A New High-performance, Energy-efficient Replication Storage System with Reliability Guarantee

Jiguang Wan[1], Chao Yin[1], Jun Wang[2] and Changsheng Xie[1]

[1] Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, China
e-mails: jgwan@mail.hust.edu.cn, yinchao408@gmail.com, cs_xie@mail.hust.edu.cn

[2] School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL32816, USA
e-mails: jwang@eecs.ucf.edu

Corresponding author: Changsheng Xie

***Abstract*—In modern replication storage systems where data carries two or more multiple copies, a primary group of disks is always up to service incoming requests while other disks are often spun down to sleep states to save energy during slack periods. However, since new writes cannot be immediately synchronized onto all disks, system reliability is degraded.**

This paper develops PERAID, a new high-performance, energy-efficient replication storage system, which aims to improve both performance and energy efficiency without compromising reliability. It employs a parity software RAID as a virtual write buffer disk at the front end to absorb new writes. Since extra parity redundancy supplies two or more copies, PERAID guarantees comparable reliability with that of a replication storage system. In addition, PERAID offers better write performance compared to the replication system by avoiding the classical small-write problem in traditional parity RAID: buffering many small random writes into few large writes and writing to storage in a parallel fashion. By evaluating our PERAID prototype using two benchmarks and two real-life traces, we found that PERAID significantly improves write performance and saves more energy than existing solutions such as GRAID, eRAID.

Keywords - reliability; power consumption; performance; flush disk

I. INTRODUCTION

With the rapid growth of data production in companies, the requirement of storage space grows very largely as well. The large-scale parallel I/O system is widely used in high-performance mass computer systems. Typical applications need large-scale parallel I/O systems to do mass data processing. Due to the scale of parallel I/O systems increasing continuously, the proportion of the energy consumption of I/O system to the total cost of ownership (TCO) grows larger and larger. In the data center, the energy consumption proportion of the disk systems to the whole systems has reached 27% in 2002. While increasing the storage system capacity and reducing the average response time, the energy consumption of storage system will be higher.

Saving power in computer storage often degrades reliability, i.e., trade off reliability for power conservation[1]. In replication storage systems, we often power down one or more disk groups to save energy when the system load is light. Cite power-proportional layout solutions[2], such as EERAID[12], GRAID[14], PARAID[11]. However, due to the deferred writes to be executed on sleep groups in a later time rather than in parallel with active groups, reliability is degraded before these operations are finished.

We have developed an energy efficient replication storage system without compromising reliability by adding a parity software RAID (RAID5, RAID6) as a write buffer. This parity RAID based write buffer is able to buffer many random small writes into few large writes to produce high write performance. More importantly, it does not compromise reliability by writing deferred data to both the primary group and other groups at the same time right after we wake up sleeping groups. This salient feature is without introducing any hardware cost.

Before the new data are flushed to the other non-primary groups, they may lose new data in the event of disk failures. In DCD[3], it uses a small log disk as cache disk, but the cache disk and the data disk are in one disk. Our idea is to use half of the disk and compose a write buffer into RAID5. We use the characteristic of RAID5 to guarantee reliability of the new data in the system, which will be described in Section III. At the same time, we use the logging write technology to solve the write performance.

We designed a flush algorithm to prevent the old data from loss. Because we have used the logging write technology, the same block data will be stored in different place of write cache that the performance of read will be degraded. In order to improve the read speed, we should reduce the read hit. Then, the read operations are turned to the primary groups, and the read speed is improved.

The contributions of this paper are described as follows:

- The introduction and evaluation of the storage system, and its realization called PERAID, capable of providing ideal reliability of new data and old data. It also improves the performance without increasing the energy consumption.
- For high write-to-read-ratio I/O workloads, PERAID could significantly improve write performance by 77% compared to current energy saving solutions (GRAID, eRAID0 etc) when the request is all write request. In addition, it conserves energy consumption by 29.4% compared to RAID10, and 7.7% to GRAID.
- A comprehensive sensitivity study indicates that, PERAID improves performance when increasing write-to-read ratio and under random write workloads, and has degradations for power and energy consumption when increasing write buffer

size. PERAID is also flexible with write buffers made of any type of parity based storage architecture such as RAID5, RAID6.

The rest of this paper is organized as follows. Section 2 describes architecture and the design of PERAID. The experimental result and evaluation is introduced in Section 3. Section 4 describes the related work in disk arrays and motivation. Section 5 is the conclusion.

II. PERAID

A. The Design of PERAID

The power consumption in standby mode is far less than that in active mode and idle mode, but it needs a lot of energy to change from standby mode to active mode. Therefore, considering the energy consumption, we should design a way of organization which makes a part of disk be in standby mode as long as possible, and reduce the transition numbers from the standby mode to active mode as much as possible. The main standard is the average time of system response and bandwidth to measure storage system performance. We should let the parallel system disk be dormant as much as possible, and ensure that there is a good prefetching and caching algorithm at the same time.

When the replica is in standby mode, data will be written to the primary only. The data in the primary will be inconsistent with that of the replicas after a period of time, which will lead to the backup data out of date and reduce the system reliability. A RAID5 write cache is made to enhance the system reliability in PERAID. In small applications, write request is no more than 20 GB in a day. So we consider using a part of the capacity of a RAID5 data disk as a cache. If any disk is damaged, it will recover the data through the log cache and the main disk by this way. When one of the active main disks is damaged, disk array can get complete log data through the rest of the primary to restore lost data with the replicas. So, it can improve the reliability of the system without additional disks.

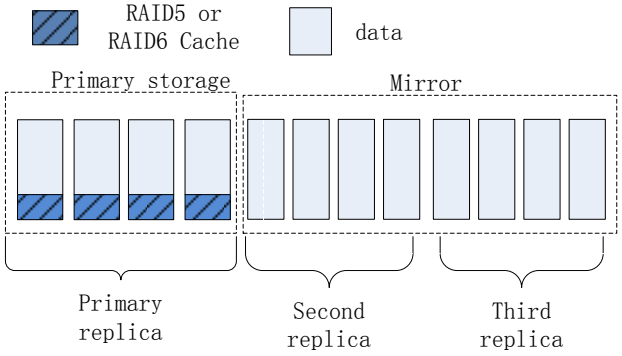


Figure 1. Architecture of PERAID.

Based on the above considerations, the architecture of the disk array is made as Figure 1 shows. Disks are divided into two groups. The first part is disk1, disk2, disk3 and disk4 which are organized into RAID0 as the primary. Another part is eight disks which are organized into two RAID0 as the replicas. In the first four disks, a little space is made of RAID5 as write cache. The replica is usually in standby mode. When read requests come, only the write cache and primary are used. Upon receiving write requests, PERAID send them to write cache of RAID5 at first. When the write cache is full, it will wake up the replicas and flush the data. The third replica is the same as the second one.

B. The Prototype System

This System runs in a server developed from Linux as shown in Figure 2. The module between iSCSI-target module and MD module is what we need to design and implement, which is responsible for the energy saving of the RAID10 between processing.

The iSCSI-target module is responsible for the transformation of iSCSI and the construction of the target. A RAID10 energy saving processing module which is using RAID arrays of redundant information through part or all of the disk in the mirror disks goes into standby mode to save energy. When the mirror disk goes into standby mode, read requests of mirror disks will be redirected to the matching main disk plate while write requests will be written to the controller buffer, which will wait until the mirror disk turns into active mode for writing. MD module manages disk array and realizes software RAID by virtual block equipment. The system will offer software RAID for the data disk and cache.

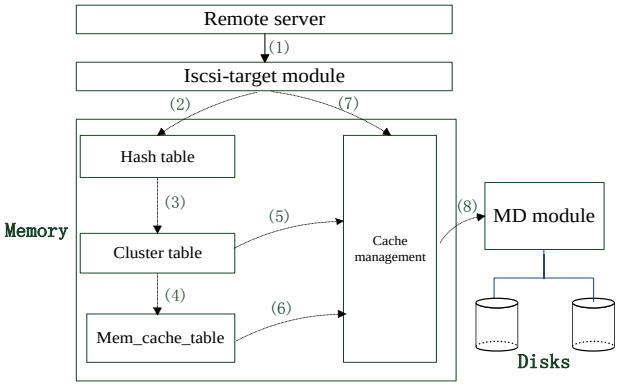


Figure 2. The Prototype system of PERAID.

When PERAID receives a write request, it will send it to the iSCSI-target module, as shown in step 1. Then it refers to the hash table in step 2. During step 3, if the search hits, it will get the cluster that the current block is in now. Otherwise, it will get the current cluster. If the hash hits, it inquires the cluster data whether in memory or not. If it is in memory, it calculates the value of pos, copies the memory, updates the map simultaneously in step 4, and enters cache management in step 6. If it is not in memory, it redistributes a hash node and inserts it into the same block in the list in step 5. If the hash table is a miss, it inserts hash node, at the same time it updates the information in step 7. It then calculates the value of pos, copies memory, and updates the map simultaneously. Finally, it inquires the cluster whether it is used up or not. If the number of used clusters is more than an 80% threshold value, it wakes up the thread to pour disks.

When a read request is made, PERAID will refer to the hash table at first, too. If the search hits, PERAID will read the data from the cache, otherwise it will read the data from main disk. If the search is a miss, it will get the current cluster to redistribute a hash node and refresh the node information. The other operation is the same as receiving a write request.

C. Data Structure

PERAID uses a hash table to keep the relationship between cache and data disks as shown in Figure 3. Considering the reliability of the system, the hash table whose size is 4 KB is stored in chain of NVRAM. The main variables are explained below:

Cluster_id records the cluster number of block;

Mem_cache_id records the index of the memory cache list;

Clu_offset cluster is the offset in the cluster. It can fix the position of the data in the cache through the cluster number and the offset of the cluster in order to search conveniently.

Map is an optional unit of eight mapping. There are eight sectors and each sector records in 1 bit. When the value is one, it means it is effective and the data in the sector is new. If it is zero, the things are opposite.

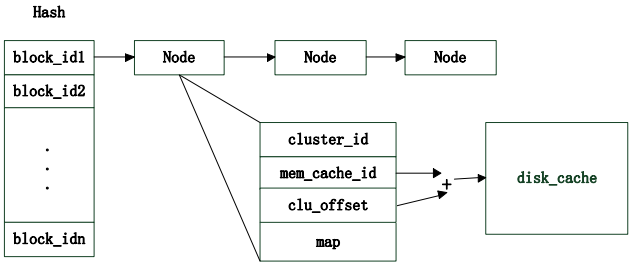


Figure 3. The structure of hash table.

D. Flush Disk

PERAID can improve the write performance in logging writes, but it will reduce read performance. It needs to read the same LBA data from different positions of the cache and combine them into read request data while having read hits. In order to improve read performance, we need to reduce read hits. We should flush the block which has been read many times first to reduce the read time. The principle of flush disk is described as follows:

- Release the buffer space.
- Flush disk read more than once and reduce read hits.
- Flush the block which has been written many times preferentially.
- Follow the spatial locality.

Cache sets are organized in the form of stripes, which is divided into blocks of 4k bytes. Data is written to cache sets according to addresses from low to high, in order. So we should choose the right stripe to flush. The data to be flushed is put on the memory at first, and it is reorganized in memory before being written to disk. Our algorithm is better than FIFO and LRU to do this. We can analyze the advantage of our algorithm from the flush process, which is described as follows:

- Choose the appropriate stripe. We choose the highest frequency stripe which contains the blocks to be read and written many times. At the same time, we should select the stripe which shares the same blocks with the selected stripe. If we use the algorithm of FIFO, we should then choose the stripe one by one, which will lead the highest frequency stripe not to be flushed. The LRU is just opposite, which chooses the stripes to visit the lowest recently. It will be contrary to our original intention.
- Reduce the data reconstruction in memory window. We will consider the stripes only inside the window. We should reduce the I/O numbers of the flush disk and reduce the movement of the disk head. We calculate the correlation of stripes through iteration, which will be described clearly in the next section. At this point, neither FIFO nor LRU will be considered.

III. EVALUATION METHODOLOGY

A. Experimental Setup

PERAID is an implementation based on the Linux soft RAID. Here, we will introduce all parameters for our PERAID prototype and the configuration of our baseline systems. We compare PERAID with GRAID, standard RAID10 and ERAID. We implement PERAID with 8 disks to compose RAID10, each primary disk reserves 20 GB to perform as four members of RAID5 as a write cache. We implement GRAID with 4 primary disks, 4 mirror disks and 1 log disk. We implement RAID10 and ERAID with 4 primary disks and 4 mirror disks. When receiving requests normally, it means that there is no operation of flushing the disk. There are 4 disks running in our system, 5 disks running in GRAID, 8 disks running in RAID10 and 4 disks running in ERAID. When flushing the disk, all the disks will run. The performance evaluation is conducted on a platform of server-class hardware with an Intel(R) Xeon(R)5110 1.60GHz processor and 2G DDR memory. In the system, the disk module is 500GB hard disk and runs in Fedora Linux 8 i386 operating system, which connect in 1000M Ethernet card.

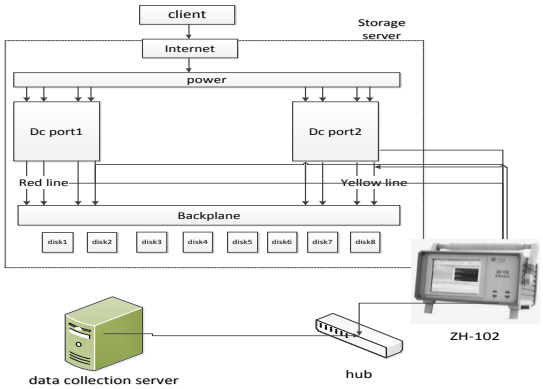


Figure 4. The topological structure of PERAID.

We use the ZH-102 portable wave analysis device to test the energy consumption. When testing, we connect the direct current (DC in short) port of the ZH-102 with the power supply circuit in the storage server disk. Through the Ethernet, the ZH-102 sends data to a data collection server. A storage server, the ZH-102 and the data collection server connection topological structure are shown in Figure 4. Among them the yellow line voltage is 12V and the red line voltage is 5V.

We use three true traces collected from the real work environment as loaded to evaluate the performance and energy consumption. They are financial-1 and financial-2, which is described in TABLE I. In the client, we use the trace tools btreplay to replay the trace. Through the replay, the request in the trace is sent to PERAID in the form of iSCSI request.

TABLE I. SPC TRACE INFORMATION

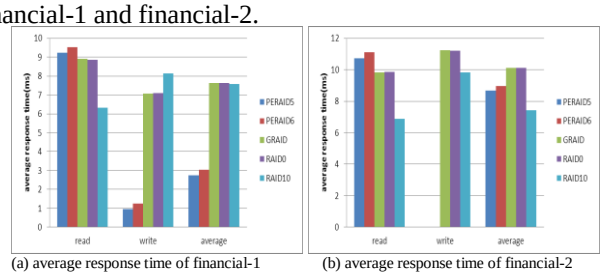
Trace File	Write Request Ratio	Average Request Size (KB)	Total Request Number
Financial-1	76.84%	3.38	5，334，987
Financial-2	17.65%	2.39	3，699，195

B. Experimental results

1) Performance comparison

a) Compare PERAID with other disk arrays.

We use financial-1 and financial-2 to test the response time. In financial-1 the size of read request is 2707MB and the size of write request is 14901MB. In financial-2 the size of read request is 6778MB and the size of write request is 1860MB. We set the value of chunk as 32k and the value of stripe 96k, which will prove to be the best value of stripe in the latter experiments. The number of stripes for memory is 100, which means the member is 96*100k. The speed of trace is 64 which will also prove to be the best value in the latter. Figure 5(a) and Figure 5(b) show the test data of financial-1 and financial-2.



(a) average response time of financial-1 (b) average response time of financial-2

Figure 5. average response time with different trace.

PERAID has an advantage in writes because it uses write logging in the cache. From Figure 5(a) we can see that the average response time of reads is less than GRAID and RAID0, but it is almost equal to that of RAID10. The average response time of writes is much less than any of the others. In financial-1, the ratio of write request is much larger than read request. This will lead to PERAID having an average response time much less than any of the others. The average response time of PERAID is 67.21% less than that of GRAID, 67.23% less than that of ERAID0 and 67.00% less than that of RAID10.

In Figure 5(b), the ratio of read requests is about double than write requests. We can conclude that the performance of PERAID is not very good in this situation from the figure. Note that we can't see PERAID in average response time of writes because it is too small for us to see. In TABLE V we can see that the write request ratio is 17.65% and under our algorithm SRWLW the write response time is very small. From the test we get, its value is 0.016. The average response time of PERAID is 14.04% less than that of GRAID, 14.10% less than that of ERAID0 but 16.73% more than that of RAID10. The reason that the average response time of PERAID more than that of RAID10 is that our system deals with write requests well but not with read requests. In the latter experiment, we tested the ratio of reads and writes in PERAID, and the average response time is less than other systems.

Because the ratio of write request is bigger in financial-1, the effect is more obviously. The data can be parallel processed in RAID10 because the data can be read from main disks and mirror disks at the same time. And with the read ratio increased in financial-2, the response time of RAID10 is less than RAID0 and the others.

b) Compare in PERAID with different parameters.

In order to verify the performance analysis in the third section, we use the tool Iometer to test the response time in different random ratios. We test it in 100% write ratio and the request size is 4KB. SRWLW algorithm deals with the random request so that the random ratio of request has little effect on PERAID. We can conclude from Figure 6 that the response time of GRAID, ERAID and RAID10 will increase with the random ratio request increasing, but it will have no effect on PERAID. The line of PERAID is almost a horizontal line. When the request is write and random request, the response time of PERAID will be the lowest which will be 78.1% less than GRAID, 74.4% less than ERAID and 79.4% less than RAID10.

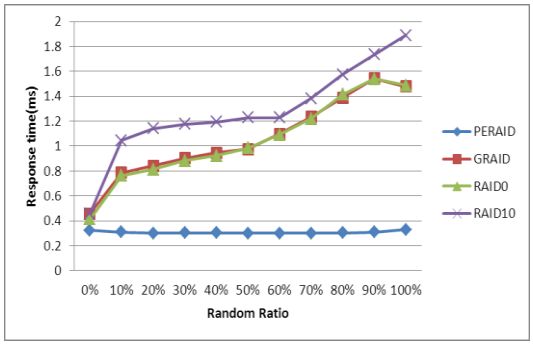


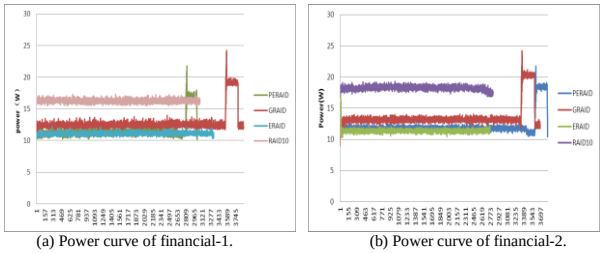
Figure 6. average response time with different random write ratio.

2) Energy consumption comparison

The ZH-102 can measure the value of dc lines in real time. The data acquisition server records the current value every one second. In measuring the total energy consumption of the system, it shows the change of the energy consumption system more clearly in power change than current changes. The power uses the equation P = IU. The parameters that the

power cord line voltage is 12 V, and red line voltage is 5 V. Combined with the data portable wave record instrument measured, it will be easy to calculate the change of the power system.

We first test an offline flush disk. We use the trace financial-1 and financial-2 to test the energy as Figure 7(a) and Figure 7(b) shows. We set the cache to 16G, which means the cache is big enough that it will not flush the data. We can see that ERAID performs best, because it opens 4 disks and has good energy efficiency, but at the expense of reliability. PERAID is 29.4% less than RAID10, and 7.7% less than GRAID. The gap between RAID10 and ERAID is based on two reasons [14]. The first reason is due to synchronous write and low write bandwidth in RAID10, which is explained in the performance comparison. The reason why GRAID is higher than PERAID is because there is a log disk in GRAID. At the end of PERAID and GRAID's line, there is a sudden increase curve. It is because the system flushes the disk and the energy consumption will increase.



(a) Power curve of financial-1. (b) Power curve of financial-2.

Figure 7. Power curve of different trace.

Secondly, we test an online flush disk. We use the trace financial-1 and set the cache as 2G, 4G, 8G and 16G as Figure 8 shows. With the capacity of cache increase, the flush time will delay and the energy will be less. We can calculate that the energy consumption when the cache is 16G is 4.3% less than 8G, 21.1% less than 4G, and 26.9% less than 2G. The reason is when flushing the disk, the energy will increase. So we will increase the capacity of the cache to delay the flush time. The best situation is to flush in its spare time such as at night.

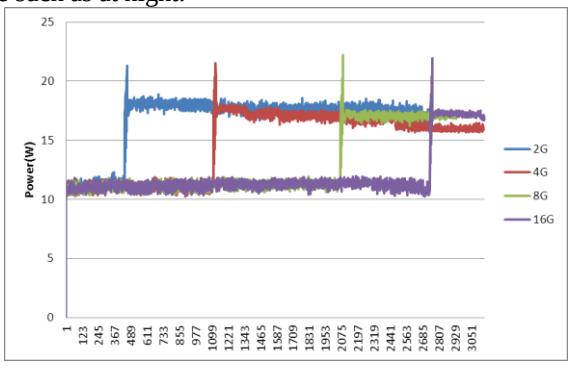


Figure 8. Energy consumption when flushing disk.

IV. RELATED WORK

In this section, we discuss the disk array layouts which are popular. Then we discuss the reliability research, which always use the MTTDL to prove the reliability of the system. At last, we discuss the small write technology which we used in the cache.

A. *Disk Array layouts*

Since RAID has been invented in 1988 by G. A. Gibson [9], the reliability has been a concern by every researcher. MAID[8] puts forward a kind of energy consumption disk array which has an alternative tape library as a backup system. It will organize a group of disks into the form of RAID0, and define a part of disk as hot disk while others as cold disk. Hot disk is active for a long time while cold disk is at low energy consumption status. This technology has solved the energy-efficient problem, but it does no effect on reliability.

The main idea of PDC[10] is to migrate the popular disk data to a subset of the disks array dynamically, so that the load becomes unbalanced and most of the disks can be sent to low-power modes. In realization, it puts the files visited the most on the first disk while putting the second-highest files in the second disk, and so on. PARAID or EERAID is almost the same as PDC. ERAID[13] saves energy by spinning down partial disk groups. Through its time-window control scheme, it can control the tradeoff between energy conserving and performance degradation. All four papers also have the same defect that they have no concern on the reliability but aim at the energy conserving.

AutoRAID[27] mixes RAID1 and RAID5 to achieve a good tradeoff between performance and energy. The architecture of PERAID is of similar design.

GRAID adds a separate log disk on the basis of RAID10 so that it has two replicas in writing new data or reading old data. Its reliability can be guaranteed. On the other hand, hardware added in GRAID leads to increase loading time and uncertainty, which is not advisable.

Rabbit[18] and Sierra[19] are about power-proportional distributed storage. They all put reliability in an important role and complete multi-replications. When a primary node fails, the non-primary one will be activated to restore the data in the primary. Besides this, each replica are grouped into gear groups. When one of the primary servers fails, gear in non-primary groups will spin up to recover the fails.

B. *The Small Write technology*

Parity Logging[15] is the first paper to introduce the logging technique used in disk arrays by D. Stodolsky, which is used to overcome the small write problem of RAID5. In 1995, Log-structured Array[16] was proposed which combines LFS, RAID5 and a non-volatile cache. LSA writes the updated data into new disk locations instead of writing in place to improve the write performance of RAID5. Similar to Parity Logging, Logging RAID[17] is also proposed to solve the small write problem of RAID5. Parity Logging, LSA and Logging RAID are all based on RAID5.

Storage system designers should consider how to balance performance, power consumption and reliability. It motivates us to propose a RAID system with a high proportion of performance and energy consumption without degrading reliability. Our proposal has the main advantage of keeping a

high proportion of performance and energy consumption without requiring any additional hardware.

V. CONCLUSIONS

In this paper, we develop a high-reliability, high-performance and energy-efficient storage system named PERAID. It has multiplex replicas to provide power proportionality for general reads and writes. It saves power by spinning down the replica groups without migrating data and imposing extra requirements. It uses part of a primary disk composing parity software RAID5 as a write buffer to prevent data loss while receiving write requests. This parity RAID merges many random small writes into few large writes to gain high write performance. At the same time, it provides a flush disk mechanism to accelerate the flush rate and choose the right time to flush.

Implementing the system and doing experiments with real system traces, our results show that PERAID is effective in improving performance and saving energy. The result shows that PERAID's performance is better than that of other disk arrays especially when write ratio in request more than 80 percent. At the same time, PERAID can save energy 29.4% compared to RAID10, and 7.7% compared to GRAID.

We will see in Table II the performance and energy in different disk arrays. From the table we can see that the composite score of PERAID is the highest.

TABLE II. PERFORMANCE AND ENERGY IN DIFFERENT DISK ARRAYS

Scheme	Performance	Energy Efficient	Reliability
PERAID	1	2	2
GRAID	3	3	3
RAID10	4	4	1
ERAID	2	1	4

There is still much work to do in the future which mainly contains two directions. First, because of experimental conditions, we have not tested the performance and energy efficient of the replica which will be the next experiment. Second, we also can make the cache to be RAID10, RAID1 and so on. The purpose is to find out which combination is the best for performance, energy efficiency and reliability.

In conclusion, we believe that PERAID is an attractive disk array design: one that offers high performance, while achieving significant energy savings.

ACKNOWLEDGMENT

This Research is supported by the National Basic Research Program (973) of China (No. 2011CB302303), the National Natural Science Foundation of China (No. 60933002), the Natural Science Foundation of Hubei province (NO. 2010CDB01605), the HUST Fund under Grant (Nos.2011QN053 and 2011QN032), the Fundamental Research Funds for the Central Universities, the US National Science Foundation Grant CCF-0811413, CNS-1115665, and National Science Foundation Early Career Award 0953946.

REFERENCES

[1] Hyeonsang Eom and Jeffrey K. Hollingsworth. Speed vs. accuracy in simulation for i/o-intensive applications. IPDPS, IEEE Computer Society Press, 2000. 315~322.

[2] Jun Wang, Xiaoyu Yao, and Huijun Zhu. Exploiting in-memory and on-disk redundancy to conserve energy in storage systems. IEEE Trans. Comput, 2008, 57(6): 733~747.

[3] Y. Hu and Qing Yang . DCD---Disk Caching Disk: A New Approach for Boosting I/O Performance. The 23rd Annual International Symposium on Computer Architecture, Philadelphia PA May, 1996.

[4] Maximum Institution Inc., Power, heat, and sledgehammer, White Paper, 2002.

[5] H. Kim, EJ Kim, RN Mahapatra. Power Management in Raid Server Disk System Using Multiple Idle States. In Proceedings of International Workshop on Unique Chips, 2005.

[6] Gurumurthi S, Sivasubramaniam A, Kandemir M, Franke H, etc. Reducing Disk Power Consumption in Servers with DRPM. Volume: 36,Issue: 12,page(s): 59- 66, 2003.11.

[7] D. Colarelli and D. Grunwald. Massive Arrays of Idle Disks For Storage Archives. In Proceedings of the 15th High Performance Networking and Computing Conference, November 2002.

[8] Ethan Miller and Randy Katz. An analysis of file migration in a unix supercomputing environment. In USENIX Winter Technical Conf. Proceedings, 1993. 421~433.

[9] D. A. Patterson, G. A. Gibson, and R. H. Katz. A case for redundant arrays of inexpensive disks (RAID). In Proceedings of the 1988 ACM SIGMOD international conference, Chicago, Illinois, USA, June 1988.

[10] E. Pinheiro and R. Bianchini. Energy Conservation Techniques for Disk Array-based Servers. In ICS'04, Jun. 2004.

[11] C. Weddle, M. Oldham J. Qian, A. A. Wang, P. Reiher and G. Kuenning. PARAID: The Gear-Shifting Power-Aware RAID. In FAST'07, Feb. 2007.

[12] D. Li and J. Wang. EERAID: Energy-Efficient Redundant and Inexpensive Disk Array. In I I th ACMSIGOPS European Workshop, Sep. 2004.

[13] D. Li, J. Wang. eRAID A Queuing Model Based Energy Saving Policy. In Proceedings of the 14th IEEE International Symposium on Modeling, Analysis, and Simulation. Boston. 2006. New York: ACM, 2006. 77~86.

[14] Bo Mao, Dan Feng, Hong Jiang, et al. GRAID: A Green RAID Storage Architecture with Improved Energy Efficiency and Reliability. In proceedings of the 16th Annual Meeting of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'08). Baltimore, MD. USA. Sep 8-10, 2008. 113~120.

[15] D. Stodolsky, G. Gibson, and M. Holland. Parity Logging Overcoming the Small Write Problem in Redundant Disk Arrays. In ISCA '93, May. 1993.

[16] J. Menon. A Performanee Comparison of RAID-5 and log-Struetured Arrays. In HPDC'95, Aug. 1995.

[17] Y. Chen, W. Hsu, and H. Young. Logging RAID An Approach to Fast, Reliable, and Low-Cost Disk Arrays. In Euro-Par'00, Aug. 2000.

[18] Hrishikesh Amur, James Cipar, Varun Gupta, Michael Kozuch, Gregory Ganger, and Karsten Schwan. Robust and flexible power-proportional storage. In Proc. Symposium on Cloud Computing (SOCC), Indianapolis, IN, June 2010.

[19] Eno Thereska, Austin Donnelly, and Dushyanth Narayanan. Sierra: Practical Power-proportionality for Data Center Storage. EuroSys'11, April, 2011.