Proc. the 24th Annual ACM Symposium on Applied Computing, March 2009.

DARAW: A New Write Buffer to Improve Parallel I/O Energy-Efficiency

Xiaojun Ruan, Adam Manzanares, Kiranmai Bellam, Xiao Qin†

Department of Computer Science and Software Engineering
Auburn University, Auburn, AL 36830, USA
{xzr0001, acm0008, kzb0008}@eng.auburn.edu, xqin@auburn.edu

Ziliang Zong
Department of Mathematics and Computer Science
South Dakota School of Mines and Technology
Rapid City, SD 57701
ziliang.zong@sdsmt.edu

Abstract

In the past decades, parallel I/O systems have been used widely to support scientific and commercial applications. New data centers today employ huge quantities of I/O systems, which consume a large amount of energy. Most large-scale I/O systems have an array of hard disks working in parallel to meet performance requirements. Traditional energy conservation techniques attempt to place disks into low-power states when possible. In this paper we propose a novel strategy, which aims to significantly conserve energy while reducing average I/O response times. This goal is achieved by making use of buffer disks in parallel I/O systems to accumulate small writes to form a log, which can be transferred to data disks in a batch way. We develop an algorithm - dynamic request allocation algorithm for writes or DARAW - to energy efficiently allocate and schedule write requests in a parallel I/O system. DARAW is able to improve parallel I/O energy efficiency by the virtue of leveraging buffer disks to serve a majority of incoming write requests, thereby keeping data disks in low-power state for longer period times. Buffered requests are then written to data disks at a pre-determined time. Experimental results show that DARAW can significantly reduce energy dissipation in parallel I/O systems without adverse impacts on I/O performance.

1. Introduction

In the past few years, large-scale storage systems have been developed to achieve high I/O performance and large storage capacity for a wide variety of data-intensive applications [6][7][8][16]. Much attention has been paid to the issues of performance and security in storage systems [10][11][13]. Making data disks active even when they are sitting idle is an important avenue to maintain high performance, because disks can immediately start serving disk requests newly arrived. However, this approach can waste a huge amount of energy in large-scale parallel disk systems.

SAC'09, March 8-12, 2009, Honolulu, Hawaii, U.S.A.

This is true especially when there are many long idle periods. Traditional energy conservation techniques (e.g., dynamic power management) improve disk I/O energy efficiency by turning disks into the low-power state if the disks are sitting idle. Unfortunately, the conventional dynamic power management strategies for single disk systems are inadequate for parallel disk systems because of the following three reasons. First, idle periods under some workload conditions are too short to turn disks into a low-power state to conserve energy. Second, although energy can be conserved by frequently place disks into the low-power state, an excessive number of power-state transitions inevitably have adverse impacts on the reliability of parallel I/O systems. Third, numerous power-state transitions impose significant energy overhead as well as response time penalties.

It is evident that the existing dynamic power management strategies ultimately encounter the problem of long power-state transition times and noticeable power state transition energy overhead. Although disk active times in the parallel storage system can be shortened, energy dissipation in the storage system may not necessarily be reduced. This is due the fact that power-state transitions introduce a significant amount of energy overhead.

Recognizing that energy overhead and response time penalties induced by power-state transitions negatively affect energy efficiency of parallel I/O, in this study we seek to reduce the number of power-state transitions for writes processed by a parallel disk system. We focus on write requests, because there exist a considerable number of write-intensive applications like transaction processing, log file updates, and data collection [21]. In this paper, we present the design and implementation of parallel storage systems with buffer disks processing write requests. Specifically, we aim to develop a dynamic request allocation algorithm for writes or DARAW, which dynamically and energy efficiently allocates buffer disks or data disks to serve write requests. Request allocations depend on not only data sets residing in buffer disks contain but also the power states of data disks. Data sets cached in buffer disks will be transferred to corresponding data disks when a set of conditions are satisfied. These conditions may be configured by system administrators to tune the performance of storage systems. Experimental results show that DARAW is conducive to conserving energy consumption in parallel storage systems while efficiently reducing response times of write requests.

2. Related Work

Energy conservation techniques for disk systems have attracted much attention in the past few years. For example, energy dissipation in disk I/O can be efficiently reduced by applying multi-speed disks as the power-state transition penalties are relatively small [1]. Song and Kandemir developed novel energy-aware compilers for multi-speed disks [20]. Although next-generation disks are likely having multiple speeds, most disks utilized today are non-multi-speed disks. It is expected that future generation multi-speed disks are more expensive than conventional disks. The energy conservation technique investigated in this study does not rely on multi-speed disks.

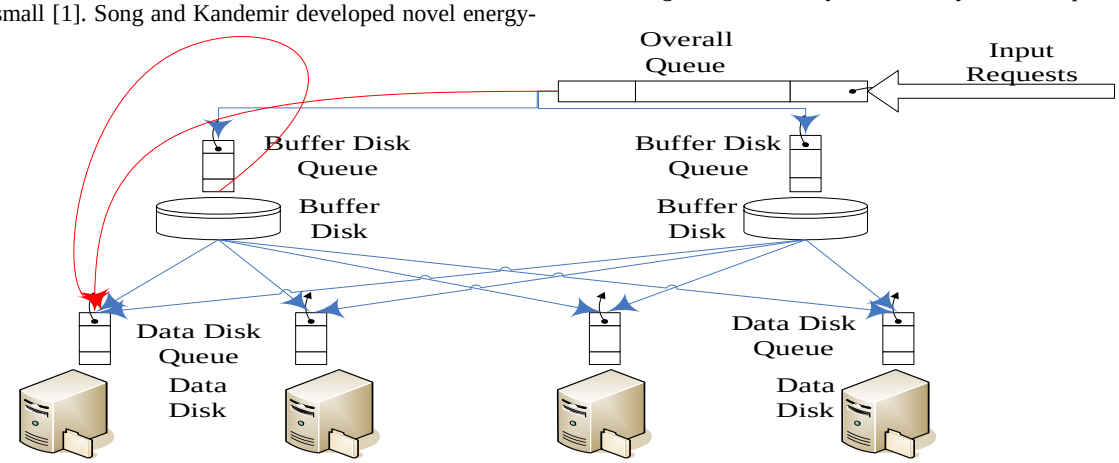

Fig. 1. The architecture of parallel storage system with buffer disks

Modern disks make use of cache to substantially improve disk I/O performance [9]. Our storage system architecture use disks as I/O buffer. Compared with cache, disks are slower and less energy efficient. However, disks are very cost effective and could buffer much more data than cache. Moreover, disks are non-volatile storage, meaning that once data is buffered on disks, it could be considered as safe even a power failure occurs. A research for non-volatile caches is done by Gill and Modha [15]; the research focused on single disk, RAID-10 and RAID-5. It is possible to expend the research to energy-aware parallel storage systems.

To improve parallel disk buffer management, Kallahalla and Varman leveraged a shared buffer to improve I/O performance [14]. Rangaswami *et al.* investigated a way of employing disks to buffer data for streaming media servers in order to bridge the widening performance gap between dynamic random access memory and disk drives in the memory hierarchy [18]. Goyal *et al.* explored the issue of quality of service in the context of storage system caches [3]. The fundamental difference between our research and the above three studies is that the goal of our approach is reducing energy consumption in parallel I/O systems.

If the data size of each request is so large that it is worth to spin up and spin down disks for each request, the traditional power management strategy is an efficient energy conservation technique. However, small and sequential data requests in modern scientific applications are very prevalent [16]. Moreover, small writes cause not only an energy consumption problem but also an efficiency problem [17]. Hence, it is imperative for us to develop an energy saving technique that is suitable to small writes issued to parallel I/O systems.

Please note that our approach can be readily applied to distributed network storage systems, where storage nodes are aggregated together into a larger cohesive storage system [4].

3. Architecture with Write Buffers

In this section, we first introduce our energy-efficient disk architecture. Then, we present a dynamic request allocation algorithm for writes or DARAW. Finally, we build an energy consumption model to quantify energy dissipation in parallel I/O systems.

3.1 Parallel Storage Systems with Buffer Disks

Let us present our energy-efficient disk architecture with buffer disks (see Fig. 1.). This architecture is unique when compared to traditional parallel storage system architectures. We classify disks in a parallel storage system into two categories: buffer disks and data disks. All disks in the system are separated into two distinct layers. Requests issued to the parallel storage system are written temporally into buffer disks first and then be transferred into data disks at appropriate time periods.

Each disk, regardless of buffer or data disks, has its own queue to store incoming requests. In addition, there is an overall request queue, in which all requests enter when they are submitted to the storage system. In most cases, the number of buffer disks is less than the number of data disks. This is because our target goal in this study is to save energy by keeping a small number of active buffer disks while placing a large number of data disks into the low-power state. The ratio of the number of data disks and the number of buffer disks can largely affect the energy efficiency of the parallel storage system. Ideally, the ratio needs to be adjusted on the fly in accordance with workload conditions. In this study, we evaluate impacts of this ratio on energy efficiency of parallel I/O systems.

3.2 The DARAW Algorithm

Now we describe the dynamic request allocation algorithm for writes or DARAW, which was designed in light of the novel disk architecture depicted in Fig. 1. DARAW is an on-line algorithm, which can handle input disk requests without knowing disk access patterns in a priori. In a parallel I/O system with buffer disks, there is a buffer-disk layer and a data disk layer. This indicates that the first phase in the DARAW algorithm is to decide a buffer disk by which a request should be served. After requests are

responded by a buffer disk, it is essential to determine when to transfer the data set from the buffer disk to a corresponding data disk. Therefore, DARAW contains two parts: a buffer-disk layer scheduling scheme and a data-disk layer scheduling scheme.

Fig. 2 outlines the buffer-disk layer scheduling scheme in DARAW. When a write request enters into a buffer-disk queue or is about to be served by a buffer disk, DARAW can process the request in two ways: the request can be served by the buffer disk; or the request can be allocated to and served by the corresponding data disk. DARAW directly allocates the request to the data disk without having the data buffered in the buffer disk if the data disk is active, thereby keeping the data disk in the active-power state without turning off the data disk until all requests in its queue are completed. Hence, the requests targeting at this data disk could be written in the data disk neither going through buffer disk layer nor affording transition penalty. However, if the targeting data disk of a request is sleeping when the request needs to be served, DARAW has to either put the requests into the queue of a buffer disk or have the buffer disk immediately process the request. In this case, DARAW picks up a buffer disk that contains a list of pending requests targeting at the same data disk as the current request. The goal of choosing a buffer disk for the current request in this way is to make the data movement from buffer disks to data disks more efficient. In other words, buffering requests with the same target data disk into one buffer disk makes it possible to move data back to the target data disk in a batch manner. If there is no such a buffer disk, DARAW will pick a buffer disk that has the lightest workload, which is quantified by the data size of queued requests. In Fig. 2, there is a dashed arrow between "write buffer disk" and "write data disk" because this is where Data-Disk scheduling works.

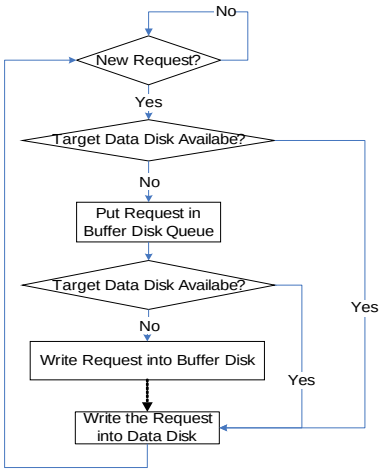

Fig. 2. Buffer-disk layer scheduling in the dynamic request allocation algorithm for writes.

To facilitate the development of DARAW, in what follows we define an important scheduling-control parameter called *Sum of Requests in Buffer*, which is referred to as SRB throughout this paper. Note that incoming write requests are separated into two groups with different writing paths which are illustrated in Fig. 1. The first processing path, shown by the blue arrows, illustrate that requests are served by buffer disks and then data sets are transferred to data disks. The second processing path, shown by red arrows, indicates that requests are directly handled by the data disk layer. When a request going through the first path is written on a buffer disk, we say it is buffered. *Each data disk has its own SRB which contains the number of buffered requests targeting the data disk.* When a request is buffered, the corresponding SRB will be increased by 1. When requests are transferred from a buffer disk to a data disk, the corresponding SRB will be decreased. It is clear that requests going through the second path will not be counted in SRB, because these requests are not buffered. We set up a threshold value SRB_{th} for SRB to decide when DARAW should transfer requests to data disks. For example, if the SRB value of a data disk exceeds SRB_{th}, then DARAW needs to wake up the data disk, to which buffered data should be transferred from buffer disks.

```
if a request comes from overall queue then
    if targeting data disk is not sleeping then
    write the request into targeting data disk
    else
        if buffer disk i having same targeting requests
            write the request in buffer disk i
        else write the lowest load buffer disk
        end if
    end if
end if
if more than 3 working buffer disks are blank then
    turn off 1
end if
for each data disk i in PSS
    if SRB_i >= SRB_th then
    turn on data disk i
        write all requests targeting at i into disk i
    turn off data disk i
    end if
end for
```

Fig. 3 The dynamic request allocation algorithm for writes or DARAW.

Recall that each data disk has a corresponding *SRB* value to track how many data sets have been buffered. This parameter power-state transitions may diminish energy conserved by placing disks into the low-power state. Keeping track of the number of buffered write requests, DARAW aims to substantially reduce the number of unnecessary power transitions in data disks.

Once a request is written into a buffer disk, *SRB* of the target data disk will be increased by 1. If there are enough number of buffered requests for measured by the *SRB* value of a data disk, DARAW writes all the buffered requests into the data disk at one time after turning the data disk into the active state. To judge whether the *SRB* value is large enough, we need to compare *SRB* of each data disk with a threshold value SRB_{th}. The larger the SRB_{th} is set, the greater the reduction in the number of power-state transitions, which ultimately lead to lower energy consumption. Let SRB_i denote the *SRB* value of data disk *i*; let SRB_i^j be the number of requests targeting on data disk *i* while being buffered in buffer disk *j*. SRB_i can be derived from SRB_i^j. In other words, SRB_i is the sum of SRB_i^j of all the buffer disks. Thus, we have

$$SRB_i = \sum_{j=1}^{n} SRB_i^j \ , \qquad (1)$$

where *n* is the number of buffer disks. Note that each *SRB* value in a parallel I/O system is updated continuously.

In the following sections, *i* represents the *i*th buffer disk number; *j* represents the *j*th data disk number; *n* is number of buffer disks; *m* is the number of data disk.

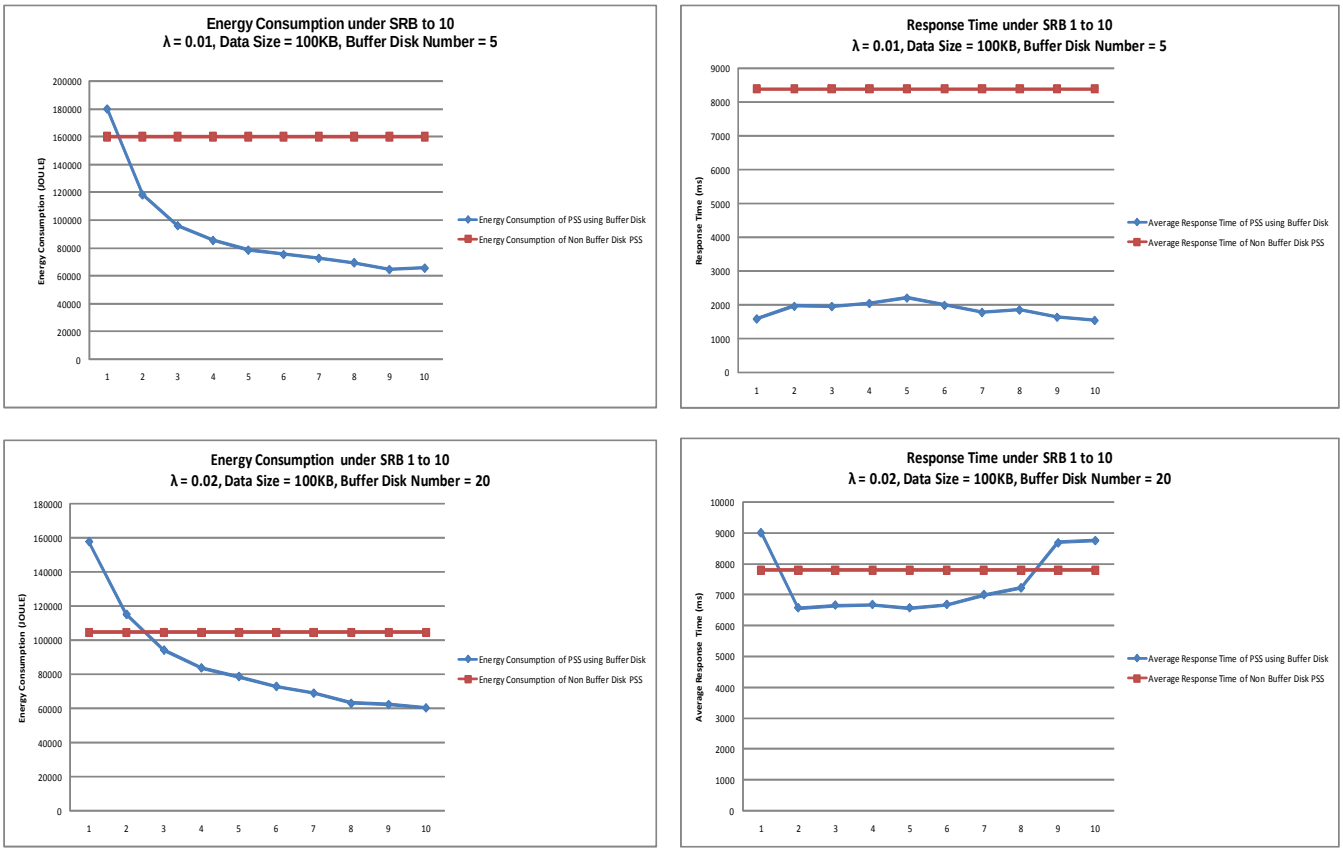

Fig. 4. IBM 36Z15 Ultrastar. Energy consumption and average response time comparison.

4. Performance Evaluation

To evaluate the performance of DARAW, we conducted extensive experiments using various disk I/O traces representing different workload conditions. Thus, the disk I/O traces used in our experiments contain different data sizes, different cylinder numbers, and different arrival rate. Each request in the traces consists of arrival time, data size, cylinder number, targeting data disk, and the like. To evaluate the performance of small writes and large writes, we tested several traces with different data sizes.

Arrival time is one very important parameter in the traces. If the arrival rate is high enough that the parallel disk system is unable to keep up with requests, all queues will overflow in the disk system. To control the arrival rate in the input traces, we need to control the inter-arrival time between each pair of continuous requests.

The inter-arrival time in our traces is controlled by rate parameter λ in exponential distribution. R in Eq. (2) is a random number between 0 and 1, λ is larger than 0, *e* is the base of natural logarithm.

$$Interval\ Time = -\log_e(R/\lambda), \qquad (2)$$

where λ is for the arrival rate, the larger λ is , the heavier the work load the trace provides.

We choose two types of real world hard disks to conduct our experiments. The first one is the high performance disk IBM 36z15 Ultrastar. The second type is the IBM 40GNX Travelstar, which has lower performance numbers. The important modeling parameters for these disks are presented in the following figures.

Table 2 IBM 36z15 Ultrastar and 40GNX Travelstar

System Parameter.	Values
Rotations/Minute	10000RPM, 5400RPM
Working Power	13.5 W, 3W
Standby Power	2.5 W, 0.25W
Spin up Energy	135 Joule, 8.7 Joule
Spin down Energy	13 Joule, 0.4 Joule
Spin up Time	10.9 sec, 3.5 sec
Spin Down Time	1.5 sec, 0.5 sec
Transfer Rate	52.8 MB/s, 25 MB/s

Table 3 Experimental Values

Parameter	Value
Disk Type	IBM 36Z15
Λ	0.01 and 0.02
Data Size/request	100KB
SRB	1-10
Buffer Disk#	5, 20

Data Disk#	100
Trace Size	1000
Disk Type	IBM 40GNX
λ	0.01 and 0.02
Data Size/request	100KB
SRB	1-10
Buffer Disk#	5, 15
Data Disk#	100
Trace Size	1000

Fig. 4 shows the energy consumption and average response time of a parallel disk system with DARAW and the same disk system without DARAW. The results plotted in Fig. 4 indicate that when we increase SRB, more energy can be saved. The results were expected since when the SRB grows, the system can write more requests into data disks with reduced number of power state transitions. However, we also observe that when the SRB equals to one, the energy consumption is even greater than the disk system without DARAW. This interesting tend can be explained as follows. Our parallel disk system has a buffer-disk layer that also consumes energy. If there is insufficient number of requests written into a data disk when a power-state transition occurs, energy conserved cannot offset energy overhead introduced by the buffer disk. When we did the experiment with a trace generated by increasing values of λ, we observe that energy consumptions in both the non-DARAW parallel disk system and the system with DARAW decrease.

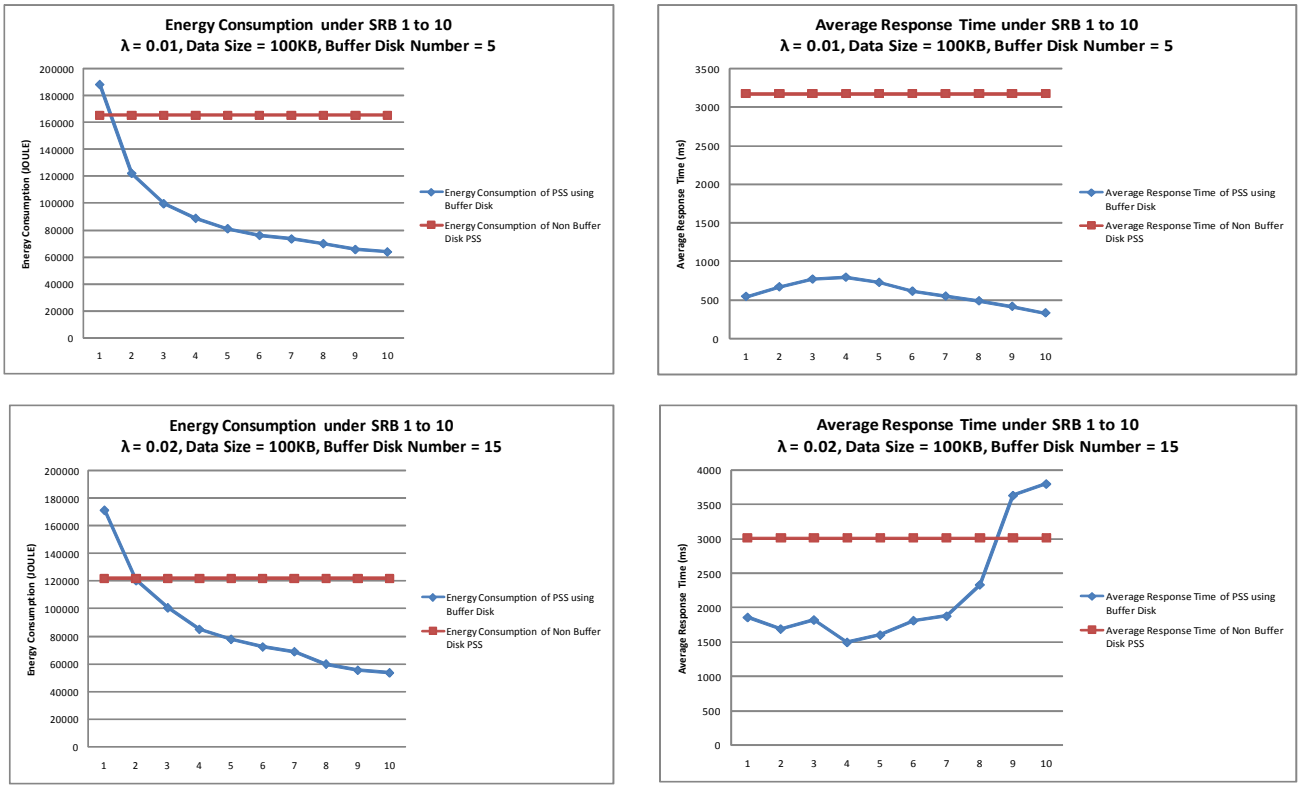

Fig. 5. IBM 40GNX Travelstar. Energy Consumption and Average Response Time Compare

Note that all the traces have the same number of disk requests. This implies the fact that when λ is high, all requests are arriving at the system within a shorter period of time, making all the disks stay in the active state for a shortened time interval. This is the reason behind the result that energy consumption of the system with DARAW when λ is set to 0.02 is slightly smaller than that of the system when λ is 0.01. However, the power consumption of the non-DARAW disk system is significantly smaller when λ is 0.01 as compared to λ = 0.02. Once the arrival rate goes up, each data disk in the non-DARAW system has greater probability to receive a request when it is working. Thus, the number of power-state transitions can be noticeably reduced. When λ is set to 0.02, there is less of an opportunity to simultaneously save energy and satisfy response times. When we increase the number of buffer disks from 5 to 20, DARAW can conserve energy while guaranteeing reasonably short response times. An appealing result shown in Fig. 4 is that compared with the parallel I/O system without DARAW, our approach not only achieves significant energy savings, but also reduces response times. In DARAW, the response time is the time when a request is written in to a data or buffer disk. Since buffer disks can serve coming requests when data disks are sleeping, the response time can be noticeably shortened.

Figs. 4 and 5 show that DARAW works well for parallel I/O systems with both high performance disks and mobile disks. DARAW achieves promising results when the arrival rate is low. When the request arrival rate rises, we can either use high-performance hard drives or add more buffer disks to boost I/O performance. If the arrival rate is high, all data disks are busy serving requests, leaving no opportunity to save energy. As the SRB parameter grows, DARAW is given a greater window of opportunity to conserve energy. However, if the SRB is too large,

it may cause a "traffic jam" inside the parallel I/O system with buffer disks.

5. Conclusion

In this paper, we first presented the design of parallel I/O systems with buffer disks. To conserve energy in parallel I/O systems serving write requests, we developed an algorithm - dynamic request allocation algorithm for writes or DARAW - to energy efficiently allocate and schedule disk requests. This goal is achieved by making use of buffer disks in parallel I/O systems to accumulate small writes to form a log, which can be transferred to data disks in a batch way. DARAW is able to improve parallel I/O energy efficiency by the virtue of employing a small number of buffer disks to serve a majority of write requests, thereby keeping a large number of data disks in low-power state for longer period times. For each data disk in a parallel I/O system, DARAW keeps track of an important parameter called Sum of Requests in Buffer or SRB, which is the number of buffered requests targeting the data disk. The concept of SRB makes it possible to determine how many buffered write requests should DARAW transfer into the corresponding data disk at one time. When SRB is increased, energy savings and response times may both increase. When response times increase due to high workload, we can either use high-performance hard drives or add more buffer disks to boost I/O performance. To quantify the energy efficiency and performance of DARAW, we carried out experiments using parallel I/O systems with buffer disks. Experimental results show that DARAW is conducive to reducing energy dissipation in parallel disk systems while maintaining reasonably low response times. Compared to parallel I/O systems with high-performance disks, parallel I/O systems with mobile disks can achieve better energy efficiency by the virtue of DARAW.

In this research, we focused on parallel I/O systems with homogenous disks. Currently, we are developing write-buffer schemes to improve energy efficiency of parallel I/O systems with heterogeneous disks.

Acknowledgments

The work reported in this paper was supported by the US National Science Foundation under Grants No. CCF-0742187, No. CNS-0757778, No. CNS-0831502, No. OCI-0753305, No. DUE-0621307, and No. DUE-0830831, and Auburn University under a startup grant.

References

[1] E. V. Carrera, E. Pinheiro, and Ricardo Bianchini, "Conserving Disk Energy in Network Servers," *Proc. Int'l Conf. Supercomputing*, 2003

[2] R. Wijayaratne and A. L. Narasimha Reddy, "Integrated QOS management for Disk I/O," *Proc. IEEE Int'l Conf. Multimedia Comp. and Sys.*, 1999.

[3] P. Goyal, D. Jadav, D. S. Modha, and R. Tewari, "CacheCOW: Qos for Storage System Caches," *Proc. Int'l Workshop QoS*, 2003

[4] J. C. Chuang and M. A. Sirbu, "Distributed Network Storage Service with Quality-of Service Guarantees," *Proc. Conf. Internet Society INET'99*, June 1999.

[5] Z. Zong, M. Briggs, N. O'Connor, and X. Qin, "An Energy-Efficient Framework for Large-Scale Parallel Storage Systems," *Proc. Int'l Conf. Parallel and Distributed Processing Symp.*, Mar. 2007.

[6] X. Qin, "Performance Comparisons of Load Balancing Algorithms for I/O-Intensive Workloads on Clusters," *J. Network and Comp. App.*, vol.31, no.1, pp. 32-46, 2008.

[7] S. Lakshmanan, M. Ahamad, and H. Venkateswaran, "Responsive Security for Stored Data," *IEEE Trans. Parallel and Distr. Sys.*, vol. 14, no. 9, Sep. 2003.

[8] M. I. Lutwyche and M. Despont, et al, "Highly Parallel Data Storage System Based on Scanning Probe Arrays," *American Institute of Physics*, 2000.

[9] B. C. Forney, A. C. Arpaci-Dusseau and R. H. Arpaci-Dusseau, "Storage-Aware Caching: Revisiting Caching," *Proc. Int'l Conf. File and Storage Technologies*, 2002.

[10] H. Zhang, W. Wu, X. Dong, D. Qian and L. Dai, "A Study on Data Placement of Extensible Parallel Storage System," *Proc. Int'l Conf. Computer and Information Science*, 2007.

[11] W. Susilo, F. Zhang and Y. Mu, "Privacy-Enhanced Internet Storage," *Proc. AINA*, 2005.

[12] C. Ruemmler and J. Wilkes, "An Introduction to Disk Drive Modeling," *IEEE Computer*, Mar. 1994.

[13] R.Barbe, M. Kallahalla, P. Varman and J.S. Vitter, "Competitive Parallel Disk Prefetching and Buffer Management," *Proc. IOPADS*, 1997.

[14] M. Kallahalla and P. J.Varman, "Improving Parallel-Disk Buffer Management using Randomized Writeback," *Proc. Int'l Conf. Parallel Processing*, 1998.

[15] B. S. Gill and D. S. Modha, "WOW: Wise Ordering for Writes-Combining Spatial and Temporal Locality in Non-Volatile Caches," *Proc. FAST, 2005*.

[16] D. Hildebrand, L. Ward and P. Honeyman, "Large Files, Small Writes, and pNFS," *Proc. ICS, 200*6.

[17] S. H. Baek and K. H. Park, "Matrix-Stripe-Cache-Based Contiguity Transform for Fragmented Writes in RAID-5," *IEEE Trans. Comp., vol. 56, no. 8, pp. 1040-1054*, 2007.

[18] R. Rangaswami, Z. Dimitrijevic, E. Chang and K. Schauser, "MEMS-based Disk Buffer for Streaming Media Servers," *Proc. Int'l Conf. Data Eng.*, 2003.

[19] A. L. N. Reddy, J. Wyllie and K.B. R. Wijayaratne, "Disk Scheduling in a Multimedia I/O System," *ACM Trans. Multimedia Comp., Comm. and App., vol. 1, no. 1*, 2005.

[20] S. W. Song, M. Kandemir, and A. Choudhary, "Software-directed disk power management for scientific applications," *Proc. IPDPS*, April 2005.

[21] M. Nijim, X. Qin, and T. Xie, "Modeling and Improving Security of a Local Disk System for Write-Intensive Workloads," *ACM Trans. Storage*, vol. 2, no. 4, pp. 400-423, Nov. 2006.