Tsinghua Science and Technology

Volume 25 | Issue 2 Article 3

2020

Lazy Scheduling Based Disk Energy Optimization Method

Yong Dong
the School of Computer Science, National University of Defense Technology, Changsha 410073, China.

Juan Chen
the School of Computer Science, National University of Defense Technology, Changsha 410073, China.

Yuhua Tang
the School of Computer Science, National University of Defense Technology, Changsha 410073, China.

Junjie Wu
the School of Computer Science, National University of Defense Technology, Changsha 410073, China.

Huiquan Wang
the National Innovation Institute of Defense Technology, Beijing 100091, China.

See next page for additional authors

Lazy Scheduling Based Disk Energy Optimization Method

Authors

Yong Dong, Juan Chen, Yuhua Tang, Junjie Wu, Huiquan Wang, and Enqiang Zhou

TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 04/12 pp203–216
DOI: 10.26599/TST.2018.9010140
Volume 25, Number 2, April 2020

Lazy Scheduling Based Disk Energy Optimization Method

Yong Dong, Juan Chen*, Yuhua Tang, Junjie Wu, Huiquan Wang, and Enqiang Zhou

Abstract: Reducing the energy consumption of the storage systems disk read/write requests plays an important role in improving the overall energy efficiency of high-performance computing systems. We propose a method to reduce disk energy consumption by delaying the dispatch of disk requests to the end of a time window, which we call time window-based lazy scheduling. We prove that sorting requests within a single time window can reduce the disk energy consumption, and we discuss the relationship between the size of the time window and the disk energy consumption, proving that the energy consumption is highly likely to decrease with increasing window size. To exploit this opportunity, we propose the Lazy Scheduling based Disk Energy Optimization (LSDEO) algorithm, which adopts a feedback method to periodically adjust the size of the time window, and minimizes the local disk energy consumption by sorting disk requests within each time window. We implement the LSDEO algorithm in an OS kernel and conduct both simulations and actual measurements on the algorithm, confirming that increasing the time window increases disk energy savings. When the average request arrival rate is 300 and the threshold of average request response time is 50 ms, LSDEO can yield disk energy savings of 21.5%.

Key words: high-performance computing system; storage systems; disk energy savings; lazy scheduling; disk seek; time window

1 Introduction

Storage systems[1] play a key role in High-Performance Computing (HPC) systems[2] and are becoming larger and larger due to increasing demands for storage performance and capacity. Increasing energy consumption by the storage system increases the running cost of the overall HPC system and reduces the storage system reliability. Thus, it is necessary to reduce the energy consumption of storage systems. As the major component of storage systems, disks consume most of the energy of these systems. Accordingly, in the present paper, we focus on how to reduce disk energy consumption.

Handling read/write requests consumes a large amount of the overall energy used by a disk. There is a great deal of research on disk energy optimization by means of the disk rotation speed technique[3] or disk-spin-down technique[4–10]. In fact, disk energy consumption is really closely related to disk seek. Furthermore, disk seek energy is directly influenced by the order of disk read/write accesses.

The positions of disk requests are usually discontinuous and distributed over different tracks. Discontinuous disk seek often results in frequent changes to the movement direction of the disk head, thereby increasing the overhead of disk seek and wasting disk energy. Sorting requests by on-disk track can shorten disk seek and reduce disk energy. In the present work, we sort the requests within the range of discrete time windows. The larger the time window

• Yong Dong, Juan Chen, Yuhua Tang, Junjie Wu, and Enqiang Zhou are with the School of Computer Science, National University of Defense Technology, Changsha 410073, China. E-mail: {yongdong, juanchen, yhtang, junjiewu, zqzhou}@nudt.edu.cn.
• Huiquan Wang is with the National Innovation Institute of Defense Technology, Beijing 100091, China. E-mail: whqamm@163.com.
∗ To whom correspondence should be addressed.
Manuscript received: 2018-09-21; accepted: 2018-11-14

of the arrival request queue, the more disk energy savings are achieved. However, increasing the time window size also increases the request response time and reduces performance. Our objective is to choose an appropriate time window to balance energy savings with disk performance. A primary focus of the present work is to study how disk energy savings change as the size of the time window increases.

In this paper, we propose the idea of time window based lazy scheduling to reduce disk energy consumption. The most important thing is to analyze the relationship between request order and disk energy. We prove that according to the location of disk head, sorting requests by on-disk track will minimize the disk energy consumption (Theorem 1). Most importantly, we prove that as the size of time window increases, the probability of decreasing the disk energy consumption is greater than the probability of increasing it (Theorem 2 and Deduction 1). According to the above theorems, we can conclude that increasing the size of the time window yields significant disk energy savings.

Based on the above theorems, we propose the Lazy Scheduling based Disk Energy Optimization (LSDEO) algorithm, consisting of two subalgorithms: a lazy scheduling algorithm for a single time window and a window size adjustment algorithm based on feedback. The first algorithm minimizes the local disk energy by sorting and dispatching requests within each time window. The second algorithm adopts a feedback method to periodically adjust the time window size while satisfying the demanded request response time. LSDEO is implemented as a module in the Linux kernel. Simulation experimental results and real experimental results both validate that enlarging the time window is beneficial for reducing disk energy consumption. Taking the average request arrival rate of 100 as an example, compared to the default scheduling policy, the disk seek energy savings of LSDEO increase from 44.45% to 54.69% as the time window size increases from 50 to 800 ms. We also measure the average response times for various time window sizes. When the average request arrival rate is 300 and the threshold of average request response time is 50 ms, LSDEO can yield 21.5% disk energy savings.

The contributions of this article are as follows:

(1) We propose the idea of time window based lazy scheduling and analyze the relationships between request order and disk energy and between time window size and disk energy consumption.

(2) We propose the LSDEO, which saves disk energy while meeting the demand for request response time, thereby balancing the disk energy-performance tradeoff.

(3) We implement LSDEO and evaluate its effectiveness by means of both simulation and experiment.

The rest of this paper is organized as follows. Section 2 describes the principle of lazy scheduling for energy savings. Section 3 describes the design and implementation of LSDEO and discusses its scalability. Section 4 validates the effectiveness of LSDEO by means of simulation and real experiments. Section 5 reviews the related work. Section 6 concludes the paper.

2 Principle of Lazy Scheduling

2.1 First example

There are two states for a disk: busy and idle. In the idle state, the disk is rotating normally without handling any request. In the busy state, the disk is handling requests, including positioning data and executing data read/write operations. In each read/write operation, the disk head first moves to the track to be accessed along the radius of the disk, and then rotates to the specified sector. Second, the disk head reads data from or writes data to the target location.

As indicated by the following formula, the total disk energy consumption (E_{total}) consists of three parts: the energy consumed by the movement of the disk head actuator arms (E_{seek}), the energy consumed by the rotation of the platter spindle motor (E_{rotation}), and the energy consumed by the data read/write (E_{rw}).

$$E_{\text{total}} = E_{\text{seek}} + E_{\text{rotation}} + E_{\text{rw}} \tag{1}$$

Here E_{rotation} includes the energy of disk rotation both during data locating ($E^{\text{pos}}_{\text{rotation}}$) and idling($E^{\text{idle}}_{\text{rotation}}$). Then we get

$$E_{\text{rotation}} = E^{\text{pos}}_{\text{rotation}} + E^{\text{idle}}_{\text{rotation}} \tag{2}$$

According to the states of the current disks, E_{total} can be divided into two parts, E_{busy} and E_{idle}, and then we have

$$E_{\text{busy}} = E_{\text{seek}} + E^{\text{pos}}_{\text{rotation}} + E_{\text{rw}} \tag{3}$$

$$E_{\text{idle}} = E^{\text{idle}}_{\text{rotation}} \tag{4}$$

$E^{\text{pos}}_{\text{rotation}}$ is closely related to the square of the disk's rotation-speed[3]. E_{rw} can be considered as a constant for a given data size. In this paper, we focus on saving disk energy by reducing E_{seek}.

The disk head inevitably moves back and forth along different tracks because the locations of many

requests usually belong to different tracks (continuous or discontinuous). Request re-scheduling by means of sorting requests reduces disk seek by eliminating unnecessary back and forth movements of the disk head, thereby saving energy.

Besides sorting requests by on-disk track, delaying the submit time of disk requests (dispatches) can allow scheduling of more disk requests over a longer duration. Delaying scheduling lengthens the request handling queue and reduces disk seek. One general idea to achieve this goal is to use a time window to delay request dispatches. The submission times of disk requests are delayed to the end of each time window. What we are concerned about is increasing the number of sorted requests by enlarging the time window as much as possible to achieve more energy savings; this strategy is called lazy scheduling. Figure 1 shows an example. t_{win} and T_{win} represent the sizes of two time windows. The arrival request queue is {5, 13, 12, 8, 22, 7, 15}. Because in each time window, requests are demanded to be sorted (shadowed regions), we get two kinds of disk seek traces as Figs. 1a and 1b, respectively. The respective disk distances are 8+5+4+10+7+8=42 and 3+4+1+9+7+8=32. In this comparison, for the same arrival request queue, using the larger time window decreases disk seek.

Can we increase the size of time window indefinitely to increase energy savings? The answer is no, because increasing the window size decreases the disk access performance. Rather, we should increase the size of time window as much as possible while maintaining a specified level of disk access performance. Actually, we can prove that the probability of disk energy decrease is larger than that of disk energy increase with increasing time window size. This conclusion implies that sometimes the energy savings are not so high when the time window is as big as possible.

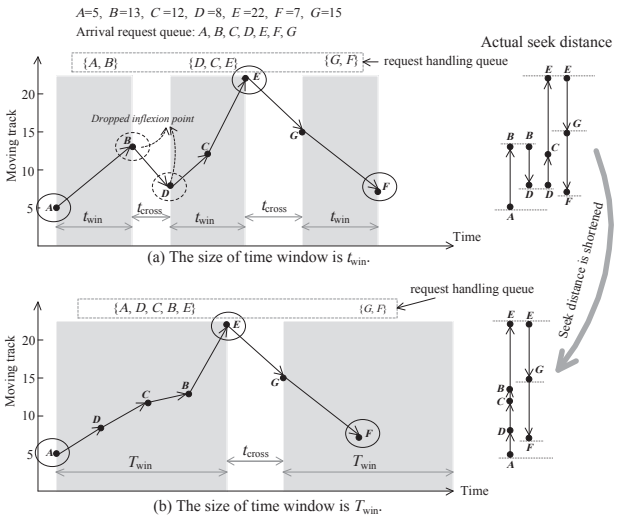

Fig. 1 An example that increasing the size of time window will shorten disk seek distance.

2.2 Relationship between request handling order and disk energy

We prove that sorting requests in ascending or descending order can minimize disk energy according to the formal statement of Theorem 1. Theorem 1 illustrates the relationship between the order of requests handled and the disk energy consumption. Let $\sigma = \langle x_1, ..., x_n \rangle$ represent a request queue containing n requests in a given order. In this paper, only two kinds of request queues are considered: arrival request queue and sorted request queue. $addr(x_i)$ and $size(x_i)$ represent the address and data size of request x_i, where $1 \leqslant addr(x_i) \leqslant R$. R denotes the maximum track address, $dist(\sigma)$ denotes the overall disk seek distance when all requests in request queue σ are accessed and $d(x_i, x_j)$ indicates disk head's moving distance from x_i to x_j, we have $d(x_i, x_j) = |addr(x_i) - addr(x_j)|$.

Theorem 1 For an arrival request queue $\sigma = \langle x_1, \ldots, x_n \rangle$, handling requests in the order of the corresponding sorted request queue $\hat{\sigma} = \langle \hat{x}_1, \ldots, \hat{x}_n \rangle$ minimizes disk energy consumption E_{busy}.

Proof According to Eq. (1), decreasing disk seek will reduce E_{seek} while leaving E_{rotation} and E_{rw} unchanged, thereby reducing E_{total}. Disk energy varies monotonically with changes in disk seek. To prove the theorem, we need to prove that $dist(\hat{\sigma})$ is the shortest. It is sufficient to prove that $\forall \sigma^*$, $dist(\hat{\sigma}) \leqslant dist(\sigma^*)$ because $\hat{\sigma}$ follows either ascending or descending order. Without loss of generality, assume that $\hat{\sigma}$ is sorted in ascending order, this gives $addr(\hat{x}_1) < addr(\hat{x}_2) < \cdots < addr(\hat{x}_n)$. The proof for the descending order case is similar.

(Proof by contradiction) Assume $\exists \sigma^{\Delta} (\sigma^{\Delta} \neq \hat{\sigma})$, $dist(\sigma^{\Delta}) < dist(\hat{\sigma})$. Because $\sigma^{\Delta} \neq \hat{\sigma}$, there are at least two requests occurring in descending order within σ^{Δ}. Assume these two requests sorted in the descending order are $\hat{x}_i$ and $\hat{x}_j$ $(1 \leqslant i < j \leqslant n)$ and we have $addr(\hat{x}_j) > addr(\hat{x}_i)$. So $\sigma^{\Delta} = \langle \hat{x}_1, \ldots, \hat{x}_j, \ldots, \hat{x}_i, \ldots, \hat{x}_n \rangle$. Except for $\hat{x}_j$ and $\hat{x}_i$, the order of the other requests in σ^{Δ} is the same as in $\hat{\sigma}$. According to the relationship between $\hat{x}_j$ and $\hat{x}_i$, we discuss the

following two cases.

(1) $i = j - 1$. $\hat{x}_j$ and $\hat{x}_i$ are adjacent. Here we use $\hat{x}_{i+1}$ to represent $\hat{x}_j$. Compare the disk seek distance for four continuous requests $\hat{x}_{i-1}$, $\hat{x}_{i+1}$, $\hat{x}_i$, and $\hat{x}_{i+2}$ in $\sigma^{\triangle}$ with that for four continuous requests $\hat{x}_{i-1}$, $\hat{x}_i$, $\hat{x}_{i+1}$, and $\hat{x}_{i+2}$ in $\hat{\sigma}$. The former is $d(\hat{x}_{i-1}, \hat{x}_{i+1}) + d(\hat{x}_{i+1}, \hat{x}_i) + d(\hat{x}_i, \hat{x}_{i+2})$ and the latter is $d(\hat{x}_{i-1}, \hat{x}_i) + d(\hat{x}_i, \hat{x}_{i+1}) + d(\hat{x}_{i+1}, \hat{x}_{i+2})$. We have $d(\hat{x}_{i+1}, \hat{x}_i) = d(\hat{x}_i, \hat{x}_{i+1})$, $d(\hat{x}_{i-1}, \hat{x}_{i+1}) > d(\hat{x}_{i-1}, \hat{x}_i)$, and $d(\hat{x}_i, \hat{x}_{i+2}) > d(\hat{x}_{i+1}, \hat{x}_{i+2})$. This gives $dist(\sigma^{\triangle}) > dist(\hat{\sigma})$, which contradicts $dist(\sigma^{\triangle}) < dist(\hat{\sigma})$. So the assumption must be false.

(2) $i < j - 1$. $\hat{x}_j$ and $\hat{x}_i$ are not adjacent. Compare disk seek distance of six requests $\hat{x}_{i-1}$, $\hat{x}_j$, $\hat{x}_{i+1}$, $\ldots$, $\hat{x}_{j-1}$, $\hat{x}_i$, $\hat{x}_{j+1}$ in $\sigma^{\triangle}$ and that of six requests $\hat{x}_{i-1}$, $\hat{x}_i$, $\hat{x}_{i+1}$, $\ldots$, $\hat{x}_{j-1}$, $\hat{x}_j$, $\hat{x}_{j+1}$ in $\hat{\sigma}$. The former is $d(\hat{x}_{i-1}, \hat{x}_j) + d(\hat{x}_j, \hat{x}_{i+1}) + d(\hat{x}_{i+1}, \hat{x}_{j-1}) + d(\hat{x}_{j-1}, \hat{x}_i) + d(\hat{x}_i, \hat{x}_{j+1})$ and the latter is $d(\hat{x}_{i-1}, \hat{x}_i) + d(\hat{x}_i, \hat{x}_{i+1}) + d(\hat{x}_{i+1}, \hat{x}_{j-1}) + d(\hat{x}_{j-1}, \hat{x}_j) + d(\hat{x}_j, \hat{x}_{j+1})$. Because $d(\hat{x}_{i-1}, \hat{x}_j) > d(\hat{x}_{i-1}, \hat{x}_i) + d(\hat{x}_i, \hat{x}_{i+1})$ and $d(\hat{x}_i, \hat{x}_{j+1}) > d(\hat{x}_{j-1}, \hat{x}_j) + d(\hat{x}_j, \hat{x}_{j+1})$, we get $dist(\sigma^{\triangle}) > dist(\hat{\sigma})$, which contradicts to $dist(\sigma^{\triangle}) < dist(\hat{\sigma})$. So the assumption must be false.

Therefore, $\forall \sigma^*$, $dist(\hat{\sigma}) \leqslant dist(\sigma^*)$. ■

2.3 Relationship between the time window size and global disk energy

In this subsection, we analyze the impact of the time window size upon E_{busy} when there are multiple time windows. The disk energy with multiple time windows (called the global disk energy, E_{multi}) is not equal to the sum of the disk energies for each time window (called local disk energy, E_{single}). Global disk energy also includes the energy of disk seek across every two adjacent time windows (E_{cross}). So the disk energy for multiple time windows is described by Eq. (5).

$$E_{\text{multi}} = \sum E_{\text{single}} + \sum E_{\text{cross}} \tag{5}$$

$\sum E_{\text{single}}$ decreases or is invariant with increase in the time window. $\sum E_{\text{cross}}$ does not have a determinate relationship with the size of the time window. With increasing time window size, $\sum E_{\text{cross}}$ may increase or decrease due to changes in the request sorting order, or changes of the disk head's movement direction across two adjacent time windows. Therefore, E_{multi} is not certain to decrease or remain invariant with an increase in the size of the time window.

Probabilistically, we prove that the probability of global disk energy decrease is larger than the probability of global disk energy increase with an increase in the size of the time window.

Suppose that the number of tracks is R and all tracks are signed as $1, 2, \ldots, R$. The total number of requests is N, and the number of requests in the i-th time window is n_i. The address width of the i-th time window is l_i, that is, $l_i = \max_{1 \leqslant j \leqslant n_i}\{\text{addr}(x_j)\} - \min_{1 \leqslant j \leqslant n_i}\{\text{addr}(x_j)\}$.

Hypothesis 1 The frequency of request arrival remains stable.

Hypothesis 2 The requests are uniformly distributed over the tracks.

Theorem 2 According to Hypothesis 1 and Hypothesis 2, given an arrival request queue, if the time window size increases in the proportion of λ, there exists a real number C ($C > 1$). If $\lambda \geqslant C$, the probability of disk seek distance decreasing is greater than probability of disk seek distance increasing.

Proof Assume that the average number of requests in each time window is n and that the address width of the time window is l, the probability under this case is denoted $P(l, n)$. According to Hypothesis 2, each request has the same probability of referring to any track. Assuming that the maximum track address is R, then n arrival requests produce R^n request distributions on R tracks. There are $(R - l)n(n - 1)(l + 1)^{n-2}$ distributions where l represents the address width of each time window, and we have

$$P(l, n) = \frac{(R - l)n(n - 1)(l + 1)^{n-2}}{R^n} \tag{6}$$

According to Hypothesis 1, let the average number of requests in each time window be n and the number of time windows be $\dfrac{N}{n}$. We define the disk seek distance of the time window with n requests as D_n, and we have

$$D_n = \sum_{i=1}^{\frac{N}{n}} (l_i + d_i) \tag{7}$$

where l_i represents the disk seek distance of the i-th time window and d_i represents the disk seek distance between the last request in the $(i - 1)$-th time window and the first request in the i-th time window.

Figure 2 shows the curve of $P(l, n)$ with different n. The address width of the time window varies from 0 to 10^4. For a given n, we have $\sum_l (P(l, n)) = 1$. Actually, when $n = 2$, $P(l, n)$ is a line of slope -1. When $n = 3$, $P(l, n)$ is a parabola. With increasing n, the maximum point of this parabola moves positively in the x direction.

Define $L_{\max} = \min_L P(l > L) > P(l < L)$, which means the probability of $l > L_{\max}$ is larger than the

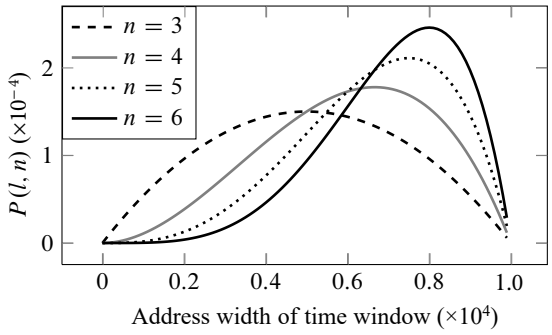

Fig. 2 ***P*(*l*, *n*) curves for various *n*.**

probability of $l < L_{\max}$ and $L_{\max}$ is the minimum among all cases. Increasing n increases $L_{\max}$, as shown in Fig. 2.

Let $D_{\lambda n}^{\max} = \sum_{i=1}^{\frac{N}{n}}(l_i + \frac{R}{2})$ and $D_n^{\min} = \sum_{i=1}^{\frac{N}{n}}(l_i)$, and we have the inequality (8).

$$D_n^{\min} \leqslant D_n \leqslant D_{\lambda n}^{\max} \tag{8}$$

Let the average address width of time window be L_n, $D_{\lambda n}^{\max}$ and $D_n^{\min}$ can be written as

$$D_n^{\min} = \frac{N}{n} L_n \tag{9}$$

$$D_{\lambda n}^{\max} = \frac{N}{\lambda n}\left(L_{\lambda n} + \frac{R}{2}\right) \tag{10}$$

According to the definition of $L_{\max}$, $D_{\lambda n}^{\max}$ can be rewritten as

$$D_{\lambda n}^{\max} = \frac{N}{\lambda n}\left(L_{\lambda n} + R - L_{\max}\right) \tag{11}$$

Define the symbol $\lessgtr$, $D_{\lambda n} \lessgtr D_n$ represents that the probability of $(D_{\lambda n} < D_n)$ is larger than the probability of $(D_{\lambda n} > D_n)$, i.e., $P(D_{\lambda n} < D_n) > P(D_{\lambda n} > D_n)$. The sufficient condition for establishing $D_{\lambda n} \lessgtr D_n$ is $D_{\lambda n}^{\max} \lessgtr D_n^{\min}$, which is equivalent to the inequality (12).

$$\frac{N}{\lambda n}(L_{\lambda n} + R - L_{\max}) \lessgtr \frac{N}{n} L_n \tag{12}$$

Let

$$C = \frac{2R}{L_{\max}} - 1 \tag{13}$$

When $\lambda \geqslant C$, we have

$$\frac{N}{\lambda n}(L_{\lambda n} + R - L_{\max}) \leqslant \frac{N}{n} L_{\max} \frac{L_{\lambda n} + R - L_{\max}}{2R - L_{\max}} \tag{14}$$

Because $\dfrac{L_{\lambda n} + R - L_{\max}}{2R - L_{\max}} < 1$, inequality (14) is rewritten into

$$\frac{N}{\lambda n}(L_{\lambda n} + R - L_{\max}) < \frac{N}{n} L_{\max} \tag{15}$$

According to the definition of $L_{\max}$, inequality (15) can deduce that inequality (12). That is, $D_{\lambda n} \lessgtr D_n$. ■

The condition of Theorem 2 is $\lambda \geqslant C$ where $C = \frac{2R}{L_{\max}} - 1$. Usually n is not too small, so we assume that $n \geqslant 3$. C is relative to n, as shown in Fig. 3. When $n = 3$, $C = 3$. When $n = 4$, $C = 2.2557\ldots$. When $n = 100$, $C = 1.034$. With further increase of n, C decreases and asymptotically approaches 1. The above analysis shows that C is not large in the actual case.

Deduction 1 According to Hypothesis 1 and Hypothesis 2, given an arrival request queue, assume that the number of requests in each time window is no less than three. If the time window size increases in the proportion λ ($\lambda \geqslant 3$), the probability of disk seek distance decrease is greater than the probability of disk seek distance increase.

Proof According to Theorem 2, we have Eqs. (9) and (10). Since $L_{\lambda n} < R$ and $\lambda \geqslant 3$, we get $\frac{1}{\lambda}(L_{\lambda n} + \frac{R}{2}) < \frac{R}{2}$. Since $\frac{R}{2} \lessgtr L_n$, we have $\frac{1}{\lambda}\left(L_{\lambda n} + \frac{R}{2}\right) \lessgtr L_n \Rightarrow \frac{N}{\lambda n}\left(L_{\lambda n} + \frac{R}{2}\right) \lessgtr \frac{N}{n} L_n \Rightarrow D_{\lambda n}^{\max} \lessgtr D_n^{\min} \Rightarrow D_{\lambda n} \lessgtr D_n$. ■

The statistical result shown in Fig. 4, including an amplification of a portion of the curve, also validates the above conclusion. Locally, the disk seek distance decreases in non-monotonically with increasing of time window size. But from the view point of the overall trend, the disk energy decreases with the increasing time window size.

3 LSDEO Algorithm

This section proposes an LSDEO algorithm, which consists of two sub-algorithms. One is a lazy scheduling algorithm for a single time window, shown as Steps 5 –

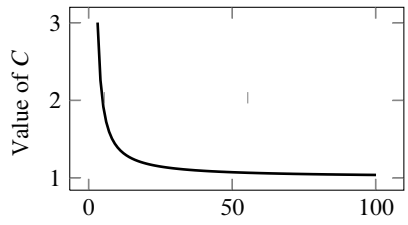

Fig. 3 ***C* versus *n*.**

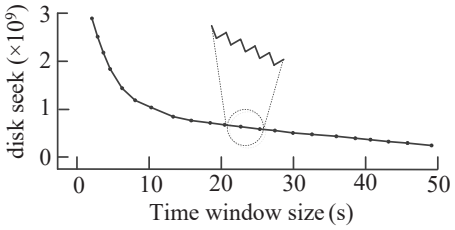

Fig. 4 Disk seek versus time window size.

8 in Algorithm 1, which achieves the minimum local disk energy consumption by sorting the disk requests within a single time window. The other is a feedback-based time window size adjustment algorithm, which periodically adjusts the size of the time window based on feedback.

By adjusting the time window size dynamically, we can achieve a good balance between disk energy and disk performance. Based on control theory, the feedback-based time window size adjustment algorithm adjusts the time window size dynamically according to the feedback control loop. The main idea of this algorithm is described as follows. First, we set an appropriate initial window size and determine the threshold of average request response time. Then we count the average request response time of each time window during the process of executing the lazy scheduling algorithm for a single time window. If the average request response time exceeds the threshold, we narrow the time window; otherwise, we enlarge it.

One typical feedback control loop consists of a monitor, a controller, and an adjustor. During the periodic execution process, the monitor records the average request response time, while the controller decides whether the window size should be adjusted and how to adjust it according to the average request response time and its threshold. Then the adjustor sets the proper time window size based on the results generated by the controller. Figure 5 illustrates the feedback control loop. We used the proportional controller in the present work.

Algorithm 1 LSDEO algorithm

Input: arrival request queue σ, current position of disk head h, time window size t_{win}, threshold of the average request response time $\tilde{t}_{\text{res}}$, ratio factor of controller K_p.

Output: request handling queue for multiple time windows.

1: Count the average request response time t_{res} for the current time window;
2: $diff = t_{\text{res}} - \tilde{t}_{\text{res}}$;
3: Calculate the adjustment range for the time window: $K_p \times diff$;
4: Calculate the size of the new time window: $t'_{\text{win}} = t_{\text{win}} - K_p \times diff$;
5: Start timing;
6: Buffer arrival requests until the end of the time window, delaying submission of requests;
7: When the end of the time window is reached, re-schedule all n disk requests in ascending order and generate a sorted request queue $\hat{\sigma}$;
8: Dispatch the request queue $\hat{\sigma}$ to the disk;
9: Test the finishing condition; if the condition is satisfied, goto Step 10); else, go to the next time window and goto Step 1);
10: End.

Assume that the relationship between the request response time t_{res} and the time window size t_{win} in the k-th work period of the feedback control loop satisfies $t_{\text{res}}(k) = \alpha' \times t_{\text{win}}(k) + \beta$ where α' and β are control parameters. In the $(k+1)$-th work period, we have $t_{\text{res}}(k+1) = t_{\text{res}}(k) + \alpha' \times diff(k)$, where $diff(k)$ represents the difference of the window size. The time-domain of the proportional controller is represented as $diff(k) = \widetilde{t}_{\text{res}} - K_p \times t_{\text{res}}(k)$ among which K_p is the ratio factor of proportional controller and $\widetilde{t}_{\text{res}}$ represents the threshold of the request response time. Thus $t_{\text{res}}(k+1) = \alpha' \times \widetilde{t}_{\text{res}} + (1 - K_p) t_{\text{res}}(k)$. We perform a Z-transform on both sides. According to the "Z-transform final value theorem", K_p satisfies $\alpha' \leqslant K_p < 2$. According to this analysis, increasing of time window will increase the request response time. Then we have $\alpha' > 0$. After determining K_p, we can obtain the time window size control method, shown as Steps 1–4 in Algorithm 1.

The block I/O request scheduler in the Linux kernel is in charge of issuing block I/O requests to the disk, and supports different block request scheduling algorithms. We implement LSDEO in the I/O schedulers in the Linux kernel, and call this implementation the lazy I/O scheduler.

As many users are concerned, the scheduling algorithm must ensure that the scheduling of different read and write requests is correct. Elevator scheduling framework and data access syntax ensure the correctness of LSDEO. In the Linux kernel, file system translates read/write operation into bio request for storage device, which is described by *struct bio*. The bio requests are submitted to the elevator scheduling framework after which they are merged and processed by the scheduling algorithm. If two requests have the same sector position, there are four different situations, described as follows: (1) Read after read: two requests are merged into one. (2) Read after write: the read

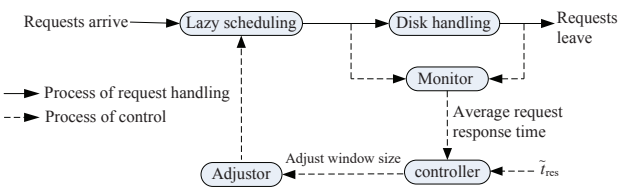

Fig. 5 Illustration of the feedback control loop.

requests get the latest data from page cache. (3) Write after write: the two requests are merged into one and the later is the correct one. (4) Write after read: the syntax of data access and file lock guarantee that the write operation can only be submitted after the read operation is accomplished.

4 Experimental Evaluation

This section evaluates the effectiveness of LSDEO. The evaluation is divided into two parts: simulation experiments based on Disksim simulator, and actual measurement of disk energy savings using a power meter. The simulator exactly models disk behavior and counts disk seek distance and disk energy consumption through its energy model. However, experimental measurements of disk energy through the power meter are ultimately necessary for evaluating the effectiveness of the algorithm, reflecting its benefit during actual use. Combining the simulation experiments and the actual measurements allows us to fully evaluate the effectiveness of LSDEO for disk energy optimization.

4.1 Simulation experiments

4.1.1 Simulation platform

We used the Disksim[11] simulator to evaluate the energy savings of the lazy scheduling. Disksim is an event-driven efficient disk simulator. Disksim collects various kinds of data during simulation of various disk operations, including disk seek and data access. Taking the disk request trace as input, Disksim simulates all kinds of disk running processes under various conditions. To evaluate the energy-savings of lazy scheduling, we improved Disksim as follows.

(1) Added disk energy simulation. The simulator can calculate energy consumption by multiplying power and time. Thus, the disk energy simulation is based on disk busy time and idle time. The simulator sets the disk power in the busy state and idle power in idle state, respectively.

(2) Implemented lazy scheduling. Disksim does not implement lazy scheduling. By handling the trace file and modifying the processing flow, we added lazy scheduling to Disksim.

The object disk of the Disksim simulation was the Seagate Cheetah15k5 SAS. Table 1 lists relevant parameters of this disk. By actual measurement using a power meter (Yokogawa WT1600)[12], we determined the disk power in the busy state and in the idle state.

Table 1 Parameters of Seagate Cheetah15k5 SAS.

Capacity	146 GB
Cylinder	72 170
Number of sectors	286 749 487
RPM	15 000
Max seek time	6.9 ms
Busy power	14 W
Idle power	9 W

To determine the execution effectiveness of lazy scheduling under various request conditions, we adopted two kinds of trace files. One was a synthetical trace, generated by a random number generator. The other was Cello99 trace[13], which is widely applied for research on disk systems.

4.1.2 Simulation experimental results

(A) Synthetic trace

For the synthetic trace, request arrival was shown to be Poisson distributed, and disk data access obeyed as uniform distribution. Each request had the same size of 4096 bytes in 8 data blocks. According to the average request arrival rate, we generated seven trace files.

Figure 6 shows the impact of variable time window size upon disk seek with lazy scheduling. For convenient comparison, the average disk seek is normalized. The basis line is the average disk seek under the default scheduling policy. Each curve represents different average request arrival rates.

The average disk seek of lazy scheduling is shorter than the default scheduling policy. With the increase of time, the average disk seek of lazy scheduling is shorter than that of the default scheduling policy. With increasing time window size, the average disk seek is decreased. For example, for the average request arrival rate of 20, the average disk seek is reduced by 22.2% when the time window size is increased from 20 to

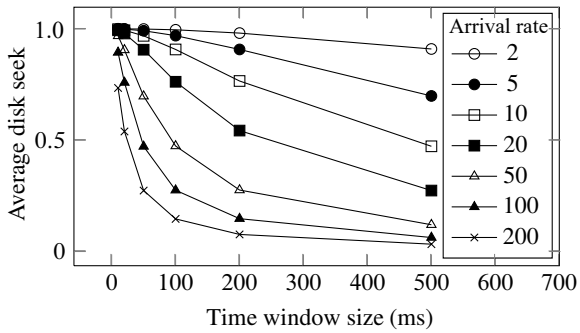

Fig. 6 Average disk seek versus time window size for various average request arrival rates.

100 ms. The reason is illustrated as follows. Given an average request arrival rate, using a larger time window results in more requests in each dispatch and thus reduced disk seeks. On the other hand, given a time window size, increasing the average request arrival rate leads to more arrival requests in single time window and thus reduced disk seeks.

Figure 7 demonstrates how reduced disk seek is beneficial in producing disk energy savings. With increasing average request arrival rate, the disk energy savings increase. When the average request arrival rate is equal to two, the proportion of disk energy savings is small, lower than 0.5%. That is because this time window includes only a few arrival requests, providing little opportunity to reduce disk seek and save disk energy. Furthermore, the energy savings for wider time windows were greater than those for narrower time windows. For example, for the average request arrival rate of 50, the disk energy savings increased from 13.7% to 23.8% when the time window size was changed from 50 ms to 100 ms.

Figure 8 shows the impact of the average request response time upon disk energy savings for various average request arrival rates and for the average request response time of 10, 50, and 100 ms. Increasing the time window lengthens the response time and worsens disk performance.

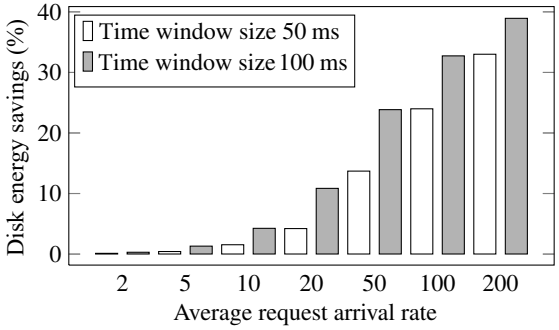

Fig. 7 Disk energy savings for various average request arrival rates.

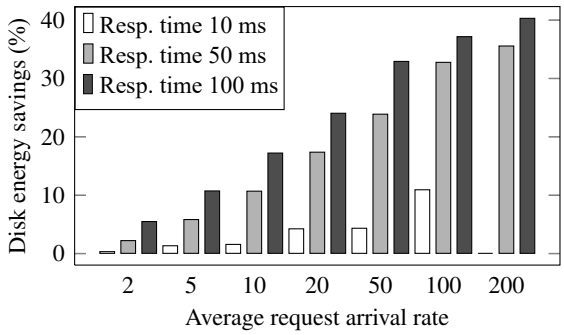

Fig. 8 Disk energy savings under various response time restrictions.

From this figure, under the same average request arrival rate, longer average response time would bring about greater disk energy savings. When the average request arrival rate is 200, the default policy cannot meet the demanded 10 ms response time. By relaxing the restriction on response time (i.e., allowing longer response time), we can obtain more disk energy savings. However, such disk energy savings would not continue increasing with further increases in the allowed response time, as can be seen from Fig. 8, the difference between the second and third columns is proportionally greater than that between the first and second columns.

(B) Cello99 trace

Next we report experimental results using the real Cello99 trace, a record of the disk access trace of a file server collected from January 14 to December 31 of 1999. This server used 22 SCSI disks. To evaluate the effectiveness of lazy scheduling, we chose one disk access record on January 31 as the simulation input. This simulation input consists of 24 trace files, and each trace file corresponds to one hour of the disk access record. So, 24 trace files correspond to one day of the disk access record. Each point in Fig. 9 represents the simulation results for one trace file.

Figure 9 compares the disk seek distances for two time window sizes, the horizontal axis represents time and the vertical axis represents the reduction in disk seeks of lazy scheduling compared to the default policy. Figure 10 shows the corresponding energy savings' proportions in the same cases.

Taking the 10 ms time window as an example, at 14:00, the disk energy savings reach the maximum

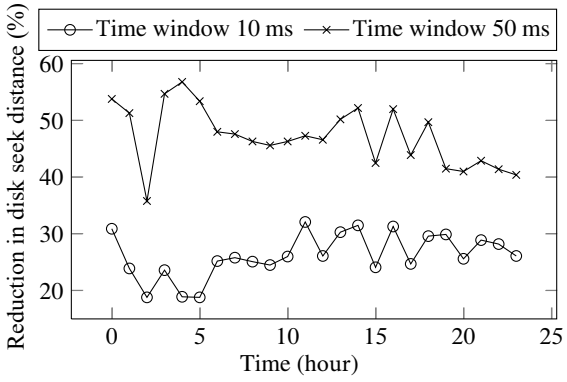

Fig. 9 Cello99 trace: Reductions in disk seek distance for two different time window sizes.

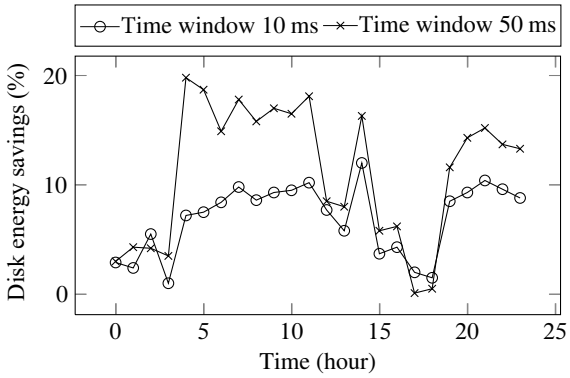

Fig. 10 Cello99 trace: Disk energy savings for two different time window sizes.

value, 12%. During the ranges 0:00 – 3:00 and 15:00 – 18:00, disk energy savings are less than 5%. This is because during these time ranges, the tracks accessed by the disk are relatively concentrated, the opportunity for reducing disk seek is relatively little, and the disk energy savings are thus relatively small. According to Theorem 3, the probability of decreasing disk seek is greater than 50% as the time window is increased, that is to say, disk seek decreases non-strictly. So, the probability of disk energy reduction is also more than 50%. The results shown in Fig. 10 illustrate this tendency. Furthermore, except for the variation of disk seek, other factors also produce disk energy increases with increasing time window size. For example, when there are many requests and the distribution of requests is concentrated in a small range, this increases the proportion of disk rotation time. Although enlarging the time window reduces disk seek, disk busy time increases because of this increased rotation time, increasing the overall disk energy.

Therefore, disk energy of lazy scheduling depends on the characteristics of request queue and the time window size. Given a fixed time window, if arrival requests are too few, arrival rate is too low. Besides, if arrival requests are too many and accessed tracks are too concentrated, the assumption that "*the requests are uniformly distributed on the tracks*" is not valid. The effectiveness of lazy scheduling in saving disk energy is influenced by the time window size. From the simulation results, as the time window increases, in most instances disk energy decreases and in rare cases increases.

4.2 Empirical experiments

(A) Measurement environment

We used a power meter (Yokogawa WT1600) to measure the disk energy savings achieved by implementing lazy scheduling. To accurately measure the disk energy consumption, we separated the system disk and the target disk and only measured the target disk. The target disk used an individual power supply. Table 2 lists the detailed measurement parameters.

Disk energy consumption is divided into circuit energy consumption and mechanical energy consumption. The circuit energy consumption comes from the read/write operation and the printed circuit board electronics. The mechanical energy comes from the arm actuator and the platter spindle motor. Lazy scheduling increases the time window size and reduces the disk seek distance, thereby reducing the arm actuator energy consumption. So, we are only concerned about the mechanical energy. There are two kinds of disk power supply inputs: 12 V and 5 V. The 12 V input provides the power for the mechanical components, and the 5 V input provides the power for the electronic components. Thus, we only need to connect the power meter to the 12 V input line to measure the mechanical energy. Except as indicated otherwise, subsequent references herein to disk energy savings all refer to energy savings from the mechanical components. The details of experimental platform are shown in Fig. 11.

In our real measurements, we used a synthetic trace meeting the following conditions: requests' arrival times obey the Possion distribution and disk data access obeys a uniform distribution. First, we measured disk energy consumption under the default scheduling policy, denoted by E_{def}. Then, we measured disk energy using lazy scheduling under various time window sizes,

Table 2 Measurement parameters.

Processor	Intel Xeon 5355 2.66 GHz
Memory	16 GB
Target disk	Seagate Barracuda ES.2 750 GB
Power meter	Yokogawa WT1600
Linux kernel	Linux-3.10
Power supply	Canlwen KYP-350ATX

Fig. 11 Experimental measurement environment.

denoted by E_{lazy}. Thus we were able to determine the disk energy savings realized by using lazy scheduling rather than the default scheduling policy. The period of time for each measurement was about four hours.

To measure the disk seek energy savings due to lazy scheduling, before measuring E_{def}, we let the disk idle for four hours and measured the disk's idle energy, denoted by E_{idle}, which comes from the Platter Spindle Motor. Disk seek energy by lazy scheduling is calculated as $E_{\text{lazy}} - E_{\text{idle}}$. Similarly, $E_{\text{def}} - E_{\text{idle}}$ is used to calculate the disk seek energy by the default scheduling. This allows us to determine the overall disk seek energy savings of lazy scheduling.

(B) Experimental results

Figure 12 graphs the disk energy savings due to lazy scheduling. The x axis represents the average request arrival rate and each group of columns corresponds to one request arrival rate. In Fig. 12, given an average request arrival rate, disk energy savings due to lazy scheduling tend to increase with the increasing time window size. Taking the request arrival rate of 100 as an example, when the time window size increased from 50 ms to 800 ms, the disk energy savings increased from 15.40% to 19.37%. This validates our theorem: enlarging the time window is beneficial for disk energy savings. Besides, we can see in some cases that increasing the time window decreases disk energy savings, which also accords with our theorem that each increase in the time window size is highly likely but not guaranteed to reduce the disk energy consumption.

Regarding variations in the request arrival rate, the higher the arrival rate, the more disk energy savings can be realized. Taking the 100 ms time window size as an example, when the request arrival rate increased from 50 to 300, the disk energy savings increased from 11.26% to 21.48%. This was because the higher request arrival rate meant that more of the disk seek distance could be avoided by means of request sorting and thus more disk energy savings could be achieved within each time window.

If we disregard the disk rotation energy of the platter spindle motor, we can measure disk seek energy savings as shown in Fig. 13.

Figure 14 graphs the impact upon the average response time of varying the average request arrival rate and time window size. With increasing average request arrival rate, the average response time increased. This was because higher average request arrival rate corresponds to more requests within each time window and longer request waiting times. Furthermore, the additional requests to be handled lead to longer disk processing time, thereby increasing the overall disk response time. Contrastingly, lower average request arrival rate corresponds to fewer requests within each time window, which leads to faster responses to requests. We also can see that in some cases, increasing

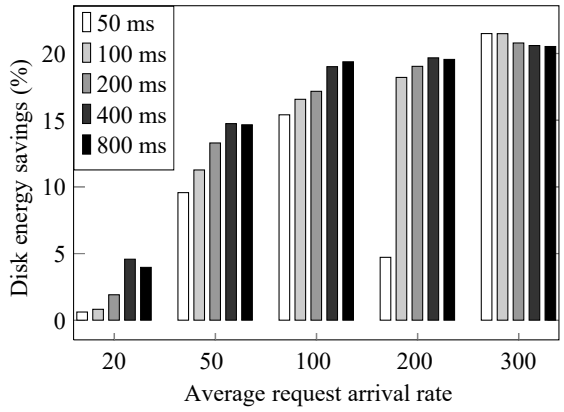

Fig. 12 Experimentally measured disk energy savings under lazy scheduling.

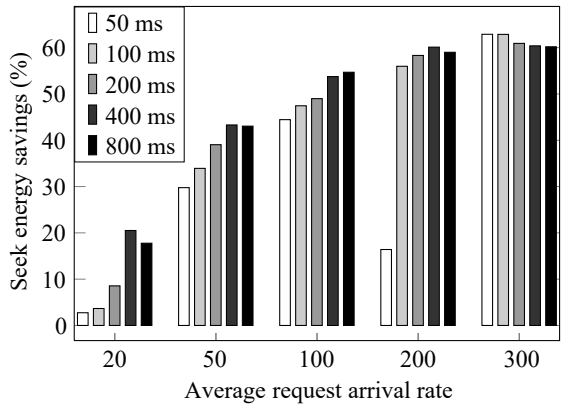

Fig. 13 Experimentally measured disk seek energy savings under lazy scheduling.

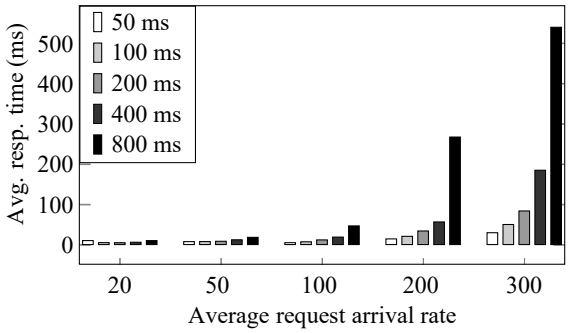

Fig. 14 Average response time versus average request arrival rate and time window size.

the time window can greatly affect the response time. For example, for the average request arrival rate of 300 and the time window of 800 ms, the response time greatly exceeds the response times in the other cases. This demonstrates the effectiveness of using the time window size adjustment method based on feedback.

Our LSDEO dynamically adjusts the time window size based on feedback. Figure 15 gives the disk energy saving results for dynamically adjusting the time window size, where the threshold of average request response time is 50 ms. According to Fig. 15, when the average request arrival rate is 300, disk energy savings were up to 21.5%.

5 Related Work

Recent research on optimizing storage devices' energy consumption can be classified into the following categories.

Building up disk power models. Effectively analyzing disk energy consumption and precisely building up its power model is the foundation of optimizing disk energy consumption. Hylick et al.[14] analyzed disk energy consumption carefully, and constructed disk energy model with exoteric parameters and performance results[15]. Greenawalt[16] simulated disk request access and disk energy using the Poisson distribution, and researched the relationship between disk energy consumption, reliability, and performance. Zedlewski et al.[17] designed the Dempsey simulator for portable disk devices, which combines disk performance and energy and exactly evaluates the energy cost of various processes such as disk seek and data read and write, and so on. Riska and Smirni[18] analyzed the savings of disk energy consumption within a certain decrease of disk performance. They also researched how to balance disk energy and performance[19].

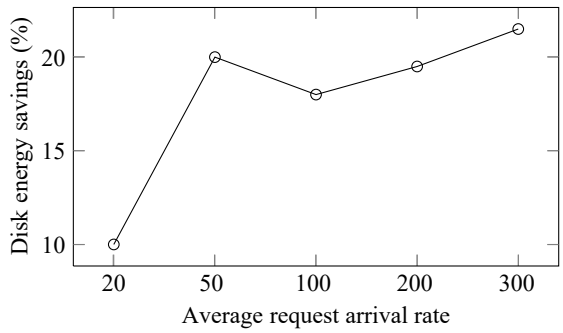

Fig. 15 LSDEO disk energy saving for 50 ms threshold of average request response time.

Caching to reduce storage device access. Generally, the data access of a program has the characteristics of data locality. By using a cache, we can delay data access to system storage, and write the modified data into the disk only once, which avoids unnecessary repeated data read/write operations and reduces the overall disk energy consumption. The cache we used here can be a kind of random access memory[20,21], buffer disk, SSD[22,23], or other storage devices. Song[24] introduced a storage allocation scheme to examine the effect of SSD cache upon disk bandwidth in a video server, and proposed an SSD bandwidth allocation algorithm to minimize energy consumption by allowing disks to run at lower speeds. Yin et al.[25] presented an SSD cache architecture that stores popular data on the SSD to allow disks to spin down when workloads are light, thereby reducing disk energy consumption. Kgil et al.[26,27] studied various types of cache architecture to reduce power consumption by the main memory and disk. Useche et al.[28] presented several schemes for an energy-aware SSD cache. These included data popularity prediction, I/O indirection, and reconfiguration schemes. Hui et al.[29] introduced a hybrid storage system that uses an SSD as a cache to allow under-utilized disks to be spun down.

Increasing disk idleness using compiler techniques. During program processing where there are no disk requests, the energy consumption of storage systems declines. Son et al.[30–32] fully exploited a program's I/O characteristics with a complier to guide data distribution on the disk. They also carried out code transformation to keep the disk idle as long as possible making disk in a lower-power state for a long time and thereby reducing the disk power consumption and average energy cost.

Dynamic Rotations Per Minute (DRPM) method. Gurumurthiy et al.[3] proposed and implemented a DRPM method to dynamically adjust the disk rotation speed and also presented a heuristic algorithm for set disk rotation speed. Carrera et al.[33] proposed a similar method. Bostoen et al.[34] examined how a multi-speed disk adapts DRPM to datacenter workloads and determined how to shape the workload to enable DRPM on conventional server disks.

Disk-spin-down technique. Disk-spin-down techniques reduce disk energy consumption by setting the spin down state for unused disks for a long time[4,9]. Wang et al.[35,36] utilized disk-spin-down to propose an energy saving method for RAID1 and RAID5. Cai

and Lu[37] combined random access memory and disk management to periodically adjust actual memory size and disk shut-down time for more energy savings. Weddle et al.[38] proposed the PARAID technique, which constructs a dynamically varying disk array structure based on variations in load.

File system power optimization. Sehgal et al.[39] evaluated the impact of the file system upon server performance and energy. Ruan et al.[40] used log files to cache small data, delay disk shut-down time, and reduce disk energy. Huang et al.[41] utilized a file copy technique to construct multi data counterparts and stored them in an idle device block. This scheduling allows users to access disk data in order as much as possible. Rush and Altiparmak[42] proposed an energy-efficient and performance-aware replica selection technique to reduce the energy consumption of data storage systems without negatively affecting their performance.

Disk block grouping technique. Essary and Amer[43] proposed an on-line data block predict engine that implemented data block copy to reduce disk operations for energy savings by means of dynamic grouping. Liao et al.[44] defined the concept of the Immediate Successor Relationship Amount (ISRA) to describe the successor relationships of data blocks. They proposed an ISRA-based grouping algorithm that reduced disk seek by grouping data blocks having frequent successive accesses and sorting these data blocks, with the main theory being to reduce disk seek and redistribute data from the angle of space. This method depends on data access repetition; more access repetition makes the method more effective. But for data centers, the multiple users and different application types bring about highly random data access, showing the limitations of Liaos algorithm. The presently proposed algorithm does not require highly repetitive access to be effective.

Data prefetching for energy reduction. Song et al.[45] examined how seek operations affect disk energy consumption, and analyzed the relationship between the amount of prefetched data and the number of seeks. They proposed a data prefetching scheme to dynamically adjust the amount of data prefetched in response to the bitrates of streams and the power characteristics of different disks in order to minimize overall disk energy consumption.

Data request scheduling policy. Chou et al.[46] analyzed several versions of the energy-aware disk scheduling problem based on assumptions regarding the arrival pattern of the requests. They concluded that the corresponding optimization problems are NP-complete by means of reduction to the set cover or the independent set problem. As a result, they proposed optimal and heuristic scheduling algorithms to maximize the energy savings of a storage system.

Besides the techniques mentioned above all, Bostoen et al.[47] surveyed power reduction techniques for data-center storage systems. They classified existing power-reduction techniques according to the disk-power factor and the storage-stack layer addressed. What's more, every energy-conservation technique should make trade-offs between power, capacity, performance, and dependability. In this research, we make trade-off between energy and request scheduling.

6 Conclusion

This article proposes a concept of time window based lazy scheduling to save disk energy. We analyzed two relationships: that between request sorting and the disk energy, and that between the time window size and the global disk energy. One of the most important contributions of the present work is that we prove that increasing the time window size is highly likely to decrease the global disk energy. We proposed and implemented an LSDEO algorithm. Finally, we evaluated the effectiveness of our algorithm by means of experimental measurements. The next step of this research is integrating LSDEO into RAID-based storage device to achive disk energy savings.

Acknowledgment

This work was supported by the National Key Research and Development Program of China (No. 2017YFB0202201).

References

[1] W. X. Xu, Y. T. Lu, Q. Li, E. Q. Zhou, Z. L. Song, Y. Dong, W. Zhang, D. P. Wei, X. M. Zhang, H. T. Chen, et al., Hybrid hierarchy storage system in MilkyWay-2 supercomputer, *Front. Comput. Sci.*, vol. 8, no. 3, pp. 367–377, 2014.

[2] X. K. Liao, L. Q. Xiao, C. Q. Yang, and Y. T. Lu, MilkyWay-2 supercomputer: System and application, *Front. Comput. Sci.*, vol. 8, no. 3, pp. 345–356, 2014.

[3] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Franke, DRPM: Dynamic speed control for power management in server class disks, in *Proc. 30th Annu. Int. Symp. Computer Architecture*, San Diego, CA, USA, 2003, pp. 169–181.

[4] F. Douglis, P. Krishnan, and B. N. Bershad, Adaptive disk spin-down policies for mobile computers, in *Proc. 2nd*

Symp. Mobile and Location- Independent Computing, Berkeley, CA, USA, 1995, pp. 121–137.

[5] K. Li, R. Kumpf, P. Horton, and T. E. Anderson, Quantitative analysis of disk drive power management in portable computers, in *Proc. USENIX Winter 1994 Technical Conf. USENIX Winter 1994 Technical Conf.*, San Francisco, CA, USA, 1994.

[6] Y. H. Lu, E. Y. Chung, T. Simunic, T. Benini, and G. de Micheli, Quantitative comparison of power management algorithms, in *Proc. Design, Automation and Test in Europe Conf. Exhibition 2000*, Paris, France, 2000, pp. 20–26.

[7] E. Otoo, D. Rotem, and S. C. Tsao, Dynamic data reorganization for energy savings in disk storage systems, in *Proc. 22nd Int. Conf. Scientific and Statistical Database Management*, Heidelberg, Germany, 2010, pp. 322–341.

[8] A. Verma, R. Koller, L. Useche, and R. Rangaswami, SRCMap: Energy proportional storage using dynamic consolidation, in *Proc. 8th USENIX Conf. File and Storage Technologies*, San Jose, CA, USA, 2010, p. 20.

[9] D. Essary and A. Amer, Predictive data grouping: Defining the bounds of energy and latency reduction through predictive data grouping and replication, *ACM Trans. Storage*, vol. 4, no. 1, p. 2, 2008.

[10] N. Nishikawa, M. Nakano, and M. Kitsuregawa, Low power management of OLTP applications considering disk drive power saving function, in *Proc. 21st Int. Conf. Database and Expert Systems Applications*, Bilbao, Spain, 2010, pp. 241–250.

[11] J. S. Bucy, J. Schindler, S. W. Schlosser, and G. R. Ganger, The DiskSim simulation environment version 4.0 reference manual, Tech Rep. CMU-PDL-08-101, http://www.pdl.cmu.edu / PDL-FTP / DriveChar / CMU-PDL-08-101.pdf, 2018.

[12] Yokogawa Test & Measurement Corporation, WT1600 digital power meter user's manual, https://cdn.tmi.yokogawa.com/IM760101-01E.pdf, 2018.

[13] Hewlett-Packard Laboratories, cello99 trace, http://iotta.snia.org/historical_section?tracetype_id=3, 2018.

[14] A. Hylick, R. Sohan, A. Rice, and B. Jones, An analysis of hard drive energy consumption, in *Proc. 2008 IEEE Int. Symp. Modeling, Analysis and Simulation of Computers and Telecommunication Systems*, Baltimore, MD, USA, 2008, pp. 103–112.

[15] A. Hylick and R. Sohan, A methodology for generating disk drive energy models using performance data, in *Proc. 22nd ACM Symp. Operating Systems Principles (SOSP) Workshop on Power Aware Computing and Systems*, Big Sky, MT, USA, 2009, pp. 1–5.

[16] P. M. Greenawalt, Modeling power management for hard disks, in *Proc. 2nd Int. Workshop on Modeling, Analysis, and Simulation on Computer and Telecommunication Systems*, Durham, NC, USA, 1994, pp. 62–66.

[17] J. Zedlewski, S. Sobti, N. Garg, F. Z. Zheng, A. Krishnamurthy, and R. Wang, Modeling hard-disk power consumption, in *Proc. 2nd USENIX Conf. File and Storage Technologies*, San Francisco, CA, USA, 2003, pp. 217–230.

[18] A. Riska and E. Smirni, Autonomic exploration of trade-offs between power and performance in disk drives, in *Proc. 7th Int. Conf. Autonomic Computing*, Washington, DC, USA, 2010, pp. 131–140.

[19] A. Riska, N. F. Mi, E. Smirni, and G. Casale, Feasibility regions: Exploiting tradeoffs between power and performance in disk drives, *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 37, no. 3, pp. 43–48, 2009.

[20] F. Chen and X. D. Zhang, Caching for bursts (C-Burst): Let hard disks sleep well and work energetically, in *Proc. 13th Int. Symp. Low Power Electronics and Design*, Bangalore, India, 2008, pp. 141–146.

[21] A. C. Manzanares, Energy efficient pre-fetching-models to implementation, PhD dissertation, Auburn University, Auburn, AL, USA, 2010.

[22] M. Nijim, A. Manzanares, X. J. Ruan, and X. Qin, HYBUD: An energy-efficient architecture for hybrid parallel disk systems, in *Proc. 18th Int. Conf. Computer Communications and Networks*, San Francisco, CA, USA, 2009, pp. 1–6.

[23] H. J. Lee, K. H. Lee, and S. H. Noh, Augmenting RAID with an SSD for energy relief, in *Proc. 2008 Conf. Power Aware Computing and Systems*, San Diego, CA, USA, 2008, p. 12.

[24] M. Song, Minimizing power consumption in video servers by the combined use of solid-state disks and multi-speed disks, *IEEE Access*, vol. 6, pp. 25737–25746, 2018.

[25] S. Yin, X. W. Li, K. L. Li, J. Z. Huang, X. J. Ruan, X. M. Zhu, W. Cao, and X. Qin, REED: A reliable energy-efficient RAID, in *Proc. 4th IEEE Int. Conf. Parallel Processing*, Beijing, China, 2015, pp. 649–658.

[26] T. Kgil and T. Mudge, FlashCache: A NAND flash memory file cache for low power Web servers, in *Proc. 2006 Int. Conf. Compilers, Architecture and Synthesis for Embedded Systems*, Seoul, Korea, 2006, pp. 103–112.

[27] T. Kgil, D. Roberts, and T. Mudge, Improving NAND flash based disk caches, in *Proc. IEEE Int. Symp. Computer Architecture*, Beijing, China, 2008, pp. 186–190.

[28] L. Useche, J. Guerra, M. Bhadkamkar, M. Alarcon, and R. Rangaswami, EXCES: External caching in energy saving storage systems, in *Proc. 14th Int. Symp. High Performance Computer Architecture*, Salt Lake City, UT, USA, 2008, pp. 89–100.

[29] J. Hui, X. Z. Ge, X. X. Huang, Y. Liu, and Q. J. Ran, E-HASH: An energy-efficient hybrid storage system composed of one SSD and multiple HDDs, in *Proc. 3rd Int. Conf. Swarm Intelligence*, Shenzhen, China, 2012, pp. 527–534.

[30] S. W. Son, G. Chen, and M. Kandemir, Disk layout optimization for reducing energy consumption, in *Proc. 19th Annu. Int. Conf. Supercomputing*, Cambridge, MA, USA, 2005, pp. 274–283.

[31] S. W. Son, M. Kandemir, and A. Choudhary, Software-directed disk power management for scientific applications, in *Proc. 19th IEEE Int. Parallel and Distributed Processing Symp.*, Denver, CO, USA, 2005, pp. 4–8.

[32] M. Kandemir, S. W. Son, and M. Karakoy, Improving disk reuse for reducing power consumption, in *Proc. 2007 Int. Symp. Low Power Electronics and Design*, Portland, OR, USA, 2007, pp. 129–134.

[33] E. V. Carrera, E. Pinheiro, and R. Bianchini, Conserving disk energy in network servers, in *Proc. 17th Annu. Int. Conf. Supercomputing*, San Francisco, CA, USA, 2003, pp. 86–97.
[34] T. Bostoen, S. Mullender, and Y. Berbers, Analysis of disk power management for data-center storage systems, in *Proc. 3rd Int. Conf. Future Energy Systems: Where Energy, Computing and Communication Meet*, Madrid, Spain, 2012, pp. 1–10.
[35] J. Wang, H. J. Zhu, and D. Li, eRAID: Conserving energy in conventional disk-based RAID system, *IEEE Trans. Comput.*, vol. 57, no. 3, pp. 359–374, 2008.
[36] D. Li, H. L. Cai, X. Y. Yao, and J. Wang, Exploiting redundancy to construct energy-efficient, high-performance RAIDs, Tech. Rep. TR-05-07-04, Computer Science and Engineering Department, University of Nebraska Lincoln, Lincoln, NE, USA, 2005.
[37] L. Cai and Y. H. Lu, Joint power management of memory and disk, in *Proc. Conf. Design, Automation and Test in Europe*, Munich, Germany, 2005, pp. 86–91.
[38] C. Weddle, M. Oldham, J. Qian, A. I. A. Wang, P. Reiher, and G. Kuenning, PARAID: A gear-shifting power-aware RAID, in *Proc. 5th USENIX Conf. File and Storage Technologies*, San Jose, CA, USA, 2007, pp. 245–260.
[39] P. Sehgal, V. Tarasov, and E. Zadok, Evaluating performance and energy in file system server workloads, in *Proc. 8th USENIX Conf. File and Storage Technologies*, San Jose, CA, USA, 2010, pp. 253–266.
[40] X. J. Ruan, A. Manzanares, S. Yin, Z. L. Zong, and X. Qin, Performance evaluation of energy-efficient parallel I/O systems with write buffer disks, in *Proc. 2009 Int. Conf. Parallel Processing*, Vienna, Austria, 2009, pp. 164–171.
[41] H. Huang, W. D. Hung, and K. G. Shin, FS2: Dynamic data replication in free disk space for improving disk performance and energy consumption, in *Proc. 20th ACM Symp. Operating Systems Principles*, Brighton, UK, 2005, pp. 263–276.
[42] E. N. Rush and N. Altiparmak, Exploiting replication for energy efficiency of heterogeneous storage systems, in *Proc. IEEE 24th Int. Symp. Modeling Analysis and Simulation on Computer and Telecommunication Systems*, London, UK, 2016, pp. 79–84.
[43] D. Essary and A. Amer, Sustainable predictive storage management: On-line grouping for energy and latency reduction, in *Proc. 4th Annu. Int. Conf. Systems and Storage*, Haifa, Israel, 2011, pp. 1–9.
[44] X. L. Liao, S. Bai, Y. P. Wang, and S. M. Hu, ISRA-based grouping: A disk reorganization approach for disk energy conservation and disk performance enhancement, *IEEE Trans. Comput.*. vol. 60, no. 2, pp. 292–304, 2011.
[45] M. Song, Y. Lee, and E. Kim, Data prefetching to reduce energy use by heterogeneous disk arrays in video servers, in *Proc. 23rd ACM Workshop on Network and Operating System Support for Digital Audio and Video*, Oslo, Norway, 2013, pp. 1–6.
[46] J. Chou, J. Kim, and D. Rotem, Energy-aware scheduling in disk storage systems, in *Proc. Int. Conf. Distributed Computing Systems*, Minneapolis, MN, USA, 2011, pp. 423–433.
[47] T. Bostoen, S. Mullender, and Y. Berbers, Power reduction techniques for data-center storage systems, *ACM Comput. Surv.*, vol. 45, no. 3, p. 33, 2013.

Yong Dong received the PhD degree from National University of Defense Technology, China in 2012. He is now an associate professor at National University of Defense Technology, China. His research interests focus on supercomputer systems and large scale parallel storage systems.

Juan Chen received the PhD degree from National University of Defense Technology, China in 2007. She is now an associate professor at National University of Defense Technology, China. Her research interests focus on supercomputer systems and energy-aware interconnection network design.

Yuhua Tang received the MS degree from National University of Defense Technology, China in 1986. She is now a professor at National University of Defense Technology, China. Her research interests focus on computer architecture and parallel computing.

Junjie Wu received the PhD degree from National University of Defense Technology, China in 2009. He is now an associate professor at National University of Defense Technology, China. His research interests focus on photonic quantum computing and quantum algorithm.

Huiquan Wang received the PhD degree from National University of Defense Technology, China in 2016. He is now an assistant professor at National Innovation Institute of Defense Technology, China. His research interests focus on computer architecture and artificial intelligence.

Enqiang Zhou received the MS degree from National University of Defense Technology, China in 1998. He is now a professor at National University of Defense Technology, China. His research interests focus on supercomputer and large scale parallel storage system.