2016 IEEE 22nd International Conference on Parallel and Distributed Systems

RESS: A Reliable Energy-Efficient Storage System

Shu Yin*†‡, Zhaoyu Xiao*, Kenli Li*†, Jianzhong Huang§, Xiaojun Ruan¶, Xiaomin Zhu‖ and Xiao Qin**
*College of Computer Science & Electronics Engineering, Hunan University, China, Email: {shuyin, lkl}@hnu.edu.cn
†National Supercomputing Center in Changsha, China
‡Dept. of Computer Science, National University of Defense Technology, China
§Dept. of Computer Science, Huazhong University of Sci. and Tech., China, Email: hjzh@hust.edu.cn
¶Dept. of Computer Science, West Chester Uni. of Pennsylvania, West Chester, PA 19383, Email: xruan@wcupa.edu
‖Sci. & Tech. on Info. System Eng. Lab, National University of Defense Technology, China, Email: xmzhu@nudt.edu.cn
**Dept. of Computer Science & Software Engineering, Auburn University, AL 36849, Email: xqin@auburn.edu

***Abstract*—Extracting high I/O performance from parallel file systems is no longer the only goal in modern data centres. As issues of the Energy Wall and the Reliability Wall become unavoidable, it is a demanding and challenging task to reduce energy consumption in large-scale storage systems in modern data centres while retaining acceptable systems reliability. Most energy conservation techniques inevitably have adverse impacts on the parallel disk systems. To address the reliability issues of energy-efficient parallel storage systems, we propose a reliable energy-efficient storage system called *RESS*, which aims at improving both energy efficiency and reliability of parallel storage systems by seamlessly integrating HDDs and SSDs. At the heart of the *RESS* is a transformative middleware layer, which reorganizes the I/O workload for the underlying parallel file systems. With the help of the middleware layer, *RESS* can distribute popular data to SSDs and put HDDs into the low-power mode under light workload conditions without modification of the parallel systems.**

***Keywords*-parallel storage system, PVFS, PLFS, energy-efficient, reliability**

I. Introduction

Traditional energy-saving techniques in disk array systems do not take into account adverse impacts of energy conservation mechanisms on system reliability. Existing solutions tend to frequently spin up and down hard disk drives (HDDs) or replace HDDs with solid state disks (SSDs). These solutions may cause severe mechanical malfunction of HDDs and lead to permanent data loss. Moreover, heavy write I/O loads shorten the lifetime of SSDs due to the limited numbers of erasure cycles of flash memory. SSDs are more expensive than HDDs; therefore, it is non-cost-effective to build a large-scale storage system using SSDs. In this paper, we propose a hybrid parallel storage system call *RESS* aiming to improve reliability and energy-efficiency of storage systems by introducing a middleware I/O layer so as to apply the hybrid disk technique without modification of existing file systems. At the heart of *RESS* is the hybrid storage technique that exploits flash-based storage devices' low power consumption as well as tradition disk-based storage appliances' large capacity and long lifetime to improve energy efficiency and reliability of parallel storage systems under fluctuate I/O workloads.

A. Motivations

The following five factors motivate us to develop the *RESS* system that is reliable and energy efficient.

- the high energy consumption of storage systems in data centres;
- the challenge of improving energy efficiency of parallel storage systems;
- the benefit of integrating solid state disks and hard drives;
- the adverse impacts of existing energy-saving techniques on parallel storage system reliability; and
- the difficulty in modification the existing parallel file systems.

Motivation 1. Increasing evidence indicates that as much as 27% of the energy in a modern data center is consumed by storage devices [2]. Even worse, such a fraction tends to go up as data storage capacity is dramatically rising by 60% annually [14]. Energy spent to operate disks leads to high heat dissipation, which poses a serious obstacle to the development of energy-efficient cooling systems. Improving energy efficiency of storage systems can substantially reduce operating costs of large-scale data centers; reducing energy consumption of storage systems offers potential economical and environmental benefits [5].

Motivation 2. Data centers that consume significant amount of energy deploy reliable RAID systems to meet the growing requirements of capacity and I/O performance. This trend makes it appealing and practical to improve the energy efficiency of RAID devices through the development of promising energy-conservation approaches. Due to the data-stripping mechanism, each disk in a RAID system stores only a portion of files; thus, all disks have to be power-on to service I/O requests even under light workload conditions. Such characteristics of RAIDs make conventional energy-saving techniques, such as dynamic power management and workload skew techniques, inadequate for RAID systems.

Motivation 3. A few existing approaches build energy-efficient RAIDs for data archival systems [7], whereas others make good tradeoffs between performance and energy efficiency [13] [21]. Controller-level cache management schemes leverage skew striping patterns [15] to reduce the number of power-on disks to meet system loads in RAID systems [10]. NAND-flash-based solid state disks (SSDs) are becoming popular storage devices thanks to their significant advantage of good shock resistance, fast data accessing, and lower energy consumption. However, SSD's drawback in low cost-efficiency and short lifetime prevent them from being widely applied to large data centers. Recent studies show that the erasure limit of SSDs is typically rated at 5,000 to 10,000 cycles per block, meaning that an SSD could be worn out within a few months under heavy write loads [1]. Hybrid techniques like HRAID6ML [20], HPDA [12], and CacheRAID [6] embrace the performance benefit of flash-based storage and the low cost of hard drives.

Motivation 4. Existing energy conservation techniques can yield significant energy savings in disks. While several energy conservation schemes like cache-based energy-saving approaches normally have marginal impact on disk reliability, many energy-saving schemes (e.g., dynamic power management and workload skew techniques) inevitably have noticeable adverse impacts on storage systems [3] [16]. For example, dynamic power management (DPM) techniques save energy by using frequent disk spin-downs and spin-ups, which in turn can shorten disk lifetime [18] [19].

Motivation 5. Existing parallel file systems are very large software

1521-9097/16 $31.00 © 2016 IEEE
DOI 10.1109/ICPADS.2016.156

IEEE computer society

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 23:04:08 UTC from IEEE Xplore. Restrictions apply.

projects that must support general use cases and ensure data integrity. In many cases, they support full POSIX compliance, which is not necessary for HPC checkpoint workloads. Relaxation of POSIX semantics or reorganization of I/O workloads within these parallel file systems can negatively impact the entire range of their supported workloads. Additionally, those sophisticated parallel file systems- take PVFS and Lustre as examples- usually have at least 200,000 lines of code, which set obstacles for modification. Lastly, any modifications should be carefully addressed based on the knowledge of global picture of the file systems. Otherwise a simple modification on a file system module could sabotage the stability of the whole system. Due to such reasons, it is very difficult to find a large-scale system for the real-world testing since barely no administrators dare to gamble systems stability on implementing an experimental project.

Previously, a reliable energy efficient disk systems called REED was designed to deal with the systems reliability and power conservation issues [17]. Such mechanism combines HDDs and SSDs to reduce negative effects on system reliability while reducing power consumption. It aims to squeeze the hard drives' benefit in high cost-efficiency, high capacity, and long lifetime as well as making use of SSDs as the replica storage to take place sets of hard drives during light I/O workload. According to the preliminary result, REED could achieve up to nearly three quarters of power conservation while maintaining reasonable reliability under the experimental setups. However, due to the aforementioned reasons, REED was only measured with the help of the DiskSim [9], which is a professional disk simulator. Our proposed *RESS* aims at the implementation of real systems via middle-ware I/O layer based on the achievements of power conservation and reliability maintaining.

The benefits of *RESS* are four-fold.

- As *RESS* uses HDD-based storage nodes as the primary storage method, it provides an un-comparable capacity budget for data centers;
- *RESS* offers a better balance among energy consumption, reliability, and hardware budget under a fluctuating system workload;
- *RESS* retains a better system reliability compared to flash-based RAID systems and provides a practical solution during the transition period from full-HDD storage to full-memory storage.
- As *RESS* reorganizes the I/O workload using a middleware layer, it can be implemented without additional modification of existing file systems.

The remainder of this paper is organized as follows. In Section II, we provide the background of REED and PLFS. The architecture and design of *RESS* is discussed in Section III. Section IV presents experimental results as well as performance analysis. Finally, Section V concludes the paper with future research directions.

II. Background

A. REED: A Reliable Energy-Efficient RAID

REED is a framework developed to offer high system reliability and energy efficiency by allocating data stripes to disks [17]. The architecture is composed of both SSDs and HDDs, among which HDDs service as a primary storage whereas SSDs perform as a caching mechanism for data replicas. Recall that the goal of REED is to reduce system energy consumption under light workload. The number of SSDs and the capacity of SSDs are determined by the performance and space requirements of applications running in REED. Obviously, increasing the number of SSDs in REED leads to a large storage system with an improved I/O performance; on the other hand, an excessive number of SSDs inevitably makes REED less cost effective. REED consists of four key functional modules including a workload monitor, a director, an SSD manager, and a disk manager. The workload manager of REED is responsible for monitoring I/O access patterns and dynamically measuring system workload. Other than the monitor, the director is in charge of determining the number of HDDs that are in the active mode; the director judiciously determines which SSDs should be involved when a subset of disks are powered off during light system load. Importantly, the director is managing I/O distributions among active HDDs and proactively loading the replicas of data strips from HDDs to SSDs. When the system runs under heavy workload, REED relies on the original RAID configuration to preserve high I/O performance. In case of light I/O load, REED identifies the number of power-on HDDs and SSDs while shutting down a subset of HDDs to boost energy efficiency.

To improve reliability of energy-efficient RAID, we ensure that REED is capable of tolerating disk failures. In energy-efficient RAIDs, life expectancy of server-class disks is likely to be reduced due to power-state transitions. This problem is addressed in REED by facilitating RAIDs with data redundancy (i.e., placing replicas of popular data into SSDs).

REED can be envisioned as an intermediate layer sitting above a software RAID, which is supported by the operating system and physical storage devices. Thus, REED inherits data redundancy, striping granularity, and disk layouts. For example, the REED-0 system, which is a reliable energy-efficient RAID-0, is able to reconstruct a failed disk from parity blocks.

One challenge we are facing is how to maintain a high reliability of SSDs in REED. SSDs have limited erasure cycles; frequent data updates under light I/O load may shorten the lifetime of REED when it is in the SSD-HDD hybrid state. Furthermore, an excessive number of power-state transitions sabotages the reliability of hard disks. For example, suppose the transition frequency is 300 per month, the annual failure (AFR) rate is increased by 0.13% [18].

In order to maintain high reliability of SSDs, REED limits the number of erasure cycles of SSDs by disabling the SSDs under heavy I/O load. Enormous number of writes issued under high workloads tends to substantially shorten the lifespan of SSDs. Reducing the number of erasure cycles is achieved by keeping SSDs active when workload conditions are light. Apart from improving the reliability of SSDs, REED is capable of maintaining high reliability of HDDs. Reliable HDDs become possible in REED, because the number of power-state transitions of HDDs are reduced by SSDs under light and fluctuating workloads.

B. PLFS: Parallel Log-structured File System

PLFS is a middleware that sits between the user application and a file system that is parallel in our case and is subsequently referred to as the underlying parallel file system .PLFS creates a *container* on the underlying parallel file system that is a physical directory sharing the same name as the logical file stored into PLFS [4]. The logical view of the file matches the I/O access pattern of the application. Within the container, there are multiple files and directories: an *access* file holds ownership and access control information, a *metadir* caches information about the logical file size, an *openhost* directory contains information about any process that currently has a PLFS file open for write access, and finally there are *subdirs* that contain the process specific *data* and *index* log files.

The major difference between PLFS and current parallel file systems is that current parallel file systems do a large amount of work during the write phase of file I/O to place logical bytes deterministically. Primarily they do so for two main reasons: one,

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 23:04:08 UTC from IEEE Xplore. Restrictions apply.

to serialize writes to the same offsets, and two, to quickly lookup data offsets for reads. In contrast, PLFS uses timestamps to resolve writes to the same offsets and defers the work of creating offset maps until the file is opened for read access. In effect, PLFS has deferred the resolution of the file from the write phase to the read phase. This deferral introduces several complications for large-scale I/O workloads and we have investigated and developed solutions to these complications in this work.

In order to take the full advantages of both flash-based and traditional disk-based storage devices, we propose a solution called *RESS* that fully utilizes traditional disk-based storage nodes under heavy workload conditions and hybrid storage nodes (i.e., SSDs and HDDs are integrated) in light workload scenarios. When the workload of a storage system is heavy, no energy-saving mechanism can be applied to conserve energy by placing disks into the low-power mode. In this case, *RESS* leverages HDDs nodes as primary storage nodes to achieve high I/O capacity and data reliability; *RESS* also enlarges the lifetime of SSDs by keeping SSDs nodes in the idle state to avoid frequent updates on SSDs. Under light workload, SSDs nodes are in active mode to perform as a cache and a buffer for HDDs nodes that are transitioning into the low-power mode to conserve energy. In doing so, SSDs boost the reliability of HDDs by eliminating the number of unnecessary power-state transitions in HDDs.

III. RESS: A Reliable Energy-Efficient Storage System

REED achieved its preliminary goals in power conservation and retaining system reliability. However, its disk-level design makes the implementation on real-world systems very challenging. There are majorly three levels to address the implementation challenge: 1) user level – modifying the I/O pattern on the application level. However, changes on application levels usually are nearly impossible because every application would need to tuned for a specific file systems, which requires rich experience or even expert skills in computer architecture or operating systems; 2) file system level – modifying the existing file systems. Unfortunately, modifications on file systems would be prohibitive because every not-yet-fully-tested trial may lead the systems to instability or even crash. As complex system software projects, the size of exiting parallel file systems can easily reach to hundred thousand lines of code. A simple update could make the systems on risk, let alone a energy-efficiency mechanism that involves data distribution, data migration and power state transitions of devices; 3) middle-ware level – interposes a layer between the application level and the file system level. Such layer preserves the users' view of files and transforming the I/O into a manageable pattern for lower level file systems. As the middle-ware layer is transparent to both users and file systems, the modification becomes easier since neither of the side will be involved. Since the middle-ware layer servers a narrow focus and is light-weighed in terms of its size, it delivers portability in addition to the convenience in the layer development and debugging.

A. RESS Architecuture

As an extension of REED, *RESS* modifies and stacks a middle-ware I/O layer, called PLFS, on top of PVFS parallel file system for the implementation purpose. The architecture of *RESS* is expressed in Fig. 1.

Similar to REED, *RESS* consists of the workload monitor, the director and the managers, which are responsible for keeping track of I/O access patterns, determining the numbers of active HDD nodes and managing the I/O distributions among active HDD nodes and proactively loading replicas to SSD nodes. Instead of sitting lower

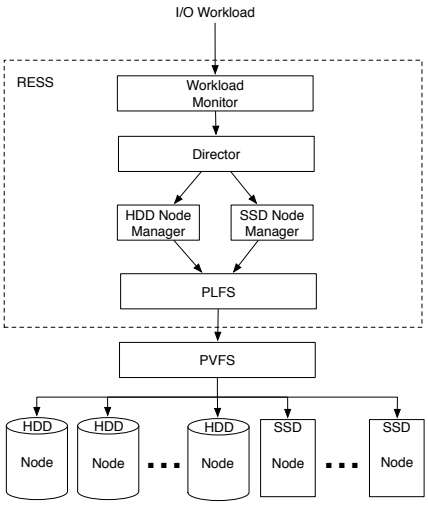


Fig. 1: The Architecture of RESS

than RAID level, *RESS* is leveraged to the level that is above the existing file system. This change makes the implementation on the real systems feasible and easier. In order to pass the reorganized I/O workload to the file systems, *RESS* applies the I/O middle-ware layer – PLFS and connects to the file system.

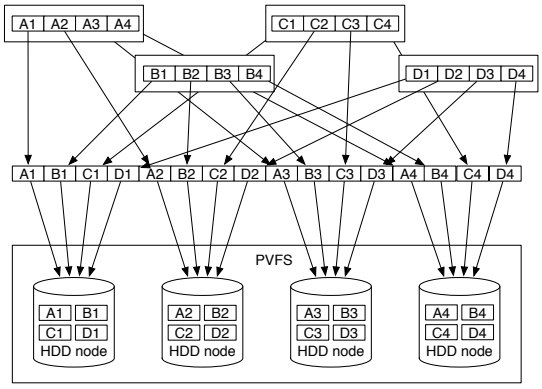
B. I/O Re-Organizaiton

The original design purpose of PLFS is to improve the I/O performance via transformation from N-N pattern to N-1 pattern while avoiding the serializations. By decoupling a file into non-shared component pieces PLFS eliminated serializations on the underlying parallel file system that were responsible for the large performance discrepancy between N-1 and N-N I/O workloads. In other words, I/O pattern was re-organized by PLFS before it is forwarded to the actual file systems. With the help of a *container* structure, this re-organization process transforms a virtual serial file (see the file *checkpoint1* in the PLFS Virtual Interposition Layer) into a group of parallel subsets and redirects each subset to the target storage node. Thanks to PLFS's transparence to the actual underlying parallel file system, the real file system process the I/O request without realizing that the order of the I/O pattern is altered. This inspires us that, if we could utilize the I/O re-organization characteristic of PLFS, we can use such the middle-ware layer to redirect blocks or data to destination node. With the modification of PLFS, we could determine the target node for all the blocks or data that belongs to the hot data and aggregate the data into, for example, SSD nodes under the light workload. Fig. 2 show a example of original I/O pattern and the re-organized I/O pattern after the tuned PLFS. We can see in Fig. 2(b) that with the small amount portion of data re-organization, the hot data (for file A in the example) will be aggregated in SSD node, so that under the light workload, some of the HDD nodes can be switched to idle mode.

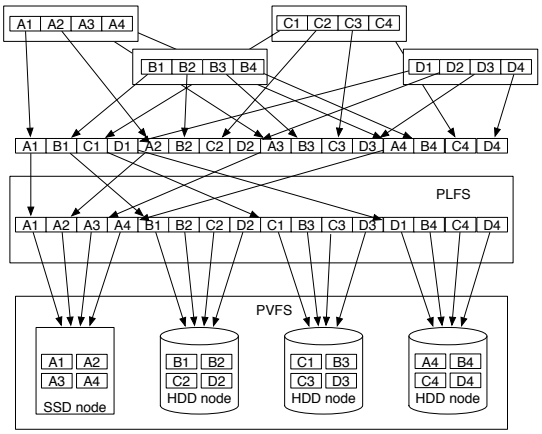
C. Energy-Efficency

RESS leverages flash-based storage nodes, which are highly energy efficient, to cache and buffer data stripes in a skewed fashion. In doing so, *RESS* strives to push a subset nodes into low-power state (e.g. idle) while maintaining the file system organization. When hard disk nodes are switch to idle state, *RESS* services I/O requests via a combination of blocks residing on power-on disk nodes as well as

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 23:04:08 UTC from IEEE Xplore. Restrictions apply.



(a) Original I/O Pattern



(b) I/O Pattern Re-Organization

Fig. 2: Original I/O Pattern vs. Re-Organized I/O Pattern

on data replicas residing on the SSD nodes. *RESS* identifies blocks that have been frequently accessed (a.k.a., hot data); replicas of hot data blocks are placed and managed into SSD nodes. Hence, in the case where a few disk nodes are in idle state, requests accessing hot data can naturally and quickly be served by SSD nodes. Evidence begins to accumulate showing that when workload is light, requests are likely to access hot data. Although a request might access data that are less popular than hot data, disk nodes may be temporarily switched to active state to provide an I/O service under the light load. Recently accessed cold data will be cached into SSD nodes after disk nodes are switched back to idle again, thereby reducing unnecessary disk power- state transitions when the cold data is accessed in the not- long-distant future.

By replicating an appropriate amount of data strips to SSD nodes based on system workload, *RESS* aims to aggressively conserve energy consumption by reducing the number of disk nodes that are in the active mode. *RESS* heritages REED's high energy efficiency thanks to SSDs low power consumption, which is one third of that of hard drives or HDDs. Additionally, since the *RESS* switches the nodes instead of disks into idle mode, the energy conservation is so significant that the expensiveness of SSDs could be counteracted.

D. Reliability

To improve reliability of parallel storage systems, we ensure that *RESS* is capable of tolerating disk failures. Previous studies indicate that life expectancy of server-class disks is likely to be reduced due to power-state transitions [19]. Parallel file systems such as PVFS originally does not replicate data as it is assumed to be running on top of a storage system where RAID-like mechanisms are applied on each node. However, PVFS does not tolerate the inter-node data failure as the organization of PVFS nodes are configured as a RAID-0, only that the storage unit is a node in stead of a disk. This problem is addressed in *RESS* by facilitating PVFS with data redundancy (i.e., placing replicas of popular data into SSDs nodes).

One challenge we are facing is how to maintain a high reliability of SSD nodes in *RESS*. SSDs have limited erasure cycles; frequent data updates under light I/O load may shorten the lifetime of *RESS* when it is in the SSD-HDD hybrid state. Furthermore, an excessive number of power-state transitions sabotages the reliability of hard disk based nodes. For example, suppose the disks transition frequency is 300 per month, the annual failure (AFR) rate is increased by 0.13% [17].

In order to maintain high reliability of SSDs, *RESS* limits the number of erasure cycles of SSDs by disabling the SSD nodes under heavy I/O load. Enormous number of writes issued under high workloads tends to substantially shorten the lifespan of SSDs. Reducing the number of erasure cycles is achieved by keeping SSDs active when workload conditions are light. Apart from improving the reliability of SSD nodes, *RESS* is capable of maintaining high reliability of HDD nodes. Reliable HDDs become possible in *RESS*, because the number of power-state transitions of HDDs are reduced by SSD nodes under light and fluctuating workloads.

IV. Experimental Results

A. Experimental Setup

In our experiments, we focus on the comparisons between *RESS* and original PLFS+PVFS file systems, which achieves high I/O performance by bridging the gap between the logical N-1 workload and the physical N-N workload. The drawback of PLFS-PVFS lies in the lack of a node-level data recovery mechanism.

During the process of evaluation, we focus on energy savings as well as system reliability. Since the original PLFS+PVFS system is designed to improve I/O performance, we consider the read performance as well by testing the read open time and the read bandwidth. Table I summarizes the configuration parameters of the file system and the disk systems underneath used throughout our experiments. The tests are implemented on a five-node cluster in our lab. The file system we are using is OrangeFS, and each node in the cluster consists of two 1TB hard disk drive while node 5 has an extra solid state disk as a cache disk performing under light workload.

B. I/O Benchmark Performance Analysis

In addition to the power consumption and reliability issues, Figs. 3-6 illustrate the general I/O performance of the *RESS* using MPI-IO Test benchmark, which is a synthetic I/O workload generator developed at Los Alamos National LAb [8]. In addition to the original PLFS+PVFS setup, we compare the read-open-time, read-bandwidth of RESS with a modified PLFS design called IndexFlatten, which was designed to improve the PLFS read performance [11].

The results show that the read performance of *RESS* is similar to the original PLFS and modified PLFS designs in a general way. It is noticed that as the number of process or concurrent I/O increase, the open time of *RESS* (PLFS-EE in figures) is slightly better than both the PLFS designs. This is majorly because that the hot data is buffered in the SSD node in advance. This difference heavily depends on the sophistication of the hot/cold data identifying algorithms.

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 23:04:08 UTC from IEEE Xplore. Restrictions apply.

TABLE I: File System and Disk System Parameters

Operating System	Debian MINT 32-bit
File System	OrangeFS 2.4.8
Node Quantity	5
Node Arrangement	HDD node *4 SSD node *1
Average Power per Node	400W
Hard Disk	
Brand	West Digit
Capacity	1TB
Quantity of Devices	10
Data Transfer Rate	126MB/s (MAX)
Solid State Disk	
Brand	Samsung
Capacity	120 GB
Quantity of Devices	1
Data Transfer Rate	Read: 540MB/s(MAX) Write: 520MB/s(MAX)

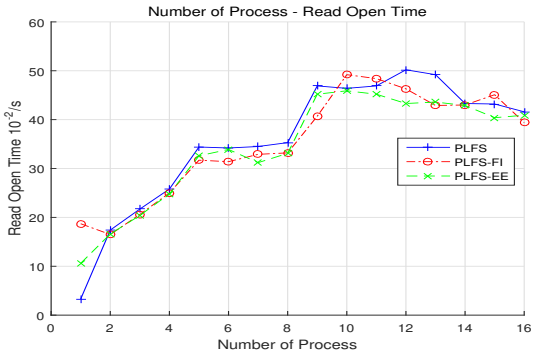


Fig. 3: Read Open Time Comparison vs. Number of Process

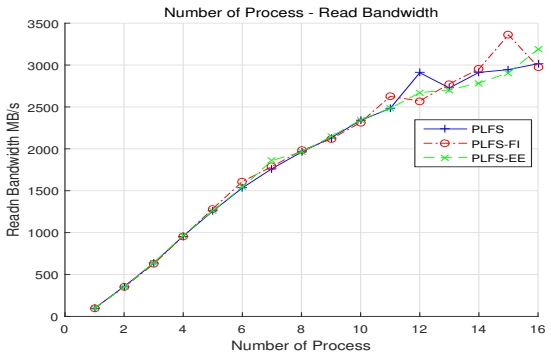


Fig. 4: Read Bandwidth Comparison vs. Number of Process

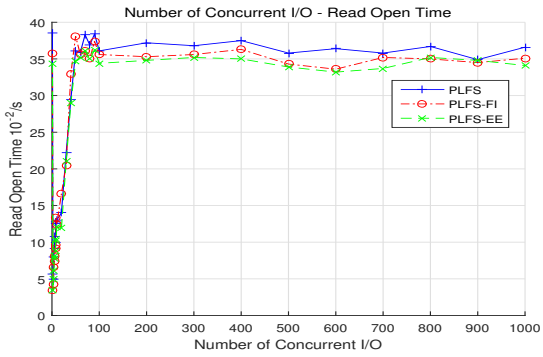


Fig. 5: Read Open Time Comparison vs. Number of Concurrent I/O

V. Conclusion

In this paper, we designed a parallel storage system called *RESS*, which is highly reliable and energy-efficient in nature. As an extension of the previous work REDD, *RESS* leverage the power management above the existing file systems via a I/O transformative layer. By forming a hybrid system with SSD nodes and HDD nodes, *RESS* pushes HDD nodes to idle mode under light workload to achieve energy consumption. Importantly, *RESS* heritages the major goal of the previous REED design to improve energy efficiency while maintaining good reliability of hard and solid state drives. More importantly, *RESS* achieves the implementation on a real file system – PVFS – and make the large-scale real world testing possible.

In the not-too-distant future, we plan to address the following open issues. First of all, the power consumption analysis of *RESS* should be more sophisticated. Current study only uses home-use power meter to monitor the power change, which leads to inaccuracy to the result. Since the home-use tool does not support continually monitoring of the system power consumption, the current power consumption analysis only represents the average case in a certain time phase. Second, we intend to design an online algorithm to automatically identify the hot data and cold data. The goal of the online algorithm is to reduce power management overhead and enhance the efficiency and accuracy for data replication. Last but not the least, besides I/O intensive workloads, the up-scale testing will be addressed in *RESS*.

VI. Acknowledgment

Shu Yin's research is supported by the National Natural Science Foundation of China under Grant 61402158, the Hunan Province Natural Science Foundation under Grant 14JJ4025, the Scientific Research Foundation for the Returned Overseas Chinese Scholars under Grant [2013]1792, Ministry of Education of the People's Republic of China, the China Postdoctoral Science Foundation under Grant 2015M572708, and Hunan University under a start-up grant. Xiao Qin's work is supported by the U.S. National Science Foundation under Grants CCF-0845257 (CAREER), CNS-0917137 (CSR), CNS-0757778 (CSR), CCF-0742187 (CPA), CNS-0831502 (CyberTrust), CNS-0855251 (CRI), OCI-0753305 (CI-TEAM), DUE-0837341 (CCLI), and DUE-0830831 (SFS).

References

[1] M. Balakrishnan, A. Kadav, V. Prabhakaran, and D. Malkhi. Differential RAID. *ACM Transactions on Storage*, (2):1–22, July 2010.

[2] B. Battles, C. Belleville, S. Grabau, and J. Maurier. Reducing data center power consumption through efficient storage. *Netapp white paper*, February 2007.

[3] K. Bellam, A. Manzanares, X. Ruan, X. Qin, and Y.-M. Yang. Improving reliability and energy efficiency of disk systems via utilization control. In *Proc. IEEE Symp. Computers and Comm.*, 2008.

[4] J. Bent, G. Gibson, G. Grider, B. McClelland, P. Nowoczynski, J. Nunez, M. Polte, and M. Wingate. Plfs: a checkpoint filesystem for parallel applications. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, pages 1–12, Nov 2009.

[5] A. Bianzino, C. Chaudet, D. Rossi, and J. Rougier. A survey of green networking research. *Communications Surveys Tutorials, IEEE*, 14(1):3–20, 2012.

[6] T.-Y. Chen, T.-T. Yeh, H.-W. Wei, Y.-X. Fang, W.-K. Shih, and T.-s. Hsu. CacheRAID: An Efficient Adaptive Write Cache Policy to Conserve RAID Disk Array Energy. In *2012 IEEE Fifth International Conference on Utility and Cloud Computing*, pages 117–124, Nov. 2012.

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 23:04:08 UTC from IEEE Xplore. Restrictions apply.

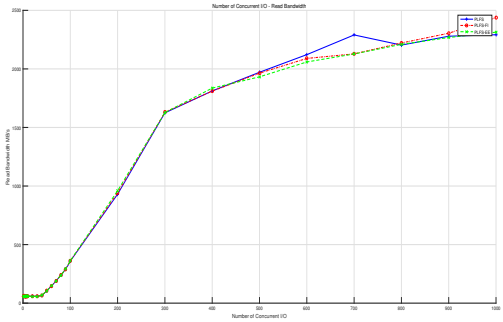


Fig. 6: Read Bandwidth Comparison vs. Number of Concurrent I/O

[7] D. Colarelli and D. Grunwald. Massive arrays of idle disks for storage archives. In *Proc. ACM/IEEE Conf. Supercomputing*, pages 1–11, 2002.

[8] G. Grider, J. Nunez, and J. Bent. Lanl mpi-io test. http://institutes.lanl.gov/data/software/, July 2008.

[9] S. W. S. John S. Bucy, Jiri Schindler and G. R. Ganger. The disksim simulation environment version 4.0 reference manual. 2008.

[10] D. Li and J. Wang. Eeraid: energy efficient redundant and inexpensive disk array. In *Proceedings of the 11th workshop on ACM SIGOPS European workshop*, EW 11, 2004.

[11] A. Manzanares, J. Bent, M. Wingate, and G. Gibson. The power and challenges of transformative i/o. In *2012 IEEE International Conference on Cluster Computing*, pages 144–154, Sept 2012.

[12] B. Mao, H. Jiang, S. Wu, L. Tian, D. Feng, J. Chen, and L. Zeng. HPDA: a hybrid parity-based disk array for enhanced perforamnce and reliablity. *ACM Transactions on Storage*, (1):1–20, Feb. 2012.

[13] E. Pinheiro and R. Bianchini. Energy conservation techniques for disk array-based servers. In *Proc. 18th Int'l Conf. Supercomputing*, 2004.

[14] E. Pinheiro, R. Bianchini, and C. Dubnicki. Exploiting redundancy to conserve energy in storage systems. *SIGMETRICS Perform. Eval. Rev.*, 34(1):15–26, June 2006.

[15] C. Weddle, M. Oldham, J. Qian, A.-I. A. Wang, P. Reiher, and G. Kuenning. Paraid: a gear-shifting power-aware raid. In *FAST '07: Proceedings of the 5th USENIX conference on File and Storage Technologies*, pages 30–30, Berkeley, CA, USA, 2007.

[16] T. Xie and Y. Sun. Sacrificing reliability for energy saving: Is it worthwhile for disk arrays? In *Proc. IEEE Symp. Parallel and Distributed Processing*, pages 1–12, April 2008.

[17] S. Yin, X. Li, K. Li, J. Huang, X. Ruan, X. Zhu, W. Cao, and X. Qin. REED: A reliable energy-efficient RAID. In *44th International Conference on Parallel Processing, ICPP 2015, Beijing, China, September 1-4, 2015*, pages 649–658, 2015.

[18] S. Yin, X. Ruan, A. Manzanares, and X. Qin. How reliable are parallel disk systems when energy-saving schemes are involved? In *Proc. IEEE International Conference on Cluster Computing (CLUSTER)*, 2009.

[19] S. Yin, Y. Tian, J. Xie, X. Qin, M. Alghamdi, X. Ruan, and M. Qiu. Reliability analysis of an energy-aware raid system. In *Performance Computing and Communications Conference (IPCCC), 2011 IEEE 30th International*, pages 1–8, 2011.

[20] L. Zeng, D. Feng, J. Chen, Q. Wei, B. Veeravalli, and W. Liu. Hraid6ml: A hybrid raid6 storage architecture with mirrored logging. In *Mass Storage Systems and Technologies (MSST), 2012 IEEE 28th Symposium on*, pages 1–6, 2012.

[21] X. Zhu, C. He, K. Li, and X. Qin. Adaptive energy-efficient scheduling for real-time tasks on dvs-enabled heterogeneous clusters. *J. Parallel Distrib. Comput.*, 72(6):751–763, June 2012.

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 23:04:08 UTC from IEEE Xplore. Restrictions apply.