Augmenting RAID with an SSD for Energy Relief

Hyo J. Lee
Computer Engineering
Hongik University
Seoul, Korea
hjlee@mail.hongik.ac.kr

Kyu H. Lee
Computer Science
Purdue University
West Lafayette, IN, 47907
kyuhlee@purdue.edu

Sam H. Noh
School of Comp. & Info. Eng.
Hongik University
Seoul, Korea
samhnoh@hongik.ac.kr

Abstract

In this paper, we describe a design of a novel architecture for RAID that uses an SSD as a large cache to conserve energy. This approach stems from the fact that short term footprints are small enough to be efficiently managed within an SSD. More specifically, in this study, we consider two simple approaches to reduce the energy consumed in RAID. First, when a read happens in RAID, a copy of the read request is copied to the SSD, so that future requests may be serviced by the SSD. Then, for writes, all writes are buffered in SSDs so that the interval between requests may be increased reducing activities at RAID disks. We incorporate these approaches into a real implementation of a RAID 5 system that consists of four hard disks and an SSD in a Linux environment. Our preliminary results in actual performance measurements using the cello99 and SPC traces show that energy consumption is reduced by a maximum of 14%.

1 Introduction

The federal Environmental Protection Agency (EPA) reported electricity consumption of U.S. data centers in 2006 was about $4.5 billion [2]. This is more than the electricity consumed by all color televisions in the U.S. These huge electricity bills are painful to the companies that run these data centers. For this reason and, more importantly, the environmental aspect, studies on energy conservation has become an active research issue. In this paper, we look into how energy can be conserved at the storage system component of a system, which is responsible for a large portion of energy consumption in today's data centers [17].

RAID is a popular form of storage system in data centers since they can provide high performance and fault-tolerance with relatively low cost. From the energy conservation perspective, however, it is more challenging due to the philosophy behind RAID. Performance of RAID comes from parallelism. RAID tries to even the load of every disk, and this forces every disk to be active even though the load may be light.

Recently, the Solid State Drive (SSD), a new type of disk device that makes use of Flash memory, is being spotlighted as a new storage medium due to its many attractive characteristics. It is fast, lightweight, durable, and noiseless, but most of all it is efficient in terms of energy. Despite these attractive characteristics, there are still a number of challenges that must be overcome for it to be adopted as a main storage component in data centers. Above all, the capacity of SSD is too small to construct data centers and the cost per capacity also can be too high for server systems.

In this paper, we describe the design of a novel architecture for RAID that uses an SSD as a large cache to conserve energy while meeting their performance goal. (Though any form of RAID is possible, we limit our study to RAID 5; hence hereafter, RAID will imply RAID 5.) If judiciously managed, this will allow the hard disks composing RAID to be inactive most of the time leading to energy savings. This approach stems from the fact that even though overall storage size is growing at a high rate, considerable space is being unused and even for used space, short term footprints are small enough to be efficiently managed within an SSD.

More specifically, in this study, we consider two simple approaches to reduce the energy consumed by RAID. For reads, when a read happens in RAID, a copy of the read request is copied to the SSD, so that future requests may be serviced by the SSD. Then, for writes, all writes are buffered in SSDs so that the interval between requests may be increased so that activities at RAID disks may be reduced.

We have implemented these approaches in a Linux environment with real hard disks and an SSD deployed. Our preliminary results in actual performance measurements using the cello99 and SPC traces show that energy consumption is reduced by a maximum of 14%, which is not significant, but encouraging. With many more optimizations possible, more convincing results are anticipated.

The rest of the paper is organized as follows. Section 2 discusses SSD basics and works related to this study. We then describe in more detail the approach we take in Section 3. Section 4 describes the implementation and the results. Finally, we summarize and give directions for future work in Section 5.

2 Background and Related Work

In this section, we first present an SSD overview and compare it with conventional disk drives. We then discuss several of the previously proposed energy management techniques for disk arrays.

2.1 SSD overview

Flash memory is widely used in many small devices and embedded systems because of its characteristics such as low energy consumption, fast data access, and light weight. Recently, many companies are producing and marketing Solid State Drives (SSDs) that are based on NAND Flash memory to adopt the many attractive Flash memory features into personal and server computers. NAND Flash memory-based SSDs are constructed from an array of Flash memory modules that are accessed in parallel. By employing parallelism and interleaving among the modules that comprise the SSD, performance of SSDs is considerably better than conventional hard disks for random reads, while sustaining comparable performance for other kinds of workloads. Performance of SSDs has been consistently increasing recently and is expected to continue increasing for some time in the future.

The greatest advantage of SSDs, though, is energy efficiency. Since SSDs do not have any mechanical moving parts energy consumption of SSDs is much smaller than disks. However at this time, SSDs have some limitations as server system storage. First, the capacity of SSD is not large enough for server computers. Currently, the 128GB SSD is the largest one available in the market, though surely in the close future, this is bound to increase. The more serious problem is that of cost per capacity; currently the cost per capacity of SSDs is no match to that of hard disks. Hence, though there have been approaches to construct RAID using only SSDs, this can be an attractive solution to the few tech savvy personal users but not for server systems in general.

2.2 Related works

There have been considerable work related to saving energy in the storage system. One approach that has been proposed is to use multispeed disks that have two or more rotational speed levels in active mode. Gurumurthi et al. [6] and Carrera et al. [5] propose a multi-speed disk that change their rotational speed dynamically, and they make use of this feature for energy consumption.

Zhu et al. [16] propose an energy management scheme that make use of two-speed disks for disk arrays. Unfortunately, to date, this type of disk is not yet readily available. Our approach is different from these approaches in that we are making use of available technology, that is, SSDs, making our approach more practical.

Another common approach to save energy in conventional disk arrays is to keep disks in standby mode as long as possible. To achieve this goal, many different data management schemes have been proposed. Some previous studies have proposed migrating frequently accessed data to a particular disk(s) so that the other disks can remain in standby mode longer [15, 16]. Some studies consider producing redundant data by making copies of the original data to some free disk area to prevent unnecessary spin up [8, 11]. If the requested data is in a disk that is in standby mode and the replicated copy is in one of the active disks, the request may be serviced without spin up overhead.

Yet another common approach used for energy savings and improved performance is to make use of memory management algorithms for the cache on the controller. Here, redundant data is stored in memory located in the hardware disk array controller. Some studies have focused on prefetching schemes [3, 13], while some studies have proposed caching schemes [4, 9, 17] for energy efficient disk arrays. However, because of the memory space limitation within the controller, the replacement policy becomes the critical issue for both caching and prefetching. Furthermore, it is very difficult, if not impossible, to expand memory in disk array controllers. Hence, use of this approach may be limited. In some sense, the approach taken in our study is similar to these approaches. First, we make use of redundant data that is kept in the SSD. Second, the SSD is used as a cache. Though conceptually the same, redundant data is kept in a much faster and energy efficient medium that does not require any spin up time. Furthermore, the SSD as a cache is much larger than could be imagined with the disk controller.

3 SSD cache in RAID

In this section we describe our idea in detail. First, we describe our observations that serve as our motivation. Then, we describe the design of our system.

3.1 Observation

Figure 1 shows two aspects of the HP cello99 trace, which is a popular I/O trace used in many I/O related studies. The dates of these traces are from November 29, Monday to December 4th, the Saturday of that week. The aspect that we need to note is the footprint of the trace relative to the whole data set size of the disk, denoted as the ’Footprint Ratio’ in the figure. The numbers tell us that the actual footprint for each day is generally less that 0.2% of the whole data set. That is, given a day of activities, the range of blocks that are accessed is very small relative to the data that the disk possesses. The 'Footprint Size' on the right y-axis reveals the actual size of the footprint. It shows that over a day's activities, only 30GBs or less are being accessed. What is more, the dark column, which represents the newly inserted data each day in comparison to the data accessed the prior day is even smaller, always being well below 10GBs.

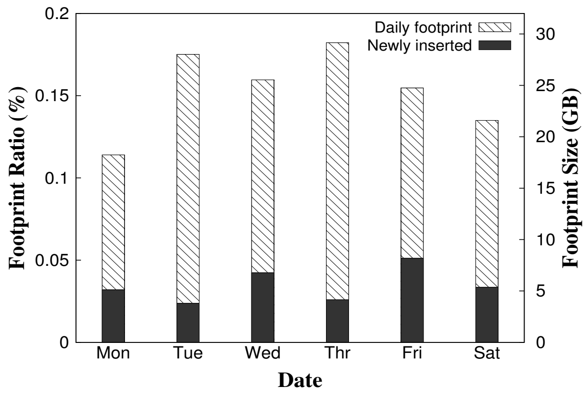


Figure 1: HP cello99 daily footprint ratio

Our motivation for this work is based on these observations. That is, they tell us that if we could take this footprint, which is relatively quite small, into a more energy efficient medium and service requests from there, benefits may be possible. More specifically, if the footprint could be cached in a low energy medium allowing the high energy consuming RAID disks to reduce their activities, then energy conservation would be possible. Finding a cache of this size and satisfying the energy requirement is the question at hand.

There can be several candidates that may be able to satisfy this large cache requirement. The traditional hard disk is one such candidate. However, a hard disk is a high energy consumer, and if one disk is used to cache a whole day's activity, performance will obviously be degraded. Furthermore, the benefit of having a RAID, that is, high performance and reliability, simply disappears. Memory in the RAID controller could be another candidate. This, however, is prohibitively expensive. Furthermore, RAM is volatile, hence the reliability of this method comes into question. The third candidate could be non-volatile RAM (NV-RAM), which resolves the volatility issue of RAM in RAID controllers. Since NV-RAM provides the memory interface, it would be easy to deploy them as a cache. However, NV-RAM is still very expensive and has limited capacity making it even more infeasible than RAM.

The medium that is energy efficient, non-volatile, large enough, and at the same time, able to provide high performance under reasonable cost today is the Solid State Drive (SSD). Due to these many merits, we employ the SSD as the large cache. Not only does SSDs consume low energy, while providing high performance, they are disks with the exact same disk interface as hard disks, hence incorporating them to RAIDs is simple. Only the controller needs to be aware of its existence.

3.2 System design

The system that we propose simply adds a single SSD to a normal RAID configuration. The key design issue is, then, how to cache the whole daily footprint so that all other disks may remain inactive. Occassionally, the RAID disks inevitably must awake. Ideally, this is when all dirty blocks residing in SSDs should be written and when all reads that are anticipated to happen in the future should be prefetched.

In the work that we present here, we are not able to provide all these features. Instead, we only consider a very simple approach. Specifically, when the very first read request of a block arrives at the controller, this read is satisfied at the RAID disk. The block, however, is copied to the SSD as well for possible later requests. For writes, all write requests are satified at the SSD to extend the RAID disk inactivity time. However, to prevent lose of data during SSD failure, we make a mirror copy in unused space of an active disk.

4 Evaluation

In this section, we discuss the evaluation of our system. We describe the implementation, the environment in which the evaluation was performed, and the results.

4.1 Evaluation environment

For our evaluation, we implement our system on the Linux software RAID environment. Our environment uses a SATA interface as we were not able to purchase SCSI SSDs. Though SATA based RAID would be slower than SCSI based RAID the idea presented here is equally applicable. As hard disks composing RAID, we use 500GB hard disks (ST3500320AS) produced by Seagate, and for the SSD cache, we use a 32GB SSD (MSD-SATA3035) produced by Mtron. According to their data sheets, this SSD consumes 0.5W when idle and 1.66-2.43W when reading/writing while the harddisk consumes 7.96-9.29W when idle and 11.16W when reading/writing [10, 12]. All our implementations and evaluations are done on the Linux 2.6.21.7. The original RAID system consists of four hard disks and our proposed system adds one SSD to that original configuration.

The Linux software RAID is implemented in the `md` (multiple device) driver and controlled by the `mdadm` application. We modified the `mdadm` to consider the

SSD cache. Furthermore, the following additions are made to the `md` device driver code.

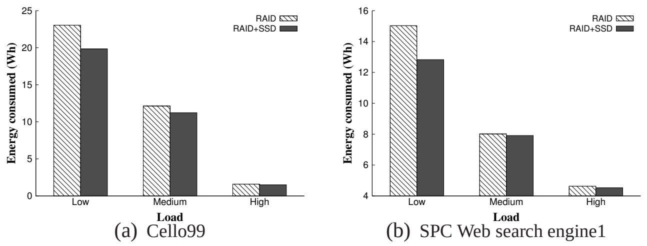
We maintain a hash table to manage data in SSD. Along with the original disk number and sector number, we keep information such as it corresponding SSD sector number and its dirty/clean status in this hash table. When a read request arrives, the SSD is first checked through the hash table. If it is not found here, then the request is serviced via the RAID hard disk. While so doing, that sector is copied to the SSD and the hash table is updated. If the read request is found in the hash table, the request is serviced via SSD directly. If the request is a write, the request is written on the SSD and on a RAID hard disk that is awake, and again, the hash table is updated.

The performance measure of interest is energy and response time. To measure how much energy is consumed, we take two approaches simultaneously. One is to implement a daemon that monitors the disk status. There are four statuses that our disks can be in, namely, sleep, standby, idle, and active. This daemon distinguishes between sleep, standby and idle/active (that is, it cannot distinguish active and idle states). At every 5 second intervals it checks to see what state the disk is in and records this into a separate disk aside from the RAID hard disks and the SSD. In the other approach, we make actual measurements of the energy being consumed using the HPM-300A, a power meter, which can measure power consumed by the entire system [1]. For the response time, we measure the time it takes for each request using the `gettimeofday` call before and after the requests are sent, driven by the workload.

For the workload we replay two traces. One is the cello99 traces that are widely used in I/O related research [7], and the other is a trace of the SPC Web search engine benchmark [14]. For the cello99 trace, three days worth of traces (dated Dec. 2 (Thursday), 3 (Friday), and 4 (Saturday)) are used. The former is write oriented, while the latter is a read oriented workload. For each of the workloads, we run the experiments at three different request rates representing the load of the system. To represent low, medium, and high loads, we generate 100, 200, and 500 requests per second, respectively. The requests are sent directly from the application to the `md` device.

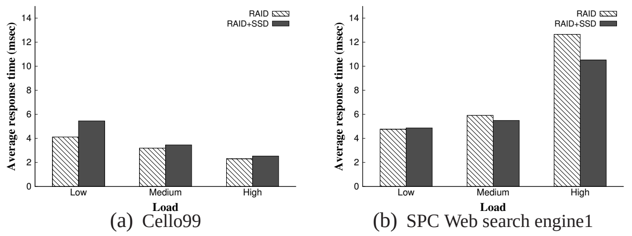
4.2 Results

The results of the experiments are presented in Figures 2 and 3. The former reports the energy consumed by the two different RAID systems. The x-axis is the load of the system, that is, the rate at which the traces are run and the y-axis reports the Watt-hours consumed during the run. The numbers reported here are the energy consumed by the disks. These were obtained by measuring the energy consumed by the system with the disks off, for the duration of the time that the trace would execute with the original RAID system, and then, subtracting this value from the actual energy measurements for executing the traces with the original RAID and the SSD attached systems. The disk off state was acheived by putting all the disks under the RAID configuration to the sleep state using the `hdparm` command. Note that more energy is being consumed for lower loads as it is taking longer. The energy savings at low loads is greater as well, saving approximately 14% for both the cello99 and SPC traces. For medium and high loads, the saving are minimal being less than 10%.



(a) Cello99 (b) SPC Web search engine1

Figure 2: Energy consumption



(a) Cello99 (b) SPC Web search engine1

Figure 3: Average response time

The savings reported here are only modest. However, they are encouraging in two aspects. First, through the disk monitoring daemon, we were able to observe that for most of the executions for both traces, the hard disks in the RAID system rarely sleep. This is because of the way the experiments are set up. We were not able to use the actual timing information available in the traces because our system is an actual implementation. Hence, we used the load approximations to inject requests to the system. This left no room for the disk to become idle, hence sleep time was reduced. We are currently investigating how to accurately model the request intervals of the traces so that a more realistic workload could be reproduced.

So one must wonder where all this savings is coming from. This is the second encouraging factor. In our current implementation, we are taking a simple and naive approach for making use of the SSD cache as described previously. Yet, the little activity that is being transferred to the SSD is resulting in around 10% savings in energy. Even though disks are not going to sleep, their activities are being reduced as some of these are being done by the SSD. This is resulting in the energy savings. With further optimizations such as intelligent prefetching schemes, further improvements should be possible.

Figure 3 shows the average response times observed by all the requests. For the SPC workload, which is largely composed of random reads, we see a roughly 17% improvement when the load is high and a small improvement for medium loads. For the low load, performance becomes worse by a small percent. For the cello99 trace, we see that response time increases by a maximum of 30% for the low load, though the other loads fair better. An interesting observation for this trace is that the response time decreases for higher loads. Though we are taking a closer look, we do not have a clear reason for this behavior though we can conjecture that the way writes are being serviced has something to do with it as the cello99 traces are write oriented and writes are being requested asynchronously.

5 Conclusion

In this paper, we proposed a design of a novel architecture for RAID that uses an SSD as a large cache to conserve energy while meeting their performance goal. This design was based on our observation that short term daily footprints are small enough and change slow enough to be efficiently managed within an SSD of today.

We incorporated our approaches into a real implementation of a RAID system using a Linux environment consisting of four hard disks and an SSD. Our preliminary results in actual performance measurements using the cello99 and SPC traces show that energy consumption is reduced by a maximum of 14%.

The performance numbers, though encouraging, were not satisfactory. We believe there is much room for improvement. In our current implementation, we took a simple and naive approach to making use of the SSD cache. In order to make better use of this cache, we need to incorporate a prefetching mechanism so that the RAID disks may remain idle for long durations. We also need to consider piggybacking reads and writes so that much of the disk activities may be done when and only when they are necessary. These, and more, are part of the research that are currently being conducted.

6 Acknowledgments

This work was supported by the Korea Science and Engineering Foundation(KOSEF) grant funded by the Korea government(MOST) (No. R0A-2007-000-20071-0)

References

[1] *HPM-300A*. http://www.adpower21.com.

[2] Report to Congress on Server and Data Center Energy Efficiency Public Law 109-431. http://www.energystar.gov, 2007.

[3] S. H. Baek and K. H. Park. Prefetching with Adaptive Cache Culling for Striped Disk Arrays. In *Proceedings of the Annual USENIX Technical Conference*, 2008.

[4] L. N. Bairavasundaram, M. Sivathanu, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau. X-RAY: A Non-Invasive Exclusive Caching Mechanism for RAIDs. In *Proceedings of the 31st Annual International Symposium on Computer Architecture (ISCA '04)*, 2004.

[5] E. V. Carrera, E. Pinheiro, and R. Bianchini. Conserving Disk Energy in Network Servers. In *Proceedings of the 17th International Conference on Supercomputing*, 2003.

[6] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Franke. DRPM: dynamic speed control for power management in server class disks. In *Proceedings of the 30th Annual International Symposium on Computer Architecture (ISCA '03)*, 2003.

[7] HP Labs. *Tools and Traces*. http://www.hpl.hp.com/research/ssp/software/.

[8] H. Huang, W. Hung, and K. G. Shin. FS2: dynamic data replication in free disk space for improving disk performance and energy consumption. In *Proceedings of the Twentieth ACM Symposium on Operating Systems Principles (SOSP '05)*, 2005.

[9] D. Li and J. Wang. EERAID: energy efficient redundant and inexpensive disk array. In *Proceedings of the 11th ACM SIGOPS European Workshop (EW11)*, 2004.

[10] Mtron. *MSD-SATA3035*. http://mtron.net/.

[11] E. Pinheiro, R. Bianchini, and C. Dubnicki. Exploiting redundancy to conserve energy in storage systems. In *Proceedings of the International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '06)*, 2006.

[12] Seagate. *ST3500320AS*. http://www.seagate.com/.

[13] S. W. Son and M. Kandemir. Energy-aware data prefetching for multi-speed disks. In *Proceedings of the 3rd ACM Conference on Computing Frontiers (CF '06)*, 2006.

[14] Storage Performance Council. *SPC-1 Specification*. http://www.storageperformance.org/specs.

[15] C. Weddle, M. Oldham, J. Qian, A.-I. A. Wang, P. Reiher, and G. Kuenning. PARAID: a gear-shifting power-aware RAID. In *Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST '07)*, 2007.

[16] Q. Zhu, Z. Chen, L. Tan, Y. Zhou, K. Keeton, and J. Wilkes. Hibernator: helping disk arrays sleep through the winter. In *Proceedings of the Twentieth ACM Symposium on Operating Systems Principles (SOSP '05)*, 2005.

[17] Q. Zhu, F. M. David, C. F. Devaraj, Z. Li, Y. Zhou, and P. Cao. Reducing Energy Consumption of Disk Storage Using Power-Aware Cache Management. In *Proceedings of the 10th International Symposium on High Performance Computer Architecture (HPCA '04)*, 2004.