No More Energy-Performance Trade-Off: A New Data Placement Strategy for RAID-Structured Storage Systems

Tao Xie and Yao Sun

Department of Computer Science, San Diego State University,
San Diego, CA 92182, USA
{xie, sun}@cs.sdsu.edu

Abstract. Many real-world applications like Video-On-Demand (VOD) and Web servers require prompt responses to access requests. However, with an explosive increase of data volume and the emerging of faster disks with higher power requirements, energy consumption of disk based storage systems has become a salient issue. To achieve energy-conservation and prompt responses simultaneously, in this paper we propose a novel energy-saving data placement strategy, called Striping-based Energy-Aware (SEA), which can be applied to RAID-structured storage systems to noticeably save energy while providing quick responses. Further, we implement two SEA-powered RAID-based data placement algorithms, SEA0 and SEA5, by incorporating the SEA strategy into RAID-0 and RAID-5, respectively. Extensive experimental results demonstrate that compared with three well-known data placement algorithms Greedy, SP, and HP, SEA0 and SEA5 reduce mean response time on average at least 52.15% and 48.04% while saving energy on average no less than 10.12% and 9.35%, respectively.

Keywords: Data placement, energy conservation, response time, RAID.

1 Introduction

Many real-world applications intensively read data stored in large-scale parallel disk storage systems like RAID, Redundant Arrays of Inexpensive Disks. To guarantee the quality of service demanded by end-users, prompt responses to read requests are essential for these applications. For example, a Video-On-Demand (VOD) server has to quickly respond access requests from multiple users so as to provide them with continuous glitch-free video [6]. It is obvious that reducing mean response time of parallel disk storage systems is a must for these applications.

There are a wide variety of ways of reducing the mean response time or improving the system throughput for parallel I/O systems [1][6][10][12]. Data placement, or file assignment, allocating of all the data onto a disk array before they are accessed, is one of such avenues that can significantly affect the overall performance of a parallel I/O system [1][12][19]. Generally, these algorithms place data onto a parallel disk array so that a special cost function or performance metrics can be optimized. While common cost functions include communication costs, storage costs, and queuing costs, popular performance metrics are mean response time and overall system throughput [5]. It is well-known that finding the optimal solution for a cost function or a performance metric in the context of data placement on multiple disks is an NP-complete problem [5]. Thus, heuristics algorithms became practical solutions.

Energy consumption of disk based storage systems has become a salient issue that not only raises the costs but also inversely affects our environment [18]. According to a recent industry report [17], storage devices contribute for around 27% of the total energy consumed by a data center. This problem will become much more severe with an explosive increase of data volume and the emerging of faster disks with higher power requirements. Therefore, energy-conservation and prompt response need to be achieved simultaneously through intelligent data placement. Unfortunately, traditional data placement algorithms such as Greedy [7], Sort Partition (SP) [12], and Hybrid Partition (HP) [12], for parallel disk systems only pursue minimized mean response times and normally ignore energy-conservation. Furthermore, most current energy-saving techniques adversely affect system performance [4][16]. Thus, seeking a good trade-off between energy-saving and graceful performance degradation becomes their feasible goal. Now the question is: can we develop a new data placement strategy so that energy-saving can be achieved without a trade-off of performance?

In this paper we address the problem of energy-saving yet quick-response data placement in a parallel disk storage system where data accesses exhibit Poisson arrival rates and fixed service times. Each data can be viewed as a file, which will be assigned onto an array of disks in a striping manner. We propose a novel energy-saving data placement strategy, called Striping-based Energy-Aware (SEA), which aims at minimizing mean response time and overall energy-consumption simultaneously. The basic idea of SEA is to statically place popular data onto a subset of the disks in the array and assign unpopular data onto the rest of disks. The rationale behind this idea is that the distribution of web page requests generally follows a Zipf-like distribution [12] where the relative probability of a request for the i'th most popular page is proportional to $1/i^{\alpha}$, with α typically varying between 0 and 1 [2][15]. Moreover, the request frequency and the file size are inversely correlated, i.e., the most popular files are typically small in size, while the large files are relatively unpopular [12]. Based on these workload characteristics, we divide all data into two categories: *popular* and *unpopular* according to their popularity weights [15]. Similarly, we separate disks in a disk array into two zones: *hot disk zone* and *cold disk zone*. Disks in hot disk zone are called *hot disks* with popular data, whereas disks in cold disk zone are named *cold disks* with unpopular data. As a result, the overall load balancing between two zones can be achieved, which improves the inter-request parallelism. Next, we employ multi-speed disks in the disk array to save energy. Specifically, hot disks are always running in a higher speed mode with more energy consumption, while cold disks are continuously operating in a lower speed mode with less energy dissipation. Although real multi-speed (more than 2 speeds) hard disks are not widely available in the market yet, a few simple variations of multi-speed disks, such as a two-speed Hitachi Deskstar 7K400 hard drive has recently been produced [9]. For simplicity, in this study we only utilize 2-speed hard disks. Note that once a disk was configured as a hot disk or a cold disk, its operation characteristics such as transfer (read) speed and energy consumption rate is fixed and it cannot be dynamically switched to the other mode during the process of serving requests. Further, to provide quick responses, we combine SEA with RAID structures so that each data (file) is partitioned into multiple same size stripe blocks, which are then allocated across an array of disks. This way all disks in the same zone can simultaneously serve a request. The implication is that the response time of the request can be dramatically decreased due to an enhanced intra-request parallelism.

The rest of the paper is organized as follows. In the next section we discuss the related work and motivation. A system model and an energy consumption model were built in Section 3 and Section 4, respectively. Section 5 presents the SEA strategy and introduces three existing algorithms. In Section 6 we evaluate performance of our algorithms based on synthetic benchmarks. Section 7 concludes the paper with summary and future directions.

2 Related Work and Motivation

Very recently energy-saving for parallel disk storage systems began to draw much attention from research community [3][4][8][10][16]. A multi-speed parallel disk system that can modulate disk speed dynamically was proposed by Gurumurthi et al. [8]. In [10] data replication was used to dynamically place copies of data in free blocks according to the disk access patterns.

Comparing with the energy-efficient techniques mentioned above, data placement shows its unique advantages. First, to save disk energy, it has no need to modify applications. Next, no extra hardware such as cache is necessary. Last, the overhead of data placement strategy is relatively low and it is easy to implement. Attracted by these advantages, a research group led by Son proposed an array of energy-aware disk layout algorithms very recently [18]. Based on our knowledge, their studies are the only results of energy-aware data placement for parallel disk storage systems reported in the literature so far. However, their techniques have some obvious limitations. First, they are only dedicated for array-based scientific applications. Still, there are many other types of disk I/O-intensive applications, where energy conservation and quick response need to be realized simultaneously through data placement. Therefore, a more general energy-response efficiency data placement scheme that can be applied to a wide range of disk I/O-intensive applications is needed. Further, to apply their algorithms, one has to modify compiler to make it be aware of disk layout information. This requirement prevents them from being readily used because it incurs an extra burden for system software programmers. Besides, to better exploit existing power-saving capabilities, their disk layout algorithms need to be combined with application code restructuring to increase length of idle periods. This strategy demands modifications of application's code, and thus brings users additional overhead. As a result, the need of a new energy-response efficiency data placement strategy that bridges the gap between the existing algorithms and the open problems is greatly felt.

In this paper, we are proposing a static heuristic energy-aware strategy SEA, which can be incorporated with RAID structures to generate energy-aware data placement algorithms like SEA0 and SEA5. Our schemes are orthogonal to existing disk layout strategies. First, there is no need to modifying any software using our methods.

Second, our schemes are not dedicated for some particular applications. Thus, they are more general in the sense that they can be applied in multiple application domains. Without loss of generality, we assume that (1) each data is viewed as an independent file; (2) each data is allocated in a striping manner across an array of disks; (3) communication delays between any pair of disks are identical and negligibly small [12]; (4) disk access (read) to each data is modelled as a Poisson process with a mean access rate λ_i; (5) a fixed service time s_i for each data; for example, each read on a data results in a sequential scan of the entire data. For large size data, this assumption is valid because when the basic unit of data access is entire data, seek time, rotation latency, and controller overhead are negligible in comparison with data transfer time.

3 System Model

Data placement algorithms such as Greedy, SP, HP, SEA0, and SEA5 allocate a set of data (hereafter file) onto a group of 2-speed disks so that the mean response time can be minimized. The set of files is represented as $F = \{f_1, ..., f_u, f_v, ..., f_m\}$, which is further categorized into a popular file set $F_h = \{f_1, ..., f_h, ..., f_u\}$ and au unpopular file set $F_c = \{f_v, ..., f_c, ..., f_m\}$ ($F = F_h \cup F_c$ and $F_h \cap F_c = \emptyset$). Since each file will be allocated onto a set of disks in a striping manner, let *sp* denote the size of a stripe in Mbyte and it is assumed to be a constant in the system. A file f_i ($f_i \in F$) is modeled as a set of rational parameters, e.g., $f_i = (s_i, \lambda_i)$, where s_i, λ_i are the file's size in Mbyte and its access rate. In this paper, requests to a file f_u are modeled as a Poisson process with a mean access rate λ_i. Also, we assume each access to file f_i is a sequential read of the entire file, which is a typical scenario in most file systems or WWW servers [11]. Besides, we assume that the distribution of file access requests is a Zipf-like distribution with a skew parameter $\theta = log\frac{X}{100}/log\frac{Y}{100}$, where X percent of all accesses were directed to Y percent of files [12]. The value of $X{:}Y$ is called *skew degree* (SD) in this paper and $\alpha = 1 - \theta$ (see Section 1 for α). In addition, the file access frequency is inversely correlated to the file size. The number of popular files in F is defined as $|F_h| = (1-\theta) * m$. Similarly, the number of unpopular files is $|F_c| = \theta * m$. Thus, the ratio between the number of popular files and the number of unpopular files in F is defined as η

$$\eta = \frac{1-\theta}{\theta}. \tag{1}$$

A disk array storage system consists of a linked group $D = \{d_1, ..., d_e, d_f, ..., d_n\}$ of n independent 2-speed disk drives, which can be divided into a hot disk zone $D_h = \{d_1, ..., d_h, ..., d_e\}$ and a cold disk zone $D_c = \{d_f, ..., d_c, ..., d_n\}$($D = D_h \cup D_c$ and $D_h \cap D_c = \emptyset$). Disks in the hot zone are all configured to their high speed modes, which always run in the high transfer rate t^h (Mbyte/second) with the high active power consumption rate p^h (Joule/Mbyte) and the high idle power consumption rate i^h (Watt). Similarly, disks in cold zone are set to their low speed modes, which continuously operate in the low transfer rate t^l (Mbyte/second) with the low active power consumption rate p^l (Joule/Mbyte) and the low idle power consumption rate i^l (Watt). In the system, a hot disk d_h ($d_h \in D_h$) is modeled as a tuple $d_h = (c, t^h, p^h, i^h)$ where c is the capacity of d_h in GByte. Similarly, a cold disk d_c ($d_c \in D_c$) is modeled as a tuple $d_c = (c, t^l, p^l, i^l)$ where c is the capacity of d_c in GByte. Since we only consider homogeneous disks, all disks have the same capacity c. We assume that disks are always large enough to accommodate files to be assigned on them. Each popular file $f_h \in F_h$ is partitioned in multiple units with the size of each unit equal to sp. All units of f_h will be allocated across the hot disks in a RAID-0 (striping without parity) or a RAID-5 fashion (striping with parity). Similarly, each unpopular file $f_c \in F_c$ is also partitioned into multiple size sp units and then allocated across the cold disks in a RAID-0 or a RAID-5 manner. Let sv_i be the expected service time of file f_i ($f_i \in F$). It can be computed by

$$sv_i = \begin{cases} s_i / t^h, \text{ if } f_i \text{ is popuplar} \\ s_i / t^l, \text{ if } f_i \text{ is unpopular} \end{cases} \tag{2}$$

Since the combination of λ_i and sv_i accurately gives the load of f_i, we define the load h_i of f_i as follows [12]:

$$h_i = \lambda_i \cdot sv_i. \tag{3}$$

The ratio between the number of hot disks and the number of cold disks is defined as γ, which is decided by the ratio between the total load of popular files and the total load of unpopular files as below

$$\gamma = \frac{\sum_{i=1, f_i \in F_h}^{(1-\theta)*m} h_i}{\sum_{j=1, f_j \in F_c}^{\theta * m} h_j} \tag{4}$$

We employ the First-Come-First-Serve (FCFS) scheduling heuristic to schedule arrival requests. Suppose there are totally u requests in a request set, which visits a file set that has been allocated on a disk array. The request set is designated as $R = \{r_1, ..., r_k, ..., r_x\}$, which can be separated into a hot request set $R_h = \{r_b, ..., r_h, ..., r_o\}$ and a cold request set $R_c = \{r_p, ..., r_c, ..., r_s\}$ ($R = R_h \cup R_c$, $R_h \cap R_c = \emptyset$). Each request is modeled as $r_k = (fid_k, a_k)$, where fid_k is the file identifier targeted by the request and a_k is the request's arrival time. For each arrival request, the FCFS scheduler uses the allocation scheme X generated in data placement stage to find the disks on which the target file of the request resides. In fact, the request workload is an m-class workload with each class of requests having its fixed λ_i.

To obtain the response time of a request r_k, two important parameters, the earliest start time and the latest finish time of r_k must be computed. We denote the earliest start time and the latest finish time of r_k by $est(r_k)$ and $lft(r_k)$, respectively. In what follows we present derivations leading to the final expressions for these two parameters. Since each file is distributed across multiple disks in a striping manner, we need to compute the start time and the finish time for each stripe of the file that request r_k is targeting on. Suppose r_k is visiting file f_i, which was distributed on a disk set $\{d_a, ..., d_g, ..., d_w\}$ ($a \leq g \leq w$, $1 \leq a, g, w \leq e$ or $f \leq a, g, w \leq$ n). The stripe set of f_i is represented as $\{ s_i^1, ..., s_i^k, ..., s_i^z \}$, where $z = \frac{s_i}{sp}$. Also, a disk d_g has its own local queue Q_g in the set $\{Q_a, ..., Q_g, ..., Q_w\}$. There are three cases when r_k arrives on disk d_g. First, d_g is idle and Q_g is empty. Second, d_g is busy but Q_g is empty. Third, d_g is busy and Q_g is not empty. Thus, the start time for a strip s_i^k on disk d_g is

$$st_g^k(r_k) = \begin{cases} a_k \text{, if } d_g \text{ is idle, } Q_g \text{ is empty} \\ a_k + r_g \text{, if } d_g \text{ is busy, } Q_g \text{ is empty} \\ a_k + r_g + \sum_{r_p \in Q_g, a_p \le a_k} t_{fid_p} \text{, otherwise} \end{cases} \tag{5}$$

where r_g represents the remaining service time of a request currently running on d_g, and $\sum_{r_p \in Q_g, a_p \le a_k} t_{fid_p}$ is the overall service time of requests in Q_g whose arrival times are earlier than that of r_k. Consequently, ft_g^k (r_k) can be calculated by

$$ft_g^k(r_k) = st_g^k(r_k) + ts_i, \tag{6}$$

where ts_i is the service time of the stripe s_i^k on disk d_g and it can be computed using the following formula

$$ts_i = \begin{cases} sp / t^h \text{, if } d_g \text{ is hot} \\ sp / t^l \text{, if } d_g \text{ is cold} \end{cases} \tag{7}$$

As a result, the earliest start time of request r_k can be obtained by

$$est(r_k)=\min\{ st_g^1(r_k) ,., st_g^k(r_k) , ., st_g^z(r_k) \}. \tag{8}$$

Consequently, the latest finish time of r_k can be calculated by

$$lft(r_k)=\max\{ ft_g^1(r_k) ,., ft_g^k(r_k) ,., ft_g^z(r_k) \}. \tag{9}$$

Hence, the response time of r_k can be obtained

$$t(r_k) = lft(r_k) - est(r_k). \tag{10}$$

Thus, the mean response time of the request set R is expressed as below

$$mrt(R) = \sum_{k=1}^{x} rt(r_k) \Big/ x \tag{11}$$

4 Energy Consumption Model

For a request r_h in the hot request set R_h, assume it accesses a popular file f_h in the popular file set F_h, which is allocated in the hot disk zone. The energy consumed by r_h can be written as below

$$e_h^{active} = \frac{s_h}{p^h} \tag{12}$$

The service time for r_h provided by a set of hot disks, where file f_h were allocated can be computed as follows

$$at_h^{active} = \frac{s_h}{t^h} \tag{13}$$

Thus, the energy consumption of the whole hot request set can be derived by

$$e_{R_h}^{active} = \sum_{h=1}^{|R_h|} e_h^{active} \tag{14}$$

Similarly, the total service time imposed by the whole hot request set R_h in the hot disk zone is

$$at_{R_h}^{active} = \sum_{h=1}^{|R_h|} at_h^{active} \tag{15}$$

In addition, we define rft_k as the finish time of request r_k. Then, we obtain the analytical formula for the energy consumed by the hot disks when they are idle:

$$e_{hot}^{idle} = i^h * (| D_h | * \max_{k=1}^{x} (rft_k) - at_{R_h}^{active}) \tag{16}$$

Hence, the total energy consumed by the hot disk zone can be computed by

$$e_{hot} = e_{R_h}^{active} + e_{hot}^{idle} = \sum_{h=1}^{|R_h|} e_h^{active} + i^h * (| D_h | * \max_{k=1}^{x} (rft_k) - at_{R_h}^{active}) \tag{17}$$

Similarly, the total energy consumed by the cold disk zone can be obtained by

$$e_{cold} = e_{R_c}^{active} + e_{cold}^{idle} = \sum_{c=1}^{|R_c|} e_c^{active} + i^l * (| D_c | * \max_{k=1}^{x} (rft_k) - at_{Rc}^{active}) \tag{18}$$

Therefore, the total energy consumption for the whole storage system is:

$$e_{total} = e_{hot} + e_{cold} \tag{19}$$

5 The SEA Strategy

In this section, we first present a detailed description of the SEA strategy. Then we briefly introduce the three baseline algorithms Greedy, SP, and HP.

Fig. 1 outlines SEA with some detailed explanations. Note that the input F has been sorted in an ascending order in terms of popularity before it is fed into SEA. In other words, file f_1 is the most popular file with the smallest file size, whereas file f_m is the most unpopular one with the largest file size. First, SEA uses the skew parameter θ to derive the number of popular files and the number of unpopular files in F based on Eq. 1 (Step 1). Second, Step 2 calculates γ, the ratio between the number of hot disks and the number of cold disks, based on Eq. 4, which in turn results in the number of hot disks HD and the number of cold disks CD. Consequently, HD of n disks are configured to their high speed modes and CD of n disks are set to their low speed modes (Step 4). Next, SEA assigns all popular files onto the hot disk zone in a striping manner (Step 5–Step 16). Similarly, all unpopular files are allocated onto the cold disk zone in a striping fashion (Step 17 – Step 28).

Input: A disk array D with n 2-speed disks, a collection of m files in the set F, and the skew parameter θ

Output: A file allocation scheme $X(m, k)$, where $k = \max_{i=1}^{m} \left(\frac{s_i}{sp} \right)$

```
1.  Use **Eq. 1** to compute the number of popular files and number of unpopular files in $F$
2.  Use **Eq. 4** to compute $\gamma$
3.  Hot disk number $HD = \frac{\gamma * n}{\gamma + 1}$, cold disk number $CD = n - HD$, $d_h$=1, $d_c$=1
4.  Configure $HD$ of $n$ disks to high speed mode and set $CD$ of $n$ disks to low speed mode
5.  for each popular file $f_p \in F_h$ do
6.      $p = 1$;
7.      for each stripe $sp_p$ of $f_p$ do
8.          $X(f_p, p) = d_h$
9.          $p = p + 1$
10.         if $d_h == HD$
11.             $d_h = 1$
12.         else
13.             $d_h = d_h + 1$
14.         end if
15.     end for
16. end for
17. for each unpopular file $f_u \in F_c$ do
18.     $u = 1$;
19.     for each stripe $sp_u$ of $f_u$ do
20.         $X(f_u, u) = d_c$
21.         $u = u + 1$
22.         if $d_c == CD$
23.             $d_c = 1$
24.         else
25.             $d_c = d_c + 1$
26.         end if
27.     end for
28. end for
```

Fig. 1. The SEA strategy

The average disk load ρ can be obtained by the following equation:

$$\rho = \frac{1}{n} \cdot \sum_{i=1}^{m} h_i \qquad (20)$$

Note that all the three existing algorithms assign nonparitioned files onto a disk array. In other words, each file must be allocated entirely onto one disk. In addition, since they only pursue minimized mean response times, all disks in the disk array are set to hot disks with high speed. The three algorithms are briefly described below.

(1) *Greedy*: It first calculates the mean load of all files and then assigns a consecutive set of files whose total load is equal to the mean load onto each disk.
(2) *SP (Sort Partition)*: It first computes the average disk utilization using Eq. 13. Next, it sorts all files into a list *I* in descending order of their service times. Finally, it allocates each disk d_j the next contiguous segment of *I* until its load $load_j$ reaches the maximum allowed threshold ρ. The remainder files (if any) after one round allocation will be assigned to the last disk d_n.
(3) *HP (Hybrid Partition)*: For each batch, HP assigns files to disks in distinct allocation intervals. It selects, for each allocation intervals l, a different disk d_k as the allocation target. It chooses the disk with the smallest accumulated load. A number of files are allocated to d_k until its load reaches the threshold T_k.

6 Performance Evaluation

6.1 Simulation Setup

We adopt the same strategy used in [16] to derive corresponding low speed mode disk statistics from parameters of a conventional Cheetah disk. The main characteristics of the 2-speed disk are shown in Table 1. The performance metrics by which we evaluate system performance include:

(1) *Mean response time:* Average response time of all file access request submitted to the simulated parallel disk storage system. Note that the mean response times are normalized in the scale [0, 1].
(2) *Energy consumption*: Energy (in Joules) consumed by the disk systems during the process of serving the entire request set.
(3) *Mean slowdown:* The ratio between average request turnaround time and average request service time.

Table 1 summarizes the configuration parameters of a simulated parallel disk array system used in our experiments and characteristics of the synthetic workload. All synthetic workload used were created by our trace generator.

Table 1. Characteristics of system parameters

Parameter	Value
Transfer rate in low mode	9.3 Mbytes/second
Idle power at low mode	2.17 Watts
Active energy at low mode (8-KB read)	43 mJoules
Transfer rate in high mode	31 Mbytes/second
Idle power at high mode	5.26 Watts
Active energy at high mode (8-KB read)	61 mJoules
Number of files	(5000)
Simulation duration	(1000) seconds
Aggregate access rate	(35) – (21~45) (1/second)
γ	3:13 ~ 10:6

6.2 Overall Performance Comparison

The goal of this experiment is to compare the proposed SEA0 and SEA5 algorithms against the three well-known file assignment schemes, and to understand the sensitivity of the five heuristics to the aggregate access rate in a parallel disk storage system, where an array of 2-speed disk drives serve requests simultaneously. The aggregate access rate varies from 21 (1/second) to 45 (1/second). The file sizes were generated according to a Zipf-like distribution with skew degree 70:30 and *file size base* is set to 1 Mbyte.

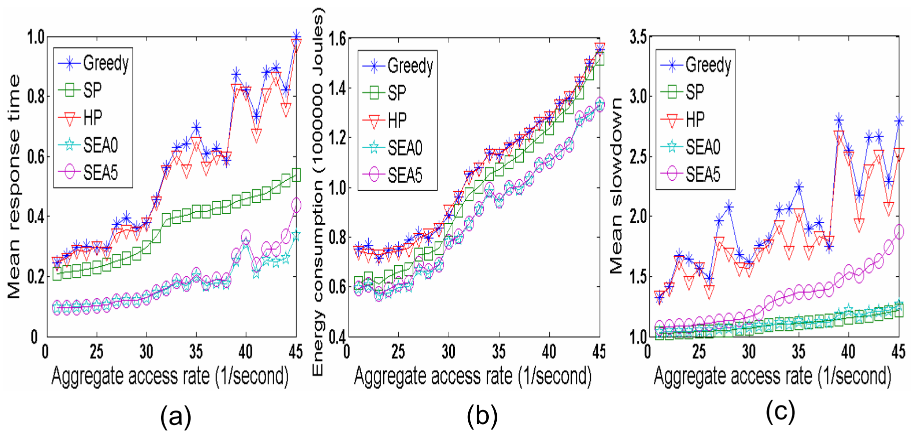

Fig. 2. Impact of aggregate access rate

Fig. 2 shows the simulation results for the five algorithms on a parallel disk array with 16 disk drives, where 5 of them are hot disks and 11 of them are cold disks. We observe from Fig. 2a that SEA0 and SEA5 consistently outperform the three exiting approaches in terms of mean response time. This is because they employ a striping-based data placement scheme, where intra-request parallelism is very high. Compared with the SP algorithm, SEA0 and SEA5 can reduce mean response time on average by 52.15% and 48.04%, while saving energy on average no less than 10.12% and 9.35%, respectively (see Fig. 2b). Although we only test a relatively light physical read workload (in the range [21, 45] 1/second), the actual system workload can be 10 times heavier (in the range [210, 450] 1/sccond) because of very low miss rates (less than 10%) provided by the high speed buffers on the data servers. The implication is that both SEA0 and SEA5 can be applied in applications where system workload is heavy. One example of such applications is OLTP (Online Transaction Processing).

7 Summary and Future Work

In this paper, we developed a new energy-saving strategy, called *s*triping-*b*ased *e*nergy-*a*ware (SEA), to generate optimized file allocation schemes. SEA0 and SEA5, two SEA-powered RAID-based data placement algorithms are implemented to evaluate the effectiveness and practicality of SEA. Comprehensive experimental results show that both SEA0 and SEA5 consistently improve the performance of parallel disk storage systems in terms of mean response time and save energy over three well-known data placement algorithms. Normally, there are two inherent drawbacks of current multi-speed disk based energy-saving approaches. First, disk speed mode transitions bring extra overhead in terms of transition time and transition energy [16], which is against their original goals. Second, frequent disk speed mode transitions are detrimental to the lifetime of hard disks [3]. SEA0 and SEA5 avoid these two shortcomings by statically configuring all disks to one of the multiple modes prior to serving requests. Furthermore, there is no speed mode transition during the process of serving the requests.

In summary, the SEA strategy realizes energy-saving not at the cost of performance degradation. Rather, it delivers much shorter mean response times compared with existing non-energy-aware data placement algorithms. Besides, it can provide fault-tolerance because of the RAID structures that it relies on. We will extend our scheme to a fully dynamic environment, where file access characteristics are not known in advance and may vary over time. As a result, a dynamic energy-saving data placement strategy is mandatory so that dynamically arrived files can be re-allocated by migrating files from one disk to another. File migration, however, incurs a relatively heavy overhead. How to make a good trade-off between migration cost and algorithm efficiency is a problem that needs to be solved.

Acknowledgments. This work was supported by the National Science Foundation under grant number CCF-0702781.

References

1. Akyürek, S., Salem, K.: Adaptive block rearrangement. ACM Trans. on Computer Systems 13(2), 89–121 (1995)
2. Breslau, L., Cao, P., Fan, L., Phillips, G., Shenker, S.: Web Caching and Zip-like Distributions: Evidence and Implications. In: Proc. IEEE INFOCOM, pp. 126–134 (1999)
3. Carrera, E.V., Pinheiro, E., Bianchini, R.: Conserving disk energy in network servers. In: Proc. 17th Supercomputing, pp. 86–97 (2003)
4. Colarelli, D., Grunwald, D.: Massive arrays of idle disks for storage archives. In: Proc. 16th Annual Int'l Conf. Supercomputing, pp. 1–11 (2002)
5. Dowdy, W., Foster, D.: Comparative Models of the File Assignment Problem. ACM Computing Surveys 14(2), 287–313 (1982)
6. Ghandeharizadeh, S., Kim, S.H., Shababi, C.: On disk scheduling and data placement for video servers. Sigmetrics Performance Evaluation 23(1), 37–46 (1995)
7. Graham, R.L.: Bounds on Multiprocessing Timing Anomalies. SIAM Journal Applied Math. 7(2), 416–429 (1969)
8. Gurumurthi, S., Sivasubramaniam, A., Kandemir, M., Franke, H.: DRPM: Dynamic speed control for power management in server class disks. In: Proc. Int'l Symp. Computer Architecture, pp. 169–179 (2003)
9. Hitachi Corp.: Hitachi Power & Acoustic Management – quietly cool. White paper (2004)

10. Huang, H., Hung, W., Shin, K.G.: FS2: dynamic data replication in free disk space for improving disk performance and energy consumption. In: Proc. 12th ACM SOSP, pp. 263–276 (2005)
11. Kwan, T., Mcgrath, R., Reed, D.: Ncsas World Wide Web Server Design and Performance. Computer 28(11), 67–74 (1995)
12. Lee, L.W., Scheuermann, P., Vingralek, R.: File assignment in parallel I/O systems with minimal variance of service time. IEEE Trans. Computers 49(2), 127–140 (2000)
13. Merialdo, P., Atzeni, P., Mecca, G.: Design and development of data-intensive web sites: The Araneus approach. ACM Trans. Internet Technology 3(1), 49–92 (2003)
14. Narris, M., Obal, J.: Performance Analysis of the Linux Buffer Cache While Running an Oracle OLTP Workload. Worcester Polytechnic Institute (2002)
15. Nishikawa, N., Hosokawa, T., Mori, Y., Yoshida, K., Tsuji, H.: Memory-based architecture for distributed WWW caching proxy. In: Proc. 7th Int'l Conf. World Wide Web, pp. 205–214 (1998)
16. Pinheiro, E., Bianchini, R.: Energy conservation techniques for disk array-based servers. In: Proc. 18th Supercomputing, pp. 68–78 (2004)
17. Power, heat, and sledgehammer (2002), http://www.max-t.com/downloads/whitepapers/SledgehammerPowerHeat20411.pdf
18. Son, S.W., Chen, G., Kandemir, M.: Disk layout optimization for reducing energy consumption. In: Proc. 19th Supercomputing, pp. 274–283 (2005)
19. Triantafillou, P., Christodoulakis, S., Georgiadis, C.: Optimal data placement on disks: a comprehensive solution for different technologies. IEEE Trans. Knowledge and Data Engineering 12(2), 324–330 (2000)