Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Beyond mirroring: multi-version disk array with improved performance and energy efficiency*

Bo MAO[1], Su-zhen WU‡[2], Dan FENG[3]

([1]*Department of Computer Science and Engineering, University of Nebraska-Lincoln, Lincoln 68588, USA*)

([2]*Computer Science Department, Xiamen University, Xiamen 361005, China*)

([3]*College of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China*)

E-mail: maobo.hust@gmail.com; suzhen@xmu.edu.cn; dfeng@hust.edu.cn

Received Nov. 26, 2010; Revision accepted May 18, 2011; Crosschecked July 29, 2011

Abstract: Performance and power consumption are two important design objectives for data centers consisting of thousands or tens of thousands of disks (or disk arrays). To leverage the two objectives, in this study we propose a multi-version disk array (MDA). The main idea of MDA is to exploit the I/O workload characteristics to guide the replication strategy by replicating multiple versions of the popular data blocks and simply offloading the write data to the free space of the reserved version region, thus achieving high performance in the burst period and low power consumption in the idle period. Our prototype implementation of MDA and the performance evaluations show that the performance of MDA outperforms that of traditional RAID10 by up to 34.4% and 42.3% in terms of the average response time for the online transaction processing (OLTP) application I/O and search engine I/O, respectively. Moreover, the energy efficiency of MDA outperforms that of RAID10 by up to 48.7% and 36.4%, respective to the aforementioned measures.

Key words: Storage systems, Disk arrays, Power consumption, Performance evaluation
doi:10.1631/jzus.C1000407 **Document code:** A **CLC number:** TP316

1 Introduction

With the development and application of multi-core processors in computer systems, the performance gap between CPU and storage continues to widen. For the slow mechanical nature of hard disk drivers, the importance of optimizing the disk performance has been well recognized. As a result, many optimization techniques, such as caching, prefetching, and parallel I/O, have been introduced. Redundant array of independent disks (RAID) (Patterson *et al.*, 1988), which uses the parallel I/O technique to improve the storage performance and reliability, is widely used. On the other hand, the energy costs and cooling infrastructures are becoming an increasing fraction of the total cost ownership (TCO) in the data center. The storage subsystem can consume 27% of the total energy, which has increased by 60% annually (Pinheiro *et al.*, 2006). The cost of the cooling systems also increases with energy consumption (Zhu *et al.*, 2005).

Moreover, the performance and energy consumption of the storage systems are inconsistent with each other. It is desirable to achieve high performance with much more disks. However, for energy efficiency, storage systems require that the active disks should be as few as possible. An effective balance of the two design objectives is difficult, but important.

With the declining price of commodity disk drives and the increasing capacity of a single disk, it is worthwhile to consider the mirroring-based disk

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (No. 61100033), the US National Science Foundation (Nos. NSF-CNS-1016609 and NSF-IIS-0916859), and the Changjiang Innovative Group of Education of China (No. IRT0725)

redundancy as an effective alternative to the more popular parity-based disk redundancy (e.g., RAID5 and RAID6) (Gray and Shenoy, 2000; Zhang *et al.*, 2002). In addition to providing a read bandwidth twice that of the latter, mirroring-based disk arrays are more efficient than parity-based disk arrays in processing small write requests. For example, high performance storage systems, such as the general parallel file system (GPFS) (Schmuck and Haskin, 2002) and Ceph (Weil *et al.*, 2006), use the mirroring technique to store the client data over thousands of disks. Moreover, recent studies indicate that mirroring-based disk arrays are more advantageous than parity-based disk arrays, as the former avoids data loss caused by parity pollution (Krioukov *et al.*, 2008) and parity inconsistencies (Bairavasundaram *et al.*, 2008) that are inevitable in the latter.

On the other hand, storage system designers must consider the characteristics of application accesses. The burstiness (Ruemmler and Wilkes, 1993; Riska and Riedel, 2006; Mi *et al.*, 2008) and popularity (Tian *et al.*, 2007) of accesses are two common and important characteristics of storage systems. Real-life workloads exhibit significant idleness and burstiness characteristics: periods of high utilization alternate with periods of little external load. The access locality in I/O workloads has been well known in the literature. Gomez and Santonja (2002) analyzed three sets of I/O traces provided by Hewlett-Packard Labs and showed that some blocks are extremely hot and popular while other blocks are rarely or even never accessed.

Previous research suggested that the burstiness often results in dramatic degradation of the performance user perceives, because the burstiness increases the queue length (Batsakis *et al.*, 2008; Mi *et al.*, 2008). Moreover, it is not worthwhile to save energy in the high load period (Gurumurthi *et al.*, 2003; Weddle *et al.*, 2007). All these investigations imply that the storage system designers should optimize the performance in the burst period and optimize the energy efficiency in the idle period.

To address the performance issue of large-scale data centers and take power consumption into consideration, in this study, we propose and evaluate a high performance and energy-efficient disk array architecture, called multi-version disk array (MDA). The design principle of MDA is to improve the performance in the burst period and reduce the power consumption in the idle period. The main idea of MDA is to exploit the I/O workload characteristics to guide the replication strategies by replicating multiple versions of popular read data blocks, thus achieving high performance in the burst period. Moreover, MDA spins down many disks to save energy in the idle period. Our prototype implementation of MDA and performance evaluations show that the performance of MDA outperforms that of RAID10 by up to 34.4% and 42.3% in terms of average response time for the online transaction processing (OLTP) application I/O workload and search engine I/O workload, respectively. Moreover, the energy efficiency of MDA significantly outperforms that of RAID10 by up to 48.7% and 36.4% for the two workloads, respectively.

2 Background

2.1 Workload characteristics

A good understanding of the I/O workload characteristics can provide a useful guideline for the design of storage systems. Many researches have revealed that the burstiness and popularity of accesses are two common and important characteristics of storage systems (Ruemmler and Wilkes, 1993; Arlitt and Williamson, 1996; Riska and Riedel, 2006). Researchers have extensively collected and analyzed the disk-level traces. Fig. 1 plots the access patterns of two applications, financial trace provided by UMass Trace Repository and computer research workload (Cello-99) provided by Hewlett-Packard Labs. We can see that the application accesses are a mix of burstiness and idleness in terms of I/O intensity: periods of high utilization alternate with periods of little external load. With the upper-layer optimizing techniques, such as buffer and request scheduling, the write requests seen at the disk level are usually bursty.

Arlitt and Williamson (1996) analyzed a Web server case and found that the access locality is one of the key characteristics of Web workloads. They also observed that 10% of the files accessed account for 90% of the requests and 90% of the bytes transferred. They further found that 20%–40% of the files are accessed only once for the Web workloads. Based on these observations, there are many studies that have identified the popular data (Hsieh *et al.*, 2006) and used the popular characteristics to im-

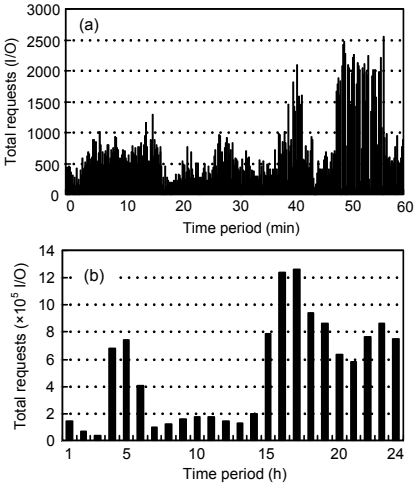

Fig. 1 Application access patterns: (a) financial application (Financial1.spc); (b) computer research workload (Cello-99, 06/01)

prove the performance of storage systems (Wang and Hu, 2002; Tian *et al.*, 2007). WOLF (Wang and Hu, 2002) sorts the active and inactive data in the memory into different segmental buffers, thus reducing the garbage collection overhead to improve the log-structured file system (LFS) performance. By using the I/O access popularity characteristics at the block level, popularity-based multi-threaded reconstruction optimization (PRO) (Tian *et al.*, 2007) reduces the reconstruction time by up to 44.7% and the average user response time by 3.6%–23.9% simultaneously.

2.2 Replication depth

The replication depth has a significant impact on the performance of storage systems for both read and write accesses, but with different effects. Fig. 2 shows the results. The user I/O requests are generated by Iometer with 20% sequential and 8 KB request size. The mix of read and write is generated with a 60%/40% read/write ratio.

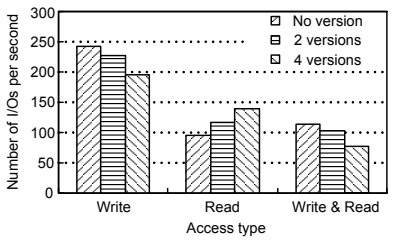

Fig. 2 The effect of the replication depth on the performance of reads and writes

From Fig. 2, we can see that if more versions are adopted, the read accesses can achieve higher performance while the write accesses achieve lower performance. The reason is that the read accesses can be gained from the read balance with multiple versions. For write accesses, if the number of versions increases, more I/O requests should be processed and thus the performance degrades. For the mix of read and write accesses, the performance also degrades when the number of versions increases. These phenomena imply that we should make the versions of the read and write data blocks differently. Based on this observation, the popular read blocks are replicated multiple versions and the write data blocks are replicated only two versions for the purpose of reliability in MDA.

2.3 Performance issue of popular data concentration

Popular data concentration (PDC) (Pinheiro and Bianchini, 2004) concentrates the loads by taking advantage of the heavily skewed file access frequencies (data popularity) to save energy of disk array servers. It puts the most popular data on the first disk, the second most popular data on the second disk, and so on. Disks are powered off in PDC based on an idleness threshold. However, PDC can incur substantial performance degradation due to the load concentration, even when all disks remain in the active mode (Zhu *et al.*, 2005; Weddle *et al.*, 2007; Mao *et al.*, 2008).

As described in Section 2.1, the popular or hot data blocks account for most accesses of applications. Since the popular data are concentrated on a few disks, the system loads are skewed on these few disks. As a result, the I/O queues of these disks are significantly large and the system performance that the user perceives is affected even all disks are active.

One design fundamental of computer systems is Amdahl's law, i.e., "make the common case fast!" (Hennessy and Patterson, 2006). Accesses of the popular data blocks account for most accesses of applications, so we should optimize these accesses to improve the performance. It is also the reason why PDC suffers from a performance problem. The intuitive idea is to replicate the popular data blocks in multiple versions in different disks to exploit the advantage of disk parallelism and read balance.

Disk-based storage is one of the largest power consumers and is also the performance bottleneck in the computer technology industry. With the declining price of commodity disk drives and the increasing capacity of a single disk, the mirroring-based disk redundancy is an effective alternative to the more popular parity-based disk redundancy. The storage system designers must take the characteristics of the application accesses into consideration. Thus, the popular read data should be kept with multiple versions to improve the performance. The write data can be simply written to two locations for the purpose of performance and reliability. Since the data have multiple versions, more disks can be turned into sleep mode to save energy in the system idle period. In this way, both performance and energy efficiency of the disk array are improved.

3 Multi-version disk array

MDA is designed for both read-intensive workloads and write workloads. In MDA, writes are replicated with two versions to reduce extra overhead, improve performance, and maintain reliability simultaneously. The data blocks that have been read many times are replicated in multiple versions to improve performance and energy efficiency. In this section, we present the architecture and data structures of MDA, and discuss the state management and request process in MDA.

3.1 Architecture of MDA

Fig. 3 shows the architecture of MDA. In our design, MDA is based on the mirroring-based disk array. An augmented module is embedded into the RAID controller software with four functional modules: popular data identifier, power management, data movement, and request director. The popular data identifier is responsible for identifying the popular read data blocks. The power management is responsible for powering off and powering on the disks based on the access patterns. The data movement module is responsible for duplicating the hot data blocks from the data regions to the version regions. The request director is responsible for directing the requests based on the system states according to the version map table. The details of directing the request will be described in Section 3.3. The version map table contains records of the list of the hot read data blocks and the redirected write data blocks in the version regions. The version map table should be kept in the non-volatile memory to avoid data loss in case of power failure. Thus, the additional cost to build MDA is the cost of the marking memory that is used to store the version map table. The marking memory uses NVRAM that is commonly configured in the storage controllers.

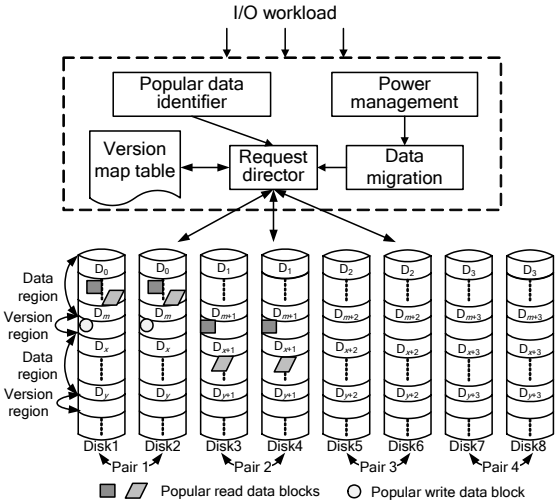

Fig. 3 The architecture of multi-version disk array (MDA)

The disks in MDA are partitioned into data regions and version regions. Converting the logical address to a physical address is based on the data regions, while the version regions are transparent to the logical level. The hot read data blocks are replicated from one pair to the adjacent pair. For example, the hot read data blocks in pair 1 are replicated to the version regions on pair 2 of disks, and the hot read data blocks in pair 2 are replicated to the version regions on pair 1 of the disks (Fig. 3). For pair 3 and pair 4, the process method is the same. In the idle period, only one disk of the four disks in pair 1 and pair 2 needs to be active, as the most popular data blocks are concentrated in the active disk. Moreover, the active disk can be active one after another among the four disks to avoid the frequent disk spinning up/off operations that impact on the reliability of disks (Xie and Sun, 2008). In many application environments, 80% of the accesses are always directed to 20% of the data. The phenomenon has been known as Pareto's principle or the '80/20 rule' (Gomez and Santonja, 2002). Thus, the space ratio of the data region to the version region is 4:1 in our design.

3.2 Data structure

The main data structure in MDA is the version map table that contains a number of version record entries and is managed by the least frequently used (LFU) list (Fig. 4).

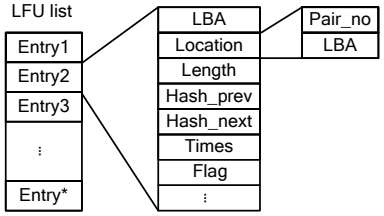

Fig. 4 Data structure of the version map table

The main variables in the entry are explained as follows: LBA indicates the offset of the data block in the logical version of the disk device and serves as the search key of the hash table; Location represents the offset of the data block in the version regions and contains two variables: Pair_no indicates the corresponding pair number, and LBA indicates the location of the replication in the disks of the pair; Length indicates the length of the data block; Two pointers Hash_prev and Hash_next are used to link the hash table; Times indicates the frequency (or times) of the entry that has been accessed to help manage the LFU list; Flag indicates the type of the data block ('0' for read and '1' for write).

Due to the critical importance in request processing and recovering operations (from disk failure), these data structures must be kept in an NVRAM in case of power failure. MDA needs only to maintain one entry per request unit rather than per disk sector. And the space that the NVRAM needs is very small. For example, with 2048 entries in the LFU list, the total memory needed is about 66 KB (each entry uses 33 bytes). To the best of our knowledge, the NVRAM is commonly equipped in the storage controller. Moreover, the memory size needed can be further reduced by periodically flushing the contents of the LFU list to the superblock on the disks. Thus, this memory overhead can be neglected and does not affect the system performance significantly.

3.3 State management

There are two states in MDA: the performance state when all disks are active, and the energy efficient state when only a few disks are active. For example, as shown in Fig. 3, the performance state indicates the period when all the eight disks are active and the energy efficient state indicates the period when only four or two disks are active (such as disk1 and disk5).

The design principle of MDA is to improve the performance in the system burstiness period and improve the energy efficiency in the idle period. Thus, the state transfers according to the intensity of the I/O workload. When the load of the system decreases to a threshold, the data movement module begins to replicate the hot read data blocks to the version regions and records them in the version map table, and then notifies the power management module to power off some disks, and the system transfers into the energy efficient state. When the load of the system exceeds a threshold, the power management module spins up some or all disks to improve the performance. The threshold value depends on the application and is adjusted according to the performance achieved.

3.4 Process flow

Processing the requests in the performance state and the energy efficient state are different. We describe them separately.

In the performance state, all disks are active. When receiving a read request, the version map table is first checked to see whether the request is in the table. If the request is in the table, the request director module directs the request to the disk that is chosen by the read balance function, and increases the corresponding Times value in the entry of the version map table. Otherwise, the request director module directs the request to its physical place according to the read balance function. For write requests, the request director module directs the data blocks to their physical places directly. If its information is in the version map table, the corresponding entry should be deleted from the table.

In the energy efficient state, only some disks are active. When receiving a read request, the request director module first checks whether the corresponding disks are in the idle state. If the disks are in the idle state, the request director module checks whether the request is in the version map table and directs the request to the version regions of the other disks. Otherwise, the power management module spins up the disks to serve the read request. When receiving a write request, the request director mod-

ule also checks whether the corresponding disks are in the idle state. If the disks are in the active state, the request director module directs the request to the physical disks directly. Otherwise, the request is directed to the version regions of the active disks and is recorded in the version map table with the Flag value marked as '1'.

No matter in which state, the popular data identifier module monitors the read requests and identifies the popular read data blocks in the disks. When the system decides to transfer into the energy efficient state, these popular read data blocks should be replicated to the version regions.

4 Performance evaluations

4.1 Experimental setup

We have implemented a prototype MDA in the Linux software RAID framework as an independent module. The performance evaluations are conducted on a platform of the server-class hardware with an Intel Xeon 3.0 GHz processor and 1 GB DDR memory. In the system, the disk module is 250 GB Barracuda 7200.10 ST3250310AS SATA disk which consumes 12.6, 9.3, and 0.8 W power in the active, idle, and standby modes, respectively. The disks are connected by the 3ware 9650SE controller. The two traces used in our experiment are the OLTP application (Financial1.spc) and the search engine (WebSearch1.spc), both obtained from the Storage Performance Council (UMass Trace Repository, 2002). The characteristics of the two traces are shown in Table 1.

Table 1 The trace characteristics

Trace	Read ratio	IOPS	ARS (KB)
Financial	32.8%	69	6.2
WebSearch	100%	113	15.1

IOPS: number of IOs per second; ARS: average request size

The performance evaluations use RAIDmeter (Tian *et al.*, 2007), a block-level trace replay tool with functions of replaying the traces and evaluating the I/O response time of storage devices. We cannot obtain the I/O throughput results since the traces are only requests with different request sizes. The average response time is appropriate and sufficient for the evaluations. For the footprint limit of the traces, we configure each disk to be a capacity of 50 GB in our experiment. The traces are replayed at their initial speed. The energy consumed by disks in the active and idle modes is charged on a per-request basis. To measure the performance impact by the various disk array architectures, we compare the average response time of I/O requests. When measuring the energy consumption, we compare the average power (i.e., the energy consumption per time unit) at each small time interval (one minute in our experiment). The evaluation method has been used in previous studies (Zhu *et al.*, 2005; Mao *et al.*, 2008).

4.2 Evaluation results

We conduct the experiments on RAID10, GRAID, and MDA with the same capacity (200 GB) and stripe unit size (4 KB), driven by the two traces. In terms of average response time, MDA outperforms RAID10 by up to 34.4% for the Financial1.spc trace and up to 42.3% for the WebSearch1.spc trace (Fig. 5).

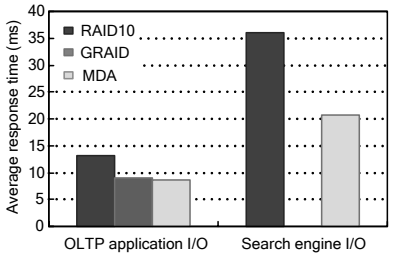

Fig. 5 Comparison of the average response time

The reasons for the performance improvement for the two traces are different. For the Financial1.spc trace, the write data are simply written in the free space of the version region. The seek delay is quite short and thus the performance is improved. For the WebSearch1.spc trace, as the popular read data are copied to multiple versions, many disks serve the read requests concurrently. Thus, the performance is improved by the parallel processing at the disk array level. Though there is a large space overhead for the WebSearch1.spc trace due to the multiple versions of read data blocks, MDA keeps multiple versions of only the hottest data blocks. Moreover, it is common that the hottest data blocks in the applications amount to only less than 20% storage space, but account for 80% of total accesses. Prior studies have indicated that there is about 50% unused storage space in the storage systems and the unused space is exploited to improve the performance and energy efficiency (Weddle *et al.*, 2007).

Thus, the space overhead of MDA is acceptable. On the other hand, since GRAID is optimized for the mixed read/write applications, it performs similar to RAID10 for the WebSearch1.spc trace that absolutely consists of read requests. We omit the performance evaluation of GRAID for the WebSearch1.spc trace in this study.

To fully understand how much energy is consumed by the different disk array schemes (RAID10, GRAID, and MDA), we plot the percentage of the energy saving per minute normalized to RAID10 (Fig. 6). There are eight data disks for both RAID10 and MDA, but one more log disk for GRAID. Fig. 6 shows that MDA saves more energy than GRAID by 17.8% for the Financial1.spc trace. The reason is that, in the system idle period, MDA has only four active disks to serve the requests. However, GRAID needs one more disk to serve as the log disk, thus increasing the energy consumption of the system. Moreover, simply issuing the write data to the free space of the version region in MDA allows the disks to stay in the active mode more shortly, thus achieving more energy savings than GRAID. Compared with RAID10, MDA saves more energy for both the Financial1.spc trace and the WebSearch1.spc trace by up to 48.7% and 36.4%, respectively. It demonstrates that, the original design objectives of RAID are the performance and reliability and do not include the energy efficiency issue.

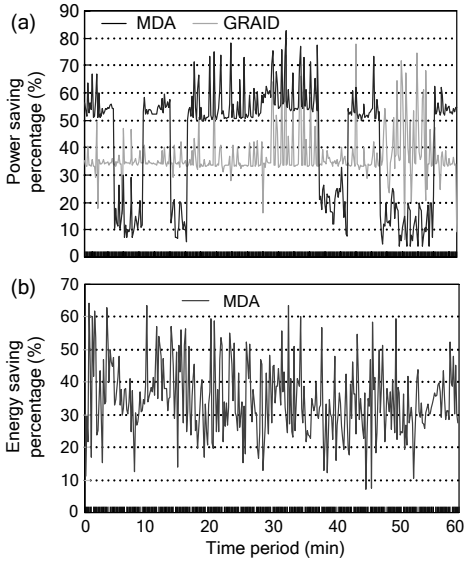

Fig. 6 The percentage of the energy saving per minute normalized to RAID10: (a) OLTP application I/O; (b) search engine I/O

5 Related work

RAID (Patterson *et al.*, 1988) was proposed to improve performance and reliability over a single device. Its original design objectives did not include energy efficiency. Different RAID levels provide variable tradeoffs among performance, reliability, and energy consumption. We briefly review the published work for improving the performance and energy efficiency of disk arrays.

5.1 High performance disk arrays

The schemes most related to ours are DDM (Orji and Solworth, 1993) and EW-Array (Zhang *et al.*, 2002). Doubly distorted mirrors or write anywhere (DDM) provides an alternative way to improve the performance of write requests in a mirror. It performs the write requests initially to the rotationally optimal but variable locations and propagates them to the fixed locations later. EW-Array, also based on RAID10, effectively integrates the mirroring and eager-writing techniques. However, the knowledge of the rotationally optimal, but variable, locations or the free space close to the current disk head position at the block level is difficult and costly (Sivathanu *et al.*, 2004).

HP AutoRAID (Wilkes *et al.*, 1996) and Hot Mirroring (Mogi and Kitsuregawa, 1996) are two schemes that combine the RAID1 and RAID5 layouts to improve the performance. However, they focus on improving the performance by keeping the hot data blocks in the mirroring regions and the cold data blocks in the parity regions. The power consumption is not considered. AFRAID (Savage and Wilkes, 1996) is designed to eliminate the small update penalty that plagues the traditional RAID5 disk arrays. It applies the data update immediately, but delays the parity update to the next quiet period between bursts of the client activities.

All these schemes consider only the performance without awareness of the power consumption issue. FS2 (Huang *et al.*, 2005), which utilizes the free space available in the disks to dynamically replicate the data blocks, improves both performance and energy efficiency. However, FS2 is an optimization at the file system level, and is not adaptable. MDA is located at the block level; thus, it can be applicable to any file system.

5.2 Energy efficient disk array

Table 2 compares the performance, reliability, and energy efficiency of different disk array schemes that take power consumption into consideration. PDC (Pinheiro and Bianchini, 2004), EERAID (Li and Wang, 2004), and PARAID (Weddle *et al.*, 2007) are designed to gain better energy efficiency with minimal impact on the performance. However, since the redundancy degree of the disk array is affected, its reliability somewhat decreases. GRAID (Mao *et al.*, 2008) is proposed to improve the energy efficiency of RAID10 by adding an extra log disk and take reliability into consideration. However, it is unsuitable for read intensive application environments, such as the search engine I/O workload.

Table 2 Summary of the different disk array schemes

Scheme	Performance	Reliability	Energy efficiency
RAID0	High	Low**	High
RAID1/10	Medium	High	Low
RAID5/6	MH*	Medium	MH
PDC	ML	Low**	High
EERAID	MH	ML	High
PARAID	MH	MH	High
GRAID	Medium	High	High
MDA	High	High	High

* Slow small write; ** No redundancy. MH: medium high; ML: medium low

Massive array of idle disks (MAID) (Colarelli and Grunwald, 2002) and Pergamum (Storer *et al.*, 2008) are designed for the energy efficient archival storage systems, which are not realistic for online transaction processing environments. Son *et al.* (2005) proposed an energy efficient approach that uses the finer-grained control over the data layout on the disks, tuning it on a per-application basis. Applications are instrumented and then profiled to obtain the access sequences of the disk array, which are then used to determine the optimal disk layouts by computing the optimal stripe factor, stripe size, start disk, and so on. However, the wisdom of combining the disk layout to the application seems debatable since it carries a significant preprocessing overhead and suffers from the inflexibility problem. Hibernator (Zhu *et al.*, 2005) and Write Off-loading (Narayanan *et al.*, 2008) optimize the energy efficiency of RAID-structured storage systems.

MDA is based on the above researches and combines both Write Off-loading and multi-version of the popular read data techniques to further improve the performance and energy efficiency of disk arrays simultaneously.

6 Conclusions

For large-scale data centers that consist of thousands or tens of thousands of disks, the performance and power consumption are two important design objectives. In this paper, we propose and evaluate a multi-version disk array (MDA) to improve the performance in the burst period and reduce the power consumption in the idle period. In detail, MDA exploits the I/O workload characteristics to guide the replication strategies by replicating multiple versions of the popular read data blocks. Through the read balance function, MDA improves the performance in the burst period. Moreover, in the idle period, MDA can spin down more disks to save energy. The trace-driven experiments conducted with our lightweight prototype implementation of MDA show that MDA outperforms RAID10 by up to 34.4% and 42.3% in terms of average response time for the OLTP application I/O workload and the search engine I/O workload, respectively. Moreover, MDA significantly outperforms that of RAID10 by up to 48.7% and 36.4% in terms of energy efficiency.

References

Arlitt, M., Williamson, C., 1996. Web Server Workload Characterization: the Search for Invariants. Proc. ACM SIGMETRICS Int. Conf. on Measurement and Modeling of Computer Systems, p.126-137. [doi:10.1145/233013.233034]

Bairavasundaram, L.N., Arpaci-Dusseau, A.C., Arpaci-Dusseau, R.H., Goodson, G.R., Schroeder, B., 2008. An analysis of data corruption in the storage stack. *ACM Trans. Storage*, **4**(3):1-28. [doi:10.1145/1416944.1416947]

Batsakis, A., Burns, R., Kanevsky, A., Lentini, J., Talpey, T., 2008. AWOL: an Adaptive Write Optimizations Layer. Proc. 6th USENIX Conf. on File and Storage Technologies, p.67-80.

Colarelli, D., Grunwald, D., 2002. Massive Arrays of Idle Disks for Storage Archives. Proc. ACM/IEEE Conf. on Supercomputing, p.1-11. [doi:10.1109/SC.2002.10058]

Gomez, M.E., Santonja, V., 2002. Characterizing Temporal Locality in I/O Workload. Proc. Int. Symp. on Performance Evaluation of Computer and Telecommunication Systems, p.1-8.

Gray, J., Shenoy, P., 2000. Rules of Thumb in Data Engineering. Proc. 16th Int. Conf. on Data Engineering, p.3-10. [doi:10.1109/ICDE.2000.839382]

Gurumurthi, S., Sivasubramaniam, A., Kandemir, M., Franke, H., 2003. DRPM: Dynamic Speed Control for Power Management in Server Class Disks. Proc.

30th Annual Int. Symp. on Computer Architecture, p.169-179. [doi:10.1109/ISCA.2003.1206998]

Hennessy, J.L., Patterson, D.A., 2006. Computer Architecture: a Quantitative Approach (4th Ed.). Morgan Kaufmann, USA, p.38-44.

Hsieh, J.W., Kuo, T.W., Chang, L.P., 2006. Efficient identification of hot data for flash memory storage systems. *ACM Trans. Storage*, **2**(1):22-40. [doi:10.1145/1138041.1138043]

Huang, H., Hung, W., Shin, K.G., 2005. FS2: Dynamic Data Replication in Free Disk Space for Improving Disk Performance and Energy Consumption. Proc. 20th ACM Symp. on Operating Systems Principles, p.263-276. [doi:10.1145/1095810.1095836]

Krioukov, A., Bairavasundaram, L.N., Goodson, G.R., Srinivasan, K., Thelen, R., Arpaci-Dusseau, A.C., Arpaci-Dusseau, R.H., 2008. Parity Lost and Parity Regained. Proc. 6th USENIX Conf. on File and Storage Technologies, p.127-141.

Li, D., Wang, J., 2004. EERAID: Energy Efficient Redundant and Inexpensive Disk Array. Proc. 11th Workshop on ACM SIGOPS European Workshop, p.1-14. [doi:10.1145/1133572.1133577]

Mao, B., Feng, D., Jiang, H., Wu, S., Chen, J., Zeng, L., 2008. GRAID: a Green RAID Storage Architecture with Improved Energy Efficiency and Reliability. Proc. Int. Symp. on Modeling, Analysis and Simulation of Computers and Telecommunication Systems, p.1-8. [doi:10.1109/MASCOT.2008.4770574]

Mi, N., Casale, G., Cherkasova, L., Smirni, E., 2008. Burstiness in multi-tier applications: symptoms, causes, and new models. *LNCS*, **5346**:265-286. [doi:10.1007/978-3-540-89856-6_14]

Mogi, K., Kitsuregawa, M., 1996. Hot mirroring: a method of hiding parity update penalty and degradation during rebuilds for RAID5. *ACM SIGMOD Rec.*, **25**(2):183-194. [doi:10.1145/235968.233331]

Narayanan, D., Donnelly, A., Rowstron, A., 2008. Write off-loading: practical power management for enterprise storage. *ACM Trans. Storage*, **4**(3):1-23. [doi:10.1145/1416944.1416949]

Orji, C.U., Solworth, J.A., 1993. Doubly distorted mirrors. *ACM SIGMOD Rec.*, **22**(2):307-316. [doi:10.1145/170036.170082]

Patterson, D., Gibson, G., Katz, R., 1988. A case for redundant arrays of inexpensive disks (RAID). *ACM SIGMOD Rec.*, **17**(3):109-116. [doi:10.1145/971701.50214]

Pinheiro, E., Bianchini, R., 2004. Energy Conservation Techniques for Disk Array-Based Servers. Proc. 18th Annual Int. Conf. on Supercomputing, p.68-78. [doi:10.1145/1006209.1006220]

Pinheiro, E., Bianchini, R., Dubnicki, C., 2006. Exploiting Redundancy to Conserve Energy in Storage Systems. Proc. Joint Int. Conf. on Measurement and Modeling of Computer Systems, p.15-26. [doi:10.1145/1140277.1140281]

Riska, A., Riedel, E., 2006. Disk Drive Level Workload Characterization. Proc. Annual USENIX Technical Conf., p.97-103.

Ruemmler, C., Wilkes, J., 1993. UNIX Disk Access Patterns. Proc. USENIX Winter Technical Conf., p.405-420.

Savage, S., Wilkes, J., 1996. AFRAID: a Frequently Redundant Array of Independent Disks. Proc. USENIX Annual Technical Conf., p.27-39.

Schmuck, F., Haskin, R., 2002. GPFS: a Shared-Disk File System for Large Computing Clusters. Proc. 1st USENIX Conf. on File and Storage Technologies, p.231-244.

Sivathanu, M., Bairavasundaram, L., Arpaci-Dusseau, A.C., Arpaci-Dusseau, R.H., 2004. Life or Death at Block-Level. Proc. 6th Conf. Symp. on Opearting Systems Design and Implementation, p.379-394.

Son, S.W., Chen, G., Kandemir, M., 2005. Disk Layout Optimization for Reducing Energy Consumption. Proc. 19th Annual Int. Conf. on Supercomputing, p.274-283. [doi:10.1145/1088149.1088186]

Storer, M.W., Greenan, K.M., Miller, E.L., Voruganti, K., 2008. Pergamum: Replacing Tape with Energy Efficient, Reliable, Disk-Based Archival Storage. Proc. 6th USENIX Conf. on File and Storage Technologies, p.1-16.

Tian, L., Feng, D., Jiang, H., Zhou, K., Zeng, L., Chen, J., Wang, Z., Song, Z., 2007. PRO: a Popularity-Based Multi-threaded Reconstruction Optimization for RAID-Structured Storage Systems. Proc. 5th USENIX Conf. on File and Storage Technologies, p.277-290.

UMass Trace Repository, 2002. OLTP Application I/O and Search Engine I/O. Available from http://traces.cs.umass.edu/index.php/storage/storage.

Wang, J., Hu, Y., 2002. WOLF—a Novel Reordering Write Buffer to Boost the Performance of Log-Structured File Systems. Proc. 1st USENIX Conf. on File and Storage Technologies, p.47-60.

Weddle, C., Oldham, M., Qian, J., Wang, A.A., Reiher, P., Kuenning, G., 2005. PARAID: a gear-shifting power-aware RAID. *ACM Trans. Storage*, **3**(3):245-260. [doi:10.1145/1288783.1289721]

Weil, S., Brandt, S., Miller, E., Long, D., Maltzahn, C., 2006. Ceph: a Scalable, High-Performance Distributed File System. Proc. 7th Symp. on Operating Systems Design and Implementation, p.307-320.

Wilkes, J., Golding, R., Staelin, C., Sullivan, T., 1996. The HP AutoRAID hierarchical storage system. *ACM Trans. Comput. Syst.*, **14**(1):108-136. [doi:10.1145/225535.225539]

Xie, T., Sun, Y., 2008. Sacrificing Reliability for Energy Saving: Is It Worthwhile for Disk Arrays? IEEE Int. Symp. on Parallel and Distributed Processing, p.1-12. [doi:10.1109/IPDPS.2008.4536247]

Zhang, C., Krishnamurthy, X., Yu, A., Wang, R.Y., 2002. Configuring and Scheduling an Eager-Writing Disk Array for a Transaction Processing Workload. Proc. 1st USENIX Conf. on File and Storage Technologies, p.289-304.

Zhu, Q., Chen, Z., Tan, L., Zhou, Y., Keeton, K., Wilkes, J., 2005. Hibernator: helping disk arrays sleep through the winter. *ACM SIGOPS Oper. Syst. Rev.*, **39**(5):177-190. [doi:10.1145/1095809.1095828]