SC: U: Lessons from Big Data Processing on Modern Flash-based HPC Systems

Adnan Haider
Illinois Institute of Technology ahaider3@hawk.iit.edu

***Abstract*—Progress in business analytics and scientific fields, such as climatology, bioinformatics, and high energy physics, relies on processing big data. To process big data faster, high performance computing systems have recently begun to deploy flash storage devices at large scale. However, there exists variation in *how* flash devices are deployed on high performance computing systems. We found this variation leads to unexpected performance degradations from using flash devices. The performance issue occurs when big data applications' I/O workloads are not suitable for the flash device configuration of the system. To quantify this issue, we develop a new metric to fairly compare flash-based HPC systems. Then, we characterize and generalize the performance of a diverse set of I/O workloads across different in-production flash storage architectures, using I/O benchmarks and in-production post-processing climate data software. Then, we improve existing flash storage architectures by a factor of two for our workloads, identify new bottlenecks for applications running on flash-based HPC systems, and describe how big data applications can be adapted to maximize performance on flash-based HPC systems.**

I. Problem and Motivation

Large amounts of data are generated from many scientific simulations and data analysis software in fields such as climate science, bioinformatics, and physics. In order to make use of big data, applications are used to extract and modify this data to provide scientific insights [13]. Many of these applications, which facilitate scientific discovery, are I/O intensive. I/O intensive means the speed at which they execute depends mainly on the underlying hardware on which the data is stored. Scientists can spend days waiting for their applications to execute and some applications create terabyte-scale output bursts for intermediate results [16]. This leads to significant delay in scientific discovery and productivity. The performance of these applications suffers even more considering the current ratio of I/O to compute performance is 1 GBps to 10 teraflops in high performance computing (HPC) systems [8].

To solve this problem, HPC system designers are now deploying flash storage devices at large scale. Flash storage is an order of magnitude faster than disk-based storage and thus allows data to be read and written faster. For example, Gordon released in 2012, Catalyst in 2013, Comet in 2015, and Wrangler in 2015 all have a large-scale deployment of flash devices [2] [11]. Two other HPC systems, Cori and Aurora, will be available in 2016 and 2019 with large-scale flash as well [3] [1]. Aurora is of core importance to scientific progress as it is a result of President Obama's recent signing of an executive order to develop the first exascale machine and thus the fastest supercomputer in the world[4]. This emerging large scale deployment of flash devices was not possible a decade ago, when flash costs were \$80 per GB but are now \$.2 per GB [5]. In the near future, scientists will have the capability of running their big data software on flash devices that can be orders of magnitude faster than traditional hard disk drives, potentially allowing much faster data analysis and scientific breakthroughs.

However with this recent emergence of large scale flash devices, there is significant variation in *how* the flash devices are being incorporated into HPC systems. This variation is in the location of flash devices, the interconnect to the devices, and the type of file system the flash devices are mounted on. We define the term *flash storage architecture* as how an HPC system configures its flash. We found that transitioning I/O intensive software from hard disk-based storage systems to theoretically order of magnitude faster flash devices does not necessarily ensure performance improvement. In fact, we found inappropriate flash storage architectures can negatively impact performance to the degree that flash-based I/O subsystems perform 2x worse than typical disk-based systems.

Based off this insight and the near-term deployment of many flash-based HPC systems, we believe users in many science domains will need to know how to best adapt their applications to various flash storage architectures to maximize performance and cost effectiveness. Also, system designers will need to know how to improve existing flash storage architectures to better suit big data workloads.

In this paper, we compare tradeoffs between different flash storage architectures to help both users and system designers. First we develop a new metric to fairly compare flash-based HPC systems. Using the metric, we analyze performance of a diverse set of I/O workloads using I/O benchmarks and in-production big data applications on two different flash storage architectures. Then, we improve existing flash storage architectures by two factors, identify and quantify new bottlenecks which arise for applications on flash-based architectures, and describe how big data applications can be modified to maximize performance on flash storage architectures.

II. Background and Related Work

Flash devices are a fast type of storage device, which can be orders of magnitude faster than hard disks. A common type of flash device is a solid state drive (SSD). However, any type of storage device which uses flash technology is a flash device. Flash devices are non-volatile and lack a mechanical foundation, unlike hard disks, making them significantly faster than disks. The amount of benefits from flash over disks depends on I/O workloads, such as random vs. sequential access and I/O request size or size of data fetch.

We categorize flash-based HPC systems into three different types of flash storage architecture. The first is a local flash

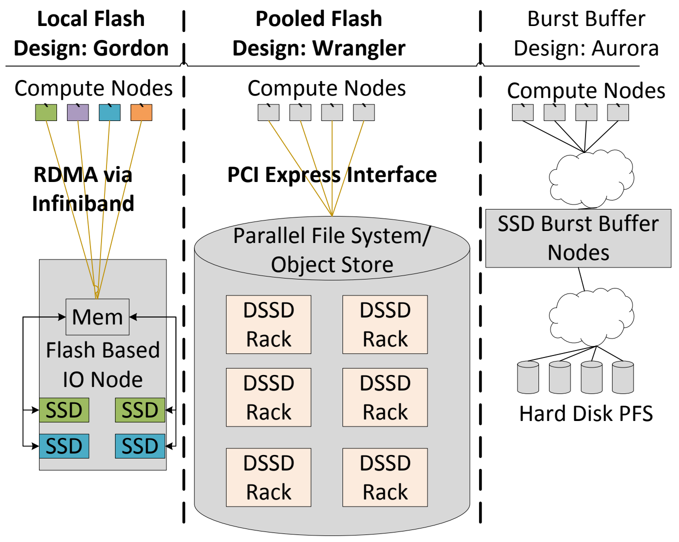


Fig. 1: Current and future I/O architectures incorporating flash. Gordon and Wrangler (bolded) are evaluated in this study.

architecture. On these systems, jobs using flash devices only have access to a *small* subset of all the flash devices in the system. The physical location of the flash devices available for the job can be on the compute node or remotely on an I/O node. This architecture was the first large-scale flash deployment in 2012 with the Gordon system. It continues to be used in systems such as Catalyst (2013) and Comet (2015) [2] [15]. The flash devices in the job are mounted on a local file system.

The second type of architecture is a pooled flash architecture. This architecture allows jobs to use a *large* subset of flash devices in the system under a parallel file system. The location of the devices is stored remotely on a flash array. Currently, there is a single pooled flash architecture in deployment, Wrangler (2015), and has an aggregate flash capacity of 500 TB [11].

The third type of flash architecture is a burst buffer. This architecture deploys flash devices as a storage layer above hard disk drives. They allow for parallel access to a large subset of the flash devices similar to a pooled architecture. However, a burst buffer architecture acts as a *caching layer* for I/O. This is different than Wrangler's pooled architecture where flash devices do not provide any automatic caching. Currently, there is no in-production environment of burst buffer flash. However, there are prototype implementations of burst buffer architecture. Burst buffer architectures will be used for the Cori system releasing in 2016 at NERSC. In addition, the Aurora HPC system will also have a burst buffer architecture releasing in 2019 [1][3]. Due to the lack of a production environment of burst buffers, we did not analyze this type of architecture.

Overall, Figure 1 shows the different types of architectures that are currently being used or will soon be deployed. We found that although all the systems in Figure 1 use flash devices, their performance benefits can vary from 2x worse performance to 6x better performance when compared to their respective disk-based I/O subsystems. This variation is due to the flash storage architecture and applications' I/O workload.

There is a large body of work related to flash devices and their benefits and drawbacks for HPC workloads. These works span multiple different topics. Some works have described how to place flash devices to improve performance of various workloads, such as using flash as a caching layer or using it for performance-critical data blocks [10]. Other works focused on integrating flash with hard disk drives under a hybrid architecture [12]. There has been research on designing burst buffer architectures for different workloads [9]. In addition, some researchers focused on how to develop a data-centric supercomputer [15], which eventually evolved into Gordon, one of the systems we analyze in our work.

Overall, most related work focus on how to integrate flash devices to speedup HPC workloads and less focus on large-scale flash storage architectures. Most of these works focused on prototype implementations of the system. However, some were so influential that they evolved into an in-production system that is used by thousands of users. Our work seeks to analyze the tradeoffs among these in-production systems as well as discover new bottlenecks and improve existing in-production flash storage architectures. Due to the recency of large-scale flash deployments, there has been no study analyzing the performance of different flash storage architectures on in-production high performance computing systems.

III. Approach for Trade-off Analysis

In order to analyze trade-offs between flash architectures, we use an in-production post-processing climate data software, PyReshaper, across five different datasets from the Community Earth System Model (CESM). Each dataset has different I/O workloads, meaning different data fetch sizes and access patterns. This application spends at least 80% of its time conducting I/O and is a major bottleneck in the workflow of climate scientists [13]. In addition to this application, we use two commonly used I/O benchmarks, IOR and mdtest, to quantify the performance of a diverse set of I/O workloads in order to provide a holistic measure of performance for users with a wide variety of different I/O workloads [7].

Analyzing tradeoffs between flash architectures is difficult since there are many variables in addition to the I/O architecture, such as CPU speed, main memory capacity/speed, cache hierarchy, and user load (multiple applications executing). Since no other work has analyzed in-production flash storage architectures, we develop a new metric used to quantify performance benefits of flash storage architectures.

The basic idea of the metric is to measure performance of the flash-based I/O subsystem of an HPC system and then compare this to the performance of the disk-based I/O subsystem of the *same* system. Both flash-based HPC systems we analyze also contain a hard disk-based I/O subsystem. The metric gives us the performance of the flash storage architecture, which includes the performance of the flash device, the interconnect to the flash device, and the file system of the flash device. The metric eliminates the influence of other hardware characteristics of systems such as CPU and main memory performance, since these will remain constant regardless of storage device. We use this flash storage architecture comparison metric to isolate the influence of flash-based architectures from other hardware parameters.

For example, in order to measure the throughput benefits provided by a flash storage architecture, we can first measure

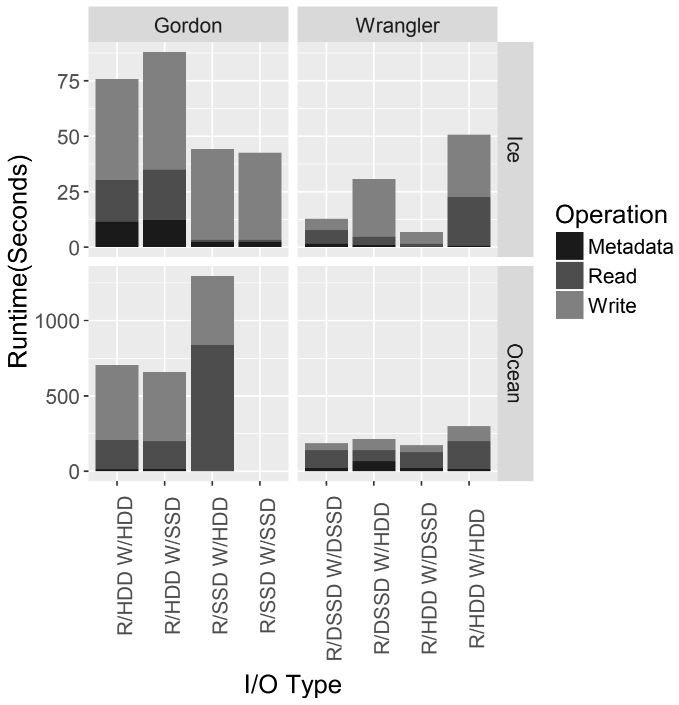
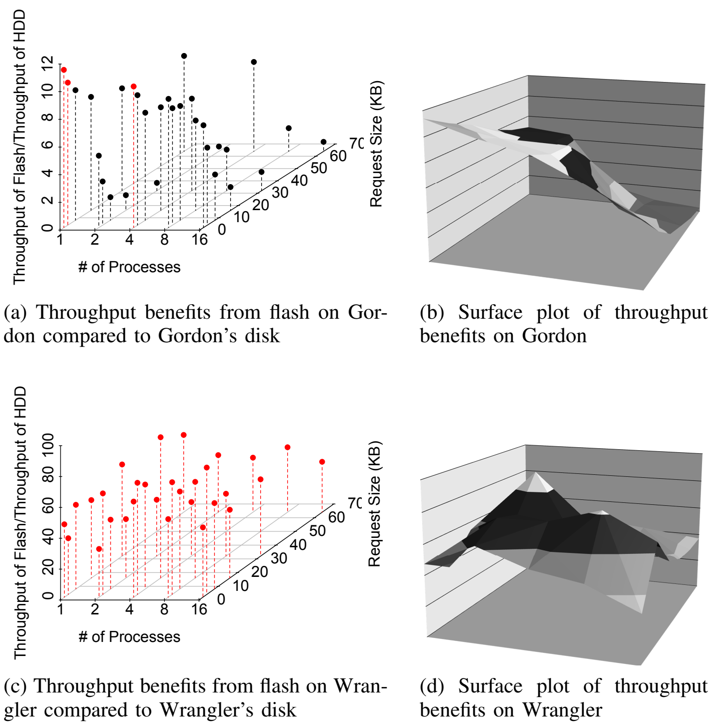


Fig. 2: Performance of Gordon and Wrangler across two datasets, Ice, which is less I/O intensive, and Ocean. R stands for read and W stands for write. Comparisons should be made between bars of the same box since in each box the only variable is the flash storage architecture. R/W SSD on Gordon for the Ocean dataset results in the flash device running out of space and thus has no bar.



(a) Throughput benefits from flash on Gordon compared to Gordon's disk

(b) Surface plot of throughput benefits on Gordon

(c) Throughput benefits from flash on Wrangler compared to Wrangler's disk

(d) Surface plot of throughput benefits on Wrangler

Fig. 3: Scatter and surface plots for throughput benefits from using flash. In both (a) and (c) red dots represent flash devices achieving order of magnitude (10x) better performance than disk. The axes of the surface plots are the same as the scatterplots.

the throughput of the flash devices. Then we measure the throughput of the hard disk drives of the *same* system. Lastly, we normalize the performance by dividing the throughput of flash devices by the throughput of hard disk drives.

In addition to using the above metric for flash storage architecture tradeoff analysis, we also focus on relative *trends* in performance as opposed to raw performance numbers since raw numbers can change with faster storage devices, interconnects, etc. Trends, such as issues in scalability, are more prone to being persistent across different generations of hardware and thus represent a characteristic of the flash architecture. We do not eliminate all potential variables occurring in in-production HPC systems for our tradeoff analysis, such as user load. However using in-production systems allows our results and insights, described in the following sections, to be directly applicable to thousands of users and systems designers who will be or are using the system for their big data workloads.

IV. Results and Contributions

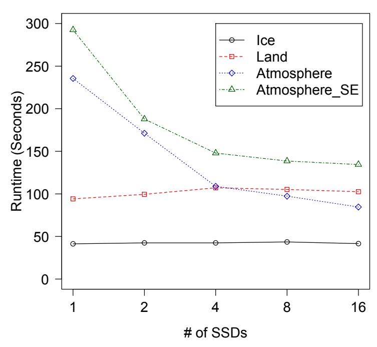
A. *Local Architecture Results*

Gordon has a local I/O architecture, meaning each compute node can read/write from/to a single solid state drive (SSD). SSDs are located on I/O nodes, which are connected to compute nodes via an infiniband interconnect [15]. In this architecture, we found two primary constraints. First, the theoretical bandwidth is capped at the bandwidth of a small subset of flash devices. Second, I/O requests may introduce network latency. In the case of Gordon, I/O requests do go through the network, but more recent systems, such as Comet, store flash devices on the compute node itself.

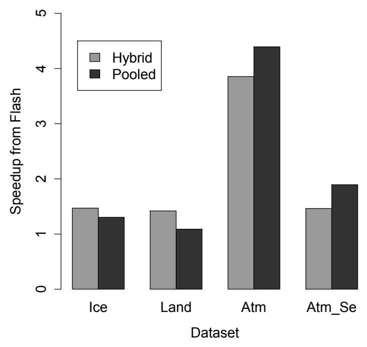
As seen in Figure 2, the flash devices *increased* runtime by a factor of 2 compared to Gordon's disk file system for the more I/O intensive dataset of CESM (Ocean). The single SSD for each compute node and single infiniband interconnect resulted in scalability and latency issues. This is due to contention over the network and accesses becoming queued on the single SSD. Gordon's disk-based parallel file system allows hundreds of hard disks to satisfy I/O requests in parallel easily out-performing a local flash architecture for the more I/O intensive climate dataset.

As shown in Figure 3(a)(b) from running IOR, we found that Gordon's flash architecture outperforms the disk-based file system for applications which output small intermediate data or have a small number of parallel I/O requests. HDDs are known to underperform for small I/O requests. Also, we found that as the number of processes, data amount, and request size increase, the performance advantage of SSDs decreases due to the inevitable saturation of the single flash device.

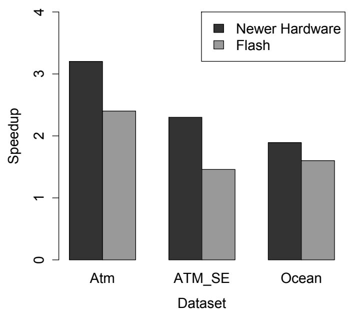
Overall, we found that local flash architectures can provide significant benefits if the I/O workload of applications issue small I/O requests and do not exploit parallel I/O. One such case occurs when scientists analyze simulation data using big data processing frameworks, such as Hadoop. In Hadoop, a



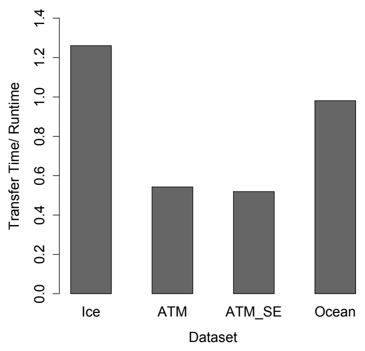
(a)



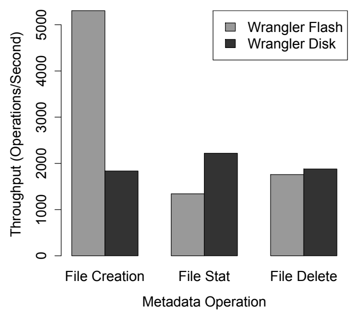
(b)



(c)



(d)



(e)

Fig. 4: Results describing lessons and insights from our experience in using flash-based HPC systems (a): Runtime while adding more flash devices (b): Performance of Hybrid vs. Pooled architectures (c): Performance of newer hardware vs. flash devices (d): Data transfer overhead measured by ratio of transfer time to application runtime (e): Metadata operation throughput

single process is responsible for outputting intermediate data for a single application task [6]. Thus, users should analyze the amount of parallel I/O and request sizes of their application in order to determine if a local architecture is appropriate.

B. *Pooled Architecture Results*

Wrangler, released in 2015, provides 500TB of DSSD storage, which is an order of magnitude lower latency flash device. Wrangler is a pooled flash architecture, meaning each compute node has access to all 500TB of flash storage via a PCI Express connection [11]. Also since all flash devices are underneath a parallel file system, scalability issues seen in a local flash architecture are non-existent. Since the interconnect is PCI Express as compared to Infiniband, interconnect latency was not a significant factor for the applications.

Wrangler provided up to a 6x *reduction* in execution time when reading from hard disk and writing to DSSD (hybrid I/O) compared to when reading and writing to Wrangler's disk-based parallel file system as shown in Figure 2. For both datasets, hybrid I/O provided the best performance with only half the flash storage consumption. When running IOR as shown in Figure 3(c)(d), the pooled DSSD devices provided consistent improvements for all request sizes and process counts, illustrating that a pooled architecture can provide significant improvement for many diverse I/O workloads. This is unlike Gordon's flash architecture which degraded in performance with larger request sizes and process counts.

For post-processing climate data software, hybrid I/O provided the best performance. This can be significant to users and system administrators, since flash storage space is and will continue to be a constraint. For example as of March 2016, there are 1155 users sharing Wrangler's flash I/O subsystem. Hybrid I/O allows reading input datasets from disk-based file systems thus considerably decreasing the storage consumption of flash. Users should analyze their application's performance using hybrid I/O since it can significantly reduce flash storage consumption, which on Wrangler costs user's vital allocation credits.

Using the insights gathered from our experience using flash-based HPC systems, we develop methods to improve existing flash architectures and identify bottlenecks of flash architectures in the next section.

C. *Lessons from Transitioning Big Data Applications to Flash Architectures*

In the previous sections, we conducted a tradeoff analysis of different I/O workloads across different flash storage architectures. From this, we developed methods to better adapt big data applications to flash storage architectures, such as using *hybrid I/O* and *small request sizes with minimal parallelism*. Apart from this analysis, we now develop a method to improve existing architectures and identify new bottlenecks which may arise for big data applications on flash architectures.

There are many local flash architectures deployed in systems. We found that local architectures suffer from scalability issues in Section IV-A. Upon further analysis, we found different datasets depending on their I/O intensity needed a different amount of SSDs to minimize runtime. Thus, allocating a configurable amount of SSDs during job configuration can provide better resource utilization and reduce runtime by 2x compared to a single flash device (Gordon's default) as shown in Figure 4(a). After discussion with the Gordon architecture team, a hybrid configuration, meaning a variable amount of flash devices per job, could be possible with additional software support. After configuring a prototype hybrid flash architecture on Gordon, it reduced runtime by 3x compared to Gordon's disk, while a pooled architecture provided 4x compared to Wrangler's disk as shown in Figure 4(b). A hybrid architecture can be seen as an intermediary architecture between local, small subset of flash, and pooled, large subset of flash. The hybrid architecture can scale between these two extremes and be dynamically configured for different workloads.

Instead of using the metric described in Section III, we simply calculated the speedup from running Pyreshaper on Gordon to running on hard disks of Wrangler. This includes the performance of many different hardware parameters since Wrangler has more modern hardware. We compared this speedup with the speedup from using flash on Wrangler compared to Wrangler's disk. This comparison is shown in Figure 4(c). We found that running on three years of newer hardware (more memory, larger caches, faster interconnects, etc), without using flash provides more improvement than running on flash with all other hardware constant for our workloads. This means that although flash devices provide significant improvement (6x reduction in execution time), other hardware could be just as important when accelerating big data workloads. Thus, users with the goal of accelerating big data applications may not necessarily need to invest in recently released flash-based systems but simply better performing, balanced all around hardware which may not have flash.

Although I/O performance can improve significantly using flash storage architectures, flash architectures can potentially bring about new bottlenecks that didn't occur on traditional disk-based I/O subsystems. Two of such bottlenecks we found are *data transferring* and *metadata operation* overhead. Since flash storage capacity is constrained and for some systems (e.g. Gordon) is only available during the length of the job, any data stored on flash needs to be transferred in and out of flash. This leads to data transfer overhead. As shown in Figure 4(d), across all datasets transferring data in and out of flash devices added on at least 50% more time to overall execution and in some cases took longer than the entire application runtime. As future work, it would be interesting to develop methods to reduce this overhead, such as overlapping data transfer with application execution.

The second bottleneck we found was metadata operations. Many scientific simulation checkpoint data during execution for fault tolerance. This requires processes to create and write to their own file, potentially leading to millions of metadata operations[14]. Because of this, metadata performance can significantly impact performance. As shown in Figure 4(e), metadata operation performance does not see nearly as much performance improvement as typical read/write performance we tested previously. Thus, workloads which have many metadata operations will improve less than those with fewer metadata operations. Thus metadata operations will consume a larger proportion of runtime. Metadata operation performance depends largely on the filesystem. In our case, Wrangler's disk-based filesystem is Lustre while flash devices are mounted on GPFS. We believe understanding how recent developments in efficient distributed metadata filesystems will perform on large-scale flash deployments will be interesting future work [14].

V. Conclusion

In the big data era, scientific progress depends on gaining insight from large quantities of data. Recently, flash-based HPC systems are being deployed to accelerate scientific discovery for thousands of scientists across many domains. We found that although all these systems use flash devices; their performance can vary considerably from 2x increase in runtime to 6x decrease in runtime compared to traditional disks of the same system. This variation occurs because of how the flash devices are deployed in the system (flash storage architecture) and the big data applications' workload. In this work, we develop a new comparison metric, quantify the tradeoffs between flash architectures, develop methods to adapt software to flash architectures, such as using hybrid I/O, identify new bottlenecks for applications, and improve an existing flash storage architecture by 2x. In the future, users and systems designers can use this work to determine which flash architecture is appropriate for their applications and know how to better adapt flash architectures to big data workloads.

References

[1] Aurora. http://aurora.alcf.anl.gov/.

[2] Catalyst. https://www.llnl.gov/news/lawrence-livermore-intel-cray-produce-big-data-machine-serve-catalyst-next-generation-hpc.

[3] Cori burst buffer. http://www.nersc.gov/users/computational-systems/cori/burst-buffer/.

[4] Executive order. https://www.whitehouse.gov/the-press-office/2015/07/29/executive-order-creating-national-strategic-computing-initiative.

[5] Flash price timeline. http://www.jcmit.com/flashprice.htm.

[6] Hadoop map tasks. https://hadoop.apache.org/docs/r1.0.4/mapred-default.html.

[7] Io benchmarks. http://www.mcs.anl.gov/ thakur/pio-benchmarks.html.

[8] N. Ali, P. Carns, K. Iskra, D. Kimpe, S. Lang, S. Lang, R. Latham, R. Ross, L. Ward, and P. Sadayappan. Scalable i/o forwarding framework for high-performance computing systems. In *IEEE Internation Conference on Cluster Computing and Workshops*, 2009.

[9] J. Bent, B. Settlemyer, N. DeBardeleben, S. Faibish, D. Ting, U. Gupta, and P. Tzelnic. On the non-suitability of non-volatility. In *7th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 15)*, 2015.

[10] F. Chen, D. Koufaty, and X. Zhang. Hystor: making the best use of solid state drives in high performance storage systems. In *Proceedings of the International Conference on Supercomputing*, 2011.

[11] N. Gaffney, C. Jordon, T. Minyard, and D. Stanzione. Building wrangler. In *IEEE International Coneference on Big Data*, 2014.

[12] S. He, X.-H. Sun, and A. Haider. Has: Heterogeneity-aware selective layout scheme for parallel file systems on hybrid servers. In *IEEE International Parallel & Distributed Processing Symposium*, 2015.

[13] S. Mickelson. Optimizing workflow for cesm. www.cesm.ucar.edu/events/ws.2015/presentations/sewg/, 2015.

[14] K. Ren, Q. Zheng, S. Patil, and G. Gibson. Indexfs: Scaling file system metadata performance with stateless caching and bulk insertion. In *SC*, 2014.

[15] S. Strande, P. Cicotti, R. Sinkovits, W. Young, R. Wagner, M. Tatineni, E. Hocks, A. Snavely, and M. Norman. Gordon: design, performance, and experiences deploying and supporting a data intensive supercomputer. In *XSEDE*, 2012.

[16] B. Xie, J. Chase, D. Dillow, O. Drokin, S. Klasky, S. Oral, and N. Podhorszki. Characterizing output bottlenecks in a supercomputer. In *SC*, 2012.