Green Multi-Disk Device Mapper Target

~~Students~~: Ming Chen and Rajesh Aavuty
~~Sponsors~~: Zhichao Li and Erez Zadok
Stony Brook University

Draft of 2012/04/09 00:29

Abstract

Storage subsystems in servers are slower than other subsystems despite the fact that they ~~contribute to a large portion of the total power consumption~~. Hybrid disks, which integrate different characteristics of various kinds of disks (e.g., SATA, SAS, SSD) in speed, capacity, price, and power consumption, is a promising solution to the problem. While researches on hybrid disk have reported significant performance boost over traditional magnetic disks, their energy usage is less studied. In this project, we consider energy efficiency of storage systems as important as performance and capacity. Our goal is to provide a multi-disk system that consists of different disks, provides a capacity equal to the sum of all disks, operates at a speed near the fastest disk, but consume less energy than the sum of all disks when they operate individually.

To achieve energy efficiency as well as good performance, our multi-disk system tries to map more frequently accessed blocks (i.e., hot data) to energy-efficient, fast, but small disks such as SSD. Conversely, cold data goes to inefficient, slow, but large disks such as SATA. This exploits spatial locality of data and has similar effect in hybrid model where SSD is used as cache disk. Meanwhile, temporal locality of data is also be exploited to group blocks that are likely to be accessed simultaneously. Instead of scattering blocks among several disks, we try to map them to a single disk. The combination of spatial and temporal locality allows I/O be served using fewer disks and other disks can spin down to save energy. To identify ~~hot data~~ and block groups, workload-specific traces are analyzed offline. Heuristic algorithm is used for adaption to hot data and block groups as they evolve over time. We are also exploring naive direct I/O map without any knowledge of the workloads as baseline approach.

1 Introduction

Increasing IT demand has seen 6X growth of server and 69X growth of storage during the last decade [17]. Data centers energy use is doubling every 5 years, whereas the global electricity prices are increasing 10-25% per year. This makes energy efficiency of computer systems a big concern recently in academic and industrial communities. Studies show that storage systems are responsible not only for the system performance bottleneck but also for up to 40% of the power consumption [14]. This situation is largely caused by the inherent drawback of traditional hard disks (HDD), which consist of physical moving parts. They consume a lot of power to keep disk platters spin at a high speed and they are slow as disk heads have to move to the right position before serving random I/O accesses.

Thanks to the development of solid state drive (SSD) technology, many of the problems related to performance and energy efficiency are alleviated. However, despite ~~of~~ having the above advantages, SSDs are not widely adopted in the server storage stack due to their high capital cost per byte. To overcome this limit, hybrid disks are proposed. Hybrid disks achieve better trade-off between performance and capacity by organizing different disks in the manner conformant to the storage hierarchy.

The most common type of hybrid disks [1] tries to reap the benefits of SSDs by having a small flash memory on each hard disk and using this flash memory as non-volatile cache for the hard disk. A host can use this non-volatile cache to achieve faster random reads since it is fast and takes constant time to access any block in the SSD cache. Although this type of hybrid disks improves performance over traditional disks, the improvement was modest due to the relatively small size of non-volatile cache on each disk. It is also difficult to balance cache load among disks because the SSD caches are separated. For example, if a workload accesses one disk ~~much~~ more frequently than others, then this would result in frequent cache misses on that particular disk, whereas the caches on other disks ~~are~~ under-utilized.

To enable servers to benefit more from the power efficiency and high performance of solid state media, we used a single but relatively large SSD as cache (it has similar effect as cache but is not exactly cache) for all other disks (referred to as secondary disks). This approach ~~would~~ achieves a good cache-hit ratio because of the large size of the cache. Thanks to data locality, it would also make the SSD capable of serving majority of the disk accesses as we can see in [3, 4, 20]. Therefore, other disks are idle for longer periods of time and thus

can be powered down to save energy.

The difference between the above two types of hybrid disks seems to be not very clear if we consider the virtualization of disk such as Linux Logical Volume Management (LVM). By grouping all hard disks to form a single virtual disk, the cache on all these hard disks can be gathered together which becomes equivalent to the large SSD we are using. This workaround alleviates the load balance problem. However, it is less helpful in saving energy because the virtualization of multi-disk is not aware of energy efficiency. If there are non-adjacent blocks which are often simultaneously accessed, we can perform data grouping and store them onto one physical disk so that only this disk has to spin up when they are accessed. In this case the optimization cannot be achieved if the LVM virtualization approach is employed.

To exploit spatial and temporal data locality to the largest extent, we collected workload-specific traces and are trying to find hot data and block groups of working sets. For hot data, we simply find blocks which are most frequently accessed. We realize that hot data changed over time. However, this offline study is still helpful because some data is inherent hot, e.g., filesystem metadata and latest data. Moreover, because our analysis is workload specific, it is likely the same I/O pattern will repeats in same or similar workload. We also try to keep hot data in cache online using the LRU heuristic algorithm. To identify block groups, we will use methods in [23]. Blocks fall into same group but spread multiple disk will be mapped to same physical disk so that less disks are required to power up. Additionally, data grouping has the benefit of isolating faults [16,23].

We use the Linux Device Mapper framework and implement our hybrid model as a Device Mapper target. This provides modularity and transparency as it enables us to create virtual disks without exposing the details of underlying physical disks. By leveraging the strengths of different disks, our virtual disk can better trade-off among power consumption, I/O performance, and storage capacity.

The rest of the paper is organized as follows. Section 2 describes our design and implementation. Section 3 presents some block trace analysis. The energy efficiency and I/O performance of our system is evaluated in Section 4. Related work is discussed in Section 5. Conclusions are drawn in Section 6.

2 Design

We present the design of our green multi-disk driver as a Linux Device Mapper target. Linux Device Mapper, as depicted in Figure 1, is a generic framework to map one block device onto another. It works by processing data passed in from a virtual block device, that it itself provides, and then passing the resultant data on to another block device [22]. The advantage of Device Mapper is that it can provide virtual devices to user programs and kernel modules above block level without exposing details of underlying devices. Thanks to this transparency, the underlying devices can be of different size and type (can in turn be virtual or physical) and the mapping can be performed in different manners. As essentially a middle-ware sitting between the Generic Block Layer and block devices. Device Mapper receives BIOs, which is a kernel structure describe I/O requests, from Generic Block Layer, then redirect them to other block devices by modifying the target device and sector offset.

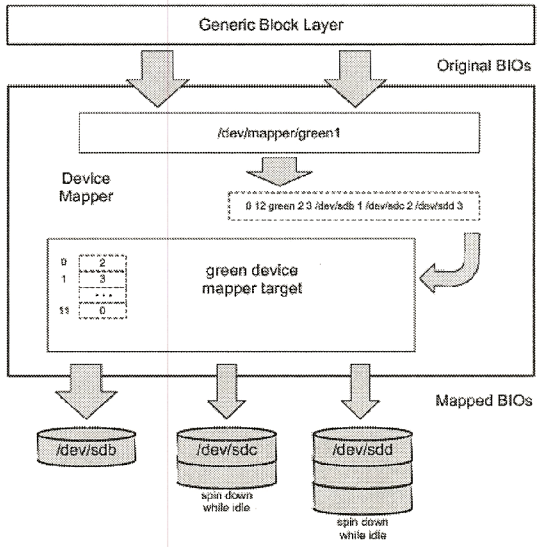

Figure 1: Linux Device Mapper and Green Device Mapper Target. '/dev/mapper/green1' is a created virtual block device. The two dashed boxes are mapping tables. The first one is used by Device Mapper to delegate mapping to target; the second one is used internally by the green target.

Device Mapper itself does not perform the redirection. It delegates this operation to Device Mapper targets, which are registered kernel modules performing certain kinds of mapping. This delegation is specified in a mapping table, which contains the type of responsible Device Mapper target for certain regions of block devices. In Figure 1, all I/O request to sectors [0, 0+12) of virtual device "/dev/mapping/green1" will be handled by a target named green; all parameters after "green" are passed to the green target. A mapping target contains functions to construct/destruct mapped devices, map I/O requests, merge adjacent I/O requests, report and control device status.

2.1 Design Goals

The design goal of our green multi-disk Device Mapper target is driven by the following guiding principles:

- **Save energy by allowing disks to spin/power down**. Be energy efficiency is one of the most im-

portant features our green multi-disk target is pursuing. The SSD cache benefits both I/O performance and energy efficiency simultaneously. In this study, we are more concerned of the latter. We have made trade-offs between energy efficiency and other desirable features such as capacity.

- **Use SSD aware data structures and algorithms**. While SSD provides better I/O performance and energy efficiency, it has its own constraints including limited erase-write cycles and inefficient random writes. This principle guides our design to avoid these constraints and extract maximum benefit out of SSD.
- **Provide stable and robust storage**. Because our green target provides customized block mapping and data grouping, it is critical to always perform correct translation from virtual to physical blocks. It should not lose mapping information in case of power failure or other hazardous incidents.

2.2 Disk Management

Device Mapper targets maintain mapping of sectors between virtual and physical disks. The straightforward method is to maintain a sector-wise mapping table. However, it is prohibitive because its size is too large to fit in memory. For example, the size of sector-wise mapping table of a 1TB disk (512-byte sectors) is as large as 8GB with 4-bytes table entries. Store the mapping table on disk (even on SSD) is not an option because it is too slow to incur extra I/O. A solution is to divide disks into larger units. Here, we adopt the LVM term extent, which is the unit of disk managed by LVM, typically 4MB. Then the mapping becomes extent-wise and its size diminishes to 1MB in the above example.

In our green target, multiple physical disks are mapped as a single virtual disk. The physical disks are linearly organized in the order of energy efficiency, i.e., the most energy-efficient one goes first and so on. In Figure 1, '/dev/sdb' is the most energy-efficient one but with smallest capacity, i.e., SSD; '/dev/sdc' and '/dev/sdd' follow decreasingly in energy efficiency and increasingly in capacity. This makes the addressing of physical sectors easy. An extent index and an offset within extent would suffice. We have noticed the case that one I/O request on logically sequential blocks might be mapped to multiple I/O requests on physically non-sequential blocks. Namely, the translation is performed per extent instead of per request. However, extent is of big size, so it is unlikely that the extent by extent translation becomes a performance bottleneck. Furthermore, the Device Mapper framework provides interface to merge adjacent I/O requests, which also alleviate this problem.

The exact size of extent is an important factor as it affects not only memory consumption but also granularity of mapping and data migration. A large size of extent has the following impacts:

- **Smaller mapping table**. As already discussed, the adoption of extent makes the mapping table becomes extent-wise and small.
- **More aggressive pre-fetch**. As energy-efficient disk such as SSD have similar effect as disk cache. When a large extent of data is moved onto SSD, it can be consider as an aggressive pre-fetch.
- **Coarse-grain data migration among disks and more sequential I/O**. Because the major latency of magnetic disk is seek time, a larger sequential migration will not significantly slow down the I/O. Moreover, with large size of migration unit, there are fewer I/O because adjacent sectors are processed in batch. This is beneficial to the life time of SSD as well considering its limited erase-write cycles.
- **High overhead**. Since each extent can represent several sectors, more sectors I/O can be wasted in case of wrong prediction thus adds overhead to the overall system.

Physical extents are managed using a bitmap. The extents on SSD are taken specially. Besides recorded in the bitmap, they are also linked in two lists, one for free extents and the other for used. This facilitates and speeds up manipulation of the extents on SSD, which happens very frequently.

Different workloads might favor different extent sizes depending on file sizes, I/O frequency and read-write ratio. Therefore, we make extent size a configurable parameter to the green target so that different trade-offs can be made via configuration.

2.3 Mapping Table

There are two mapping tables in Figure 1. One is actually a configuration file used by the Device Mapper framework; the other contains extent-wise mapping information used internally by our green target. We are talking about the second one in following discussion.

The straightforward structure of the mapping table is an array of the size of extent number on all physical disks. The mapping table is maintained in memory and can be cached, so its lookup is fast. Besides mapping information, the table also contains other fields including flags, timestamp of the latest access and number of total access (possibly distinguished read from write). Flags are used to record states of extents, which include, for example, 1) whether the extent is accessed or not recently; 2) whether the extent is under migration or not; and 3) whether the extent is updated or not when it is under migration. Timestamp of the latest access and number of total access are used to predict hot extent.

The mapping table need be saved onto disk on power

off as metadata. To be fail-safe, it has a replicate in every physical disk. The in-memory version and on-disk version of mapping table can be slightly different, since it is not necessary to save online information such as timestamp of latest access. To be robust in case of hazardous situation such as power failure, the mapping table is flushed onto disk periodically. To prevent this periodical background job from disturbing other disks' idle periods, this flush saves the table only on the cache disk. Table on other disks are only updated on request or when the system is being shutting down.

2.4 Extent Migration

There are two kinds of data migration. The first is moving an extent into the SSD disk, called promotion. The second is moving an extent out of it, called demotion (similar to eviction in cache management). Both of them are achieved using kcopyd, the infrastructure of the Device Mapper framework for copying data between disks.

Promotion occurs when an extent on secondary disks becomes hot and there is free extents on SSD. To predict hot extent, we adopt a greedy approach and take the extent just been accessed as hot. However, a trade-off between I/O performance and SSD life time is possible when performing promotion of an extent being written. Because SSD has limited erase-write cycles, it makes sense to directly map writes to secondary disk instead of promoting the extent at once. If that extent is read immediately, then we promote it. This will be helpful in case of operations like file copying and file appending, wherein disk is written just once. This does not have significant influence on energy efficiency because the secondary disk hosting that extent has to spin up no matter we promote it right now or not. Once it is up, there is a relative long timeout before it can spin down again because of the large penalty (latency and short disk life time) of disk spin-down, so no additional spin-up is needed if that extent is read soon. Actually, there is research trying to extend SSD lifetime by using HDD as write cache [18]. However, this is a configurable feature that allows user to make informed decision when workload knowledge is available.

Demotion occurs when the number of free prime extents falls below a minimum threshold. Demotion tries to evict cold extents on SSD until the number of free extents goes up to a maximum threshold. There is a daemon running in background to keep the number of free extents between the minimum and maximum thresholds. The minimum threshold is used to ensure that free extents can be readily obtained when promotions are needed. The maximum threshold is used to demote extents in batch so that secondary disks are disturbed less. However, these two thresholds are also parameters that can be changed. When both of them are 1, the demotion becomes just like cache eviction. To find cold extent on SSD, a LRU algorithm is used.

2.5 Data Grouping

We have studied the data grouping in [23]. However, right now, very little thing about data grouping has been done in our implementation. A simple heuristic we are going to experiment is to group data by frequency of access. That is, most accessed data are all mapped on to the first disk, namely, SSD and so forth. The reason behind this is that it is likely for data in same working set to exhibit access frequency at same level.

3 Trace Study

Traces are used to collect information about the system workloads. Traces can be collected at different layers in storage stack. Block traces would contain information about disk offset, number of blocks accessed, Read/Write information and timestamp for every accessed I/O. This information can be used to characterize the workloads and identify I/O access patterns. Trace analysis plays a crucial role in identifying hot data blocks across different disks.The idea is to move hot blocks to primary disk and serve most of the requests from most energy efficient disk. Hence other disks can be spun/powered down for long periods of time to achieve good power savings and increased performance. A simple approach like counting the number of times a particular block is accessed can be used to find the hotness of disk blocks. Apart from this, I/O block traces can be used for variety of other purposes. For example, block trace analysis can be used to tweak the design parameters like extent size and migration threshold.

Workloads which exhibit good data locality properties are most suitable for this project. Workloads such as video server and file server are likely to exhibit this property. The distribution of popular files on disks may change dynamically but at any particular instant, only a fraction of files the disk are popular in the case of above workloads. A workload which is completely random I/O access patterns would not benefit this approach as it would result in frequent data migration between primary disks and secondary disks.

Currently, we use I/O traces for analysis and tweaking parameters. I/O traces can also be used in offline mode to predict the hot blocks. This approach is another option and we may implement this if time permits. Also, to verify the correctness and working of our concept, we replay the traces with and without green Multi-disk target and compare the power consumption and I/O performance.

I/O block traces have been collected for variety of workloads like video server, file server and random workloads. The workloads were run for a relatively long periods (for a few hours). Below, we present the results

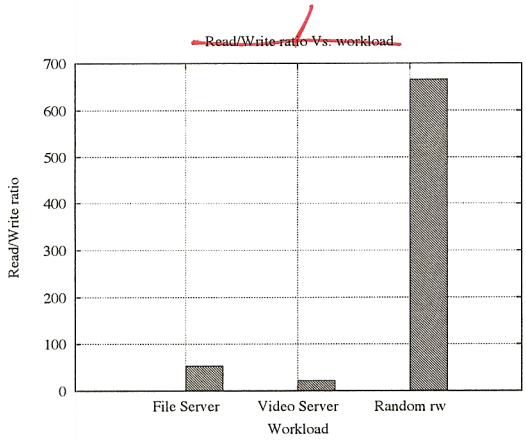

Figure 2: Read/write ratio of video server, file server and random workloads

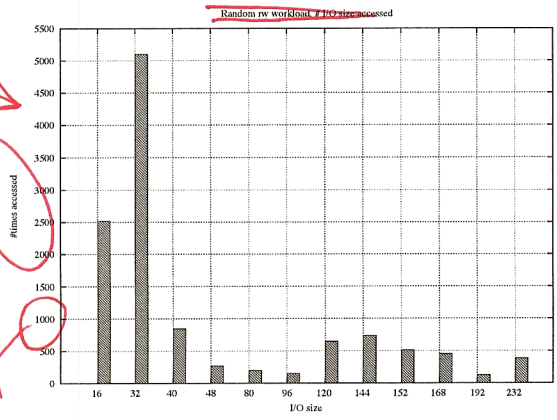

Figure 3: I/O size of the random workload

for some preliminary analysis on the traces. Figure 2 shows Read/Write ratios for the above three workloads. We have collected the number of times a particular I/O size was issued for the above three workloads. Figure 3, which shows the I/O size of the random workloads, is one of them. All these workloads were generated using Filebench and each workload runs for 2 hours. We used three different physical disks with different power consumption and performance characteristics to run these workloads.

4 Evaluation

We have finished the code. But it is still being tested. We are going to perform experiments to evaluate our system in the near future.

To evaluate our work, experiments will be performed on multi-disks systems with and without our green target. Performance results of power consumption and IO speed will be compared. To prove that the green target can save energy, system using green target should report significant less power consumption and comparative IO speed than system with same experiment setup but not using the green target. Power consumption can be measured using watt-meter connected to machine and disks; IO performance can be measured using Filebench in unit of ops/sec. To make the comparison fair, we trace block IO of workloads beforehand and replay the traces on the two systems.

The experiments will be performed under different configuration combinations. The first configuration option is workload. Because different workloads can exhibit very different IO characteristics, the comparison will be performed per workload. We will start with a simple workload, file compression, then extend to file server with large files. Workloads should be large so that most physical disks will be covered.

The second configuration option is file system. File systems with different features such as journalling and allocation policy will behave differently on block IO. We are going to consider Ext2 and Ext3.

The third configuration option is IO scheduler. IO scheduler determines how IO requests are grouped and issued. It might has great impact on the effectiveness of the green target. We plan to consider Anticipatory and Noop Scheduler in our experiment.

5 Related Work

Our study provides a multi-disk (hybrid) system in the purpose of saving energy. To achieve that, we use energy-efficient disk as cache and perform data migration and grouping using information obtained from traces study. We discuss related work that covers the following aspects: save energy, hybrid disk, data grouping and workload trace study.

Save Energy. Energy efficiency of storage systems is an interesting problem that is being actively researched. [15] studies the power consumption of servers when different workloads are running. [8] presents empirical analysis of how lossless compression influences power consumption of servers. [11] presents the first study of power consumption of really running enterprise-scale backup systems. A category of method to save energy is using data replication to decrease either disk head movement [6,7] or the number of active disks being used [21].

Hybrid disk. [1,3,4,20] are studying hybrid disk models, where SSD is used as cache. Among them, only [1] is focusing on the energy efficiency aspect. Although [24] is also studying power aware cache, their cache is in volatile RAM instead of non-volatile disk. [2] is also a hybrid storage system that combines low and high speed disks to save energy.

Data grouping. [23] presents data grouping of working set, however, it only contains analysis of offline traces. It is not implemented or experimented on real system. The Energy-Efficient File System (EEFS) groups files with high temporal access locality [9]. [6] also performed predictive data grouping to reduce head move-

ments for saving energy. [13] used the fact that regularly only a small subset of data is accessed by a system, and migrated frequently accessed data to a small number of active disks, keeping the remaining disks off, which is a kind of data grouping.

Trace study. Trace extraction is a popular method used by system researchers to analyze the system behavior. The analysis can be used to optimize several important features of the system like CPU and memory utilization, I/O throughput, latency, etc. Several reseachers have used trace analysis to find correlations between disk block accesses. Z. Li et al [12] used data mining techniques to find block correlations. T. Li et al [10] used block traces to to run-time modeling to estimate the power consumption of servers. Douglas et al. [5] used efficient algorithms based on machine learning techniques on block traces to spin down disks and extend battery life in mobile computers. V. Torasov [19] used trace analysis to evaluate performance and energy efficiency in file system server workloads extensions.

6 Conclusions

We have finished the code as discussed in Section 2. The target is being tested. We have collected some traces and are analyzing them to figure out parameters, such as extent size, we should use. We are going to replay those traces and measure power consumption to justify our green target does save energy.

References

[1] Timothy Bisson, Scott A. Brandt, and Darrell D. E. Long. A hybrid diskaware spin-down algorithm with i/o subsystem support. In *In Proceedings of the 26th IEEE International Performance, Computing and Communications Conference*, 2007.

[2] Enrique V. Carrera, Eduardo Pinheiro, and Ricardo Bianchini. Conserving disk energy in network servers. In *Proceedings of the 17th annual international conference on Supercomputing*, ICS '03, pages 86–97, New York, NY, USA, 2003. ACM.

[3] Biplob Debnath, Sudipta Sengupta, and Jin Li. Skimpystash: Ram space skimpy key-value store on flash-based storage. In *Proceedings of the 2011 international conference on Management of data*, SIGMOD '11, pages 25–36. ACM, New York, NY, USA, 2011.

[4] Biplob Debnath, Sudipta Sengupta, Jin Li, David J. Lilja, and David H. C. Du. Bloomflash: Bloom filter on flash-based storage. In *Proceedings of the 2011 31st International Conference on Distributed Computing Systems*, ICDCS '11, pages 635–644, Washington, DC, USA, 2011. IEEE Computer Society.

[5] Fred Douglis, P. Krishnan, and Brian N. Bershad. Adaptive disk spin-down policies for mobile computers. In *Proceedings of the 2nd Symposium on Mobile and Location-Independent Computing*, MLICS '95, pages 121–137, Berkeley, CA, USA, 1995. USENIX Association.

[6] David Essary and Ahmed Amer. Predictive data grouping: Defining the bounds of energy and latency reduction through predictive data grouping and replication. *Trans. Storage*, 4(1):2:1–2:23, May 2008.

[7] Hai Huang, Wanda Hung, and Kang G. Shin. Fs2: dynamic data replication in free disk space for improving disk performance and energy consumption. *SIGOPS Oper. Syst. Rev.*, 39(5):263–276, October 2005.

[8] R. Kothiyal, V. Tarasov, P. Sehgal, and E. Zadok. Energy and Performance Evaluation of Lossless File Data Compression on Server Systems. In *Proceedings of the Second ACM Israeli Experimental Systems Conference (SYSTOR '09)*, Haifa, Israel, May 2009. ACM.

[9] Dong Li and Dong Li Phd. High performance energy efficient file storage system, 2006.

[10] Tao Li and Lizy Kurian John. Run-time modeling and estimation of operating system power consumption. *SIGMETRICS Perform. Eval. Rev.*, 31(1):160–171, June 2003.

[11] Z. Li, K. M. Greenan, A. W. Leung, and E. Zadok. Power Consumption in Enterprise-Scale Backup Storage Systems. In *Proceedings of the Tenth USENIX Conference on File and Storage Technologies (FAST '12)*, San Jose, CA, February 2012. USENIX Association.

[12] Zhenmin Li, Zhifeng Chen, Sudarshan M. Srinivasan, and Yuanyuan Zhou. C-miner: Mining block correlations in storage systems. In *In Proceedings of the 3rd USENIX Symposium on File and Storage Technologies (FAST 04*, pages 173–186, 2004.

[13] Eduardo Pinheiro and Ricardo Bianchini. Energy conservation techniques for disk array-based servers. In *Proceeding of the 18th International Conference on Supercomputing (ICS 2004)*, pages 68–78, 2004.

[14] G. Schulz. Storage industry trends and it infrastructure resource management (irm), 2007. www.storageio.com/DownloadItems/CMG/MSP CMG May03 2007.pdf.

[15] P. Sehgal, V. Tarasov, and E. Zadok. Evaluating performance and energy in file system server workloads extensions. In *Proceedings of the Eighth*

USENIX Conference on File and Storage Technologies (FAST '10), pages 253–266, San Jose, CA, February 2010. USENIX Association.

[16] Muthian Sivathanu, Vijayan Prabhakaran, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. Improving storage system availability with d-graid. *Trans. Storage*, 1(2):133–170, May 2005.

[17] Andy Soon. Green & beyond: Data center actions to increase business responsiveness and reduce costs, 2009. http://www.slideshare.net/IBMAsean/green-beyond-data-center-actions-to-increase-business-responsiveness-and-reduce-costs.

[18] Gokul Soundararajan, Vijayan Prabhakaran, Mahesh Balakrishnan, and Ted Wobber. Extending ssd lifetimes with disk-based write caches. In *Proceedings of the 8th USENIX conference on File and storage technologies*, FAST'10, pages 8–8, Berkeley, CA, USA, 2010. USENIX Association.

[19] V. Tarasov, K. S. Kumar, J. Ma, D. Hildebrand, A. Povzner, G. Kuenning, and E. Zadok. Extracting Flexible, Replayable Models from Large Block Traces. In *Proceedings of the Tenth USENIX Conference on File and Storage Technologies (FAST '12)*, San Jose, CA, February 2012. USENIX Association.

[20] Vadim Tkachenko. Flashcache: First experiments, May 2010. http://www.mysqlperformanceblog.com/2010/05/10/flashcache-first-experiments/.

[21] Charles Weddle, Mathew Oldham, Jin Qian, and An i Andy Wang. Paraid: The gearshifting power-aware raid. In *In Proc. USENIX Conference on File and Storage Technologies (FAST 07*, pages 245–260. USENIX Association, 2007.

[22] Wikipedia. Device mapper, May 2012.

[23] Avani Wildani, Ethan L. Miller, and Lee Ward. Efficiently identifying working sets in block i/o streams. In *Proceedings of the 4th Annual International Conference on Systems and Storage*, SYSTOR '11, pages 5:1–5:12, New York, NY, USA, 2011. ACM.

[24] Qingbo Zhu, Francis M. David, Christo F. Devaraj, Zhenmin Li, Yuanyuan Zhou, and Computer Science. Reducing energy consumption of disk storage using power-aware cache management. In *In Proceedings of the International Symposium on High-Performance Computer Architecture (HPCA), Febuary*, pages 118–129, 2004.