanently, and disk-based parallel file systems satisfied the storage requirements for hot data.

These storage requirements are easily quantifiable for each supercomputer. An optimal checkpoint frequency is derived from the MTTI. The checkpoint size is typically no larger than 80% of the memory of the system, and HPC sites typically desire an efficiency of at least 90%. Given a checkpoint frequency and a checkpoint size, the required checkpoint bandwidth must be sufficient such that the total time spent checkpointing (and restarting and recomputing any lost work) is less than 10% of a system's operational time. The storage system also includes a capacity requirement derived from the needs of the system users and typically results in a capacity requirement between 30 and 40 times the size of the system memory.

As an example, imagine an exascale computer with 32 PB of memory and a checkpoint frequency of one hour. Ignoring restart and recompute, a checkpoint can take no more than six minutes, and therefore the required storage bandwidth must be at least 72 TB/s. The required storage capacity must be at least 960 PB.

An important characteristic of supercomputers within this era was that the minimum number of disks required for capacity was larger than the minimum number of disks required for bandwidth. This ensured that the simple model of a storage tier for performance and a second tier for capacity was economically optimal throughout this era.

During this era, total transistor counts continued to double approximately every 24 months. However, in the second half of this era, they did so *horizontally* by adding more compute nodes with larger core counts as opposed to merely increasing transistors within cores. This has had important implications which affect the storage stack: (1) although component reliability is little changed, large-scale systems built with increasing numbers of components are experiencing much shorter mean times to failure; (2) the amount of private memory available to each process is decreasing; (3) the number of processes participating in checkpoint-restart is growing; and (4) since larger supercomputers are more sensitive to performance deviations across their components, programming models which rely on bulk synchronous computing are less efficient.

Although petascale systems such as LANL's Roadrunner and ORNL's Jaguar began to stress the existing storage model in 2008, it survived until 2015.

Era: 2015–2016. In this era the inflection point in Table 1 noting the growth in the number of disk spindles required to meet checkpoint bandwidth requirements becomes a limitation. Until 2013, the system capacity requirement ensured sufficient disks to simultaneously satisfy the performance requirement. However, as disks have gotten relatively slower (as compared to their growth in capacity), a larger number of spindles is required to meet the bandwidth demand. A flash-only storage system could easily satisfy performance but would be prohibitively expensive for capacity: thus the introduction of a small performance tier situated between the application and the disk-based parallel file system as shown in Figure 2b. This tier is commonly referred to as a *burst buffer* [1, 7] because it is provisioned with enough bandwidth to meet the temporary bandwidth requirement but not sufficient capacity to meet the overall system demands. The bursty nature of the checkpoint-restart workload allows sufficient time to *drain* the data to a larger-capacity disk-based system. Initial burst buffer systems have been built on TACC's Wrangler, NERSC's Cori, and LANL's Trinity supercomputers.

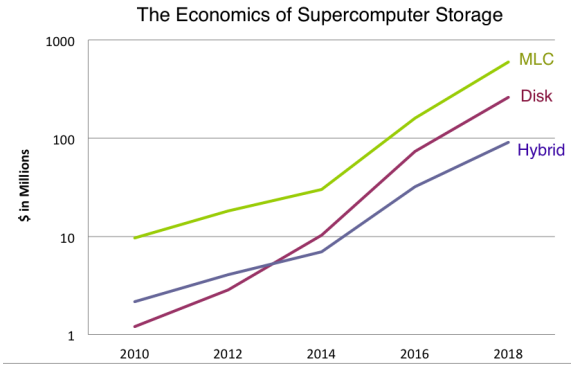


Figure 3: HPC Storage System Costs. This graph shows the media costs required to satisfy the checkpoint bandwidth and capacity requirement for 90% forward progression. We can see that in 2013, storage systems composed of one media type (disks or flash) are not as economical as a storage system that uses flash to meet bandwidth requirements and disks to meet capacity requirements.

Era: 2016–2020. A similar shift to that which motivated burst buffers now has occurred in the economic ratio between tape and disk. Although tape's primary requirement is to satisfy long-term capacity, it also has a performance requirement which is becoming increasingly expensive due to unique characteristics of tape. Specifically, tape capacity and performance are purchased separately, whereas, for disk and flash, they are purchased together; for example, a typical disk might provide a TB of capacity, 100 MBs of bandwidth, and 100 IOPS. Conversely, tape capacity is purchased in inexpensive cassettes, bandwidth in expensive drives, and IOPS in very expensive robots to connect the two. Analysis has shown that a hybrid disk-tape system was the correct economic choice for archival HPC storage as early as 2015 [6], as shown in Figure 2c. The emergence of efficient erasure coded object storage software has enabled the development of MarFS [5], the first instance of a disk-based archival storage system for HPC data that can provide the minimal bandwidth and extreme levels of data protection required for long-term data retention. For reference, the four storage tiers of LANL's 2016 Trinity supercomputer are shown in Table 2.

For future supercomputers of this era, we also expect that the physical location of the flash memory burst buffers will change. Instead of being in dedicated external nodes accessible to all of the compute nodes, flash memory will begin appearing within the supercomputer (e.g., node-local storage). Despite this change, this era will continue to be defined by the presence of a parallel file system.

Era: 2020 Onward. The four storage tiers shown in Figure 2c are an unnecessary burden on system integrators, administrators, and users. Therefore, as shown in Figure 2d, we predict a return to two-tier storage architectures for HPC systems. Also in this era, we predict the emergence of new storage interfaces to better utilize node-local storage.

Return to Two Tiers. Parallel file systems were created to handle chaotic, complex, and unpredictable streams of I/O as shown in Figure 2a. Figure 2b shows that burst buffers now absorb that chaos, leaving parallel file systems serving only orderly streams of I/O issued by system utilities. When burst buffers were first introduced, the possibility that these orderly streams might go directly to the tape archive was discussed. And while this was feasible for writes, reads would have experienced unacceptable latency. However, the move to disk-based object stores as shown in Figure 2c does provide sufficient performance for both reads and writes issued from the burst buffer. Thus, a separate parallel file system is no longer needed. The top two storage tiers in Figure 2c will combine, and the burst buffer will subsume the parallel file system.

Similarly, the bottom two tiers will merge. The erasure codes used in object storage present a much richer approach to reliability and data durability than modern tape archives. As the costs of an object store fall below those of a tape archive [6], the role of tape will increasingly be filled by the disk-based object store.

Early prototypes of a return to a two-tier HPC storage system are EMC's eponymous 2 Tiers™ and CMU's BatchFS [9]. Both expose a file system interface to the application, store hot data in a burst buffer, and can store cold data in an object store.

New Storage Interfaces. Although the burst buffer will subsume the parallel file system, over time it will decreasingly resemble current systems. Many bottlenecks within parallel file systems are due to the legacy POSIX semantics that are typically unnecessary for HPC applications. That vendors have provided POSIX is not surprising given that parallel file systems are also used by enterprise (i.e., non-HPC) customers who require more rigorous semantics.

However, as burst buffers will begin appearing as local storage within each compute node, these bottlenecks will finally become intractable. Maintaining POSIX semantics for a global

	Size	Bandwidth	Lifetime
Memory	2.1 PB	1–2 PB/s	milliseconds
Burst Buffer	3.7 PB	4–6 TB/s	hours
Parallel FS	78 PB	1–2 TB/s	weeks
Object Store	30 PB	100–300 GB/s	months
Tape Archive	50+ PB	10 GB/s	forever

Table 2: The Four Tiers of Trinity Storage. LANL's Trinity supercomputer uses Cray's DataWarp as a burst buffer, Lustre as a parallel file system, MarFS as a file system interface over an object store, and HPSS for the tape archive. The sizes above reflect the sizes at installation; the object store and the tape archive will grow over the lifetime of the machine.

namespace across distributed local burst buffers destroys the low latency of node-local storage since data access requires higher latency communications with a remote centralized metadata service. Thus, emerging systems like DeltaFS [10] and Intel's DAOS-M employ relaxed semantics to allow HPC applications with less rigorous storage requirements to achieve higher performance. We expect that these, or similar storage services, will increasingly appear as user-space libraries utilizing OS-bypass for low-latency access to local storage with eventual namespace reconciliation as first developed in Coda.

Further, the increased sensitivity to system noise is diminishing the efficiency of bulk synchronous computing models. Accordingly, programming models such as Stanford's Legion and asynchronous MPI will become attractive alternatives, but they will require asynchronous checkpointing models, such as uncoordinated checkpointing using message logging [4] or asynchronous transactions [2] with relaxed consistency semantics.

Conclusion

Unique properties of scientific simulations require unique storage architectures. Static for over a decade, HPC storage systems have now entered a period of rapid development. The opportunity for innovation in HPC storage is currently immense. We urge interested readers to join us in building new storage technologies for exascale computing.

Acknowledgments

A portion of this work was performed at the Ultrascale Systems Research Center (USRC) at Los Alamos National Laboratory, supported by the US Department of Energy contract DE-FC02-06ER25750 and a CRADA between LANL and EMC. The publication has been assigned the LANL identifier LA-UR-16-21697.

References

[1] J. Bent, S. Faibish, J. Ahrens, G. Grider, J. Patchett, P. Tzelnic, and J. Woodring, "Jitter-Free Co-Processing on a Prototype Exascale Storage Stack," in *Proceedings of the IEEE 28th Symposium on Mass Storage Systems and Technologies (MSST),* April 2012, pp. 1–5.

[2] J. Bent, B. Settlemyer, H. Bao, S. Faibish, J. Sauer, and J. Zhang, "BAD Check: Bulk Asynchronous Distributed Checkpointing and IO," in *Proceedings of the Petascale Data Storage Workshop at SC15 (PDSW15),* Nov. 2015, pp. 19–24.

[3] P. Carns, S. Lang, R. Ross, M. Vilayannur, J. Kunkel, and T. Ludwig, "Small-File Access in Parallel File Systems," in *Proceedings of the IEEE International Symposium on Parallel Distributed Processing (IPDPS 2009),* May 2009, pp. 1–11.

[4] K. B. Ferreira, "Keeping Checkpointing Viable for Exascale Systems," PhD dissertation, University of New Mexico, Albuquerque, 2011, ISBN: 978-1-267-28351-1.

[5] G. Grider et al., "MarFS—A Scalable Near-Posix Metadata File System with Cloud Based Object Backend," in *Proceedings of the 10th Parallel Data Storage Workshop (PDSW '15),* Abstract—Work-in-Progress, 2015: http://www.pdsw.org/pdsw15/wips/wip-lamb.pdf.

[6] J. Inman, G. Grider, and H. B. Chen, "Cost of Tape Versus Disk for Archival Storage," in *Proceedings of the IEEE 7th International Conference on Cloud Computing (CLOUD),* June 2014, pp. 208–215.

[7] N. Liu, J. Cope, P. Carns, C. Carothers, R. Ross, G. Grider, A. Crume, and C. Maltzahn, "On the Role of Burst Buffers in Leadership-Class Storage Systems," in *Proceedings of the 2012 IEEE Conference on Massive Data Storage,* 2012: https://www.mcs.anl.gov/papers/P2070-0312.pdf.

[8] P. Nowoczynski, N. Stone, J. Yanovich, and J. Sommerfield, "Zest: Checkpoint Storage System for Large Supercomputers," in *Proceedings of the 3rd Parallel Data Storage Workshop (PDSW '08),* Nov. 2008, pp. 1–5.

[9] Q. Zheng, K. Ren, and G. Gibson, "BatchFS: Scaling the File System Control Plane with Client-Funded Metadata Servers," in *Proceedings of the 9th Parallel Data Storage Workshop (PDSW '14),* 2014, pp. 1–6: http://dx.doi.org/10.1109/PDSW.2014.7.

[10] Q. Zheng, K. Ren, G. Gibson, B. W. Settlemyer, and G. Grider, "DeltaFS: Exascale File Systems Scale Better without Dedicated Servers," in *Proceedings of the 10th Parallel Data Storage Workshop (PDSW '15),* ACM, 2015, pp. 1–6: http://doi.acm.org/10.1145/2834976.2834984.

REGISTER TODAY!

usenix
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

2016 USENIX Annual Technical Conference

JUNE 22–24, 2016 • DENVER, CO
www.usenix.org/atc16

USENIX ATC '16 brings leading systems researchers together for cutting-edge systems research and unlimited opportunities to gain insight into a variety of must-know topics, including virtualization, system administration, cloud computing, security, and networking.

Co-located with USENIX ATC '16:

SOUPS 2016

Twelfth Symposium on Usable Privacy and Security

JUNE 22–24, 2016
www.usenix.org/soups2016

SOUPS 2016 will bring together an interdisciplinary group of researchers and practitioners in human computer interaction, security, and privacy. The program will feature technical papers, workshops and tutorials, a poster session, panels and invited talks, and lightning talks.

HotCloud '16

8th USENIX Workshop on Hot Topics in Cloud Computing

JUNE 20–21, 2016

Researchers and practitioners at HotCloud '16 share their perspectives, report on recent developments, discuss research in progress, and identify new/emerging "hot" trends in cloud computing technologies.

HotStorage '16

8th USENIX Workshop on Hot Topics in Storage and File Systems

JUNE 20–21, 2016

HotStorage '16 is an ideal forum for leading storage systems researchers to exchange ideas and discuss the design, implementation, management, and evaluation of these systems.