PRE-BUD: Prefetching for Energy-Efficient Parallel I/O Systems with Buffer Disks

ADAM MANZANARES, XIAO QIN, XIAOJUN RUAN, and SHU YIN, Auburn University

A critical problem with parallel I/O systems is the fact that disks consume a significant amount of energy. To design economically attractive and environmentally friendly parallel I/O systems, we propose an energy-aware prefetching strategy (PRE-BUD) for parallel I/O systems with disk buffers. We introduce a new architecture that provides significant energy savings for parallel I/O systems using buffer disks while maintaining high performance. There are two buffer disk configurations: (1) adding an extra buffer disk to accommodate prefetched data, and (2) utilizing an existing disk as the buffer disk. PRE-BUD is not only able to reduce the number of power-state transitions, but also to increase the length and number of standby periods. As such, PRE-BUD conserves energy by keeping data disks in the standby state for increased periods of time. Compared with the first prefetching configuration, the second configuration lowers the capacity of the parallel disk system. However, the second configuration is more cost-effective and energy-efficient than the first one. Finally, we quantitatively compare PRE-BUD with both disk configurations against three existing strategies. Empirical results show that PRE-BUD is able to reduce energy dissipation in parallel disk systems by up to 50 percent when compared against a non-energy aware approach. Similarly, our strategy is capable of conserving up to 30 percent energy when compared to the dynamic power management technique.

Categories and Subject Descriptors: B.4.3 [**Input/Output and Data Communications**]: Interconnections—*Parallel I/O*; D.4.2 [**Operating Systems**]: Storage Management—*Secondary storage*

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Prefetching, energy conservation; buffer disks

ACM Reference Format:
Manzanares, A., Qin, X., Ruan, X., and Yin, S. 2011. PRE-BUD: Prefetching for energy-efficient parallel I/O systems with buffer disks. ACM Trans. Storage 7, 1, Article 3 (June 2011), 29 pages.
DOI = 10.1145/1970343.1970346 http://doi.acm.org/10.1145/1970343.1970346

1. INTRODUCTION

The number of large-scale parallel I/O systems is increasing in today's high-performance data-intensive computing systems due to the storage space required to contain the massive amount of data. Typical examples of data-intensive applications requiring large-scale parallel I/O systems include: long-running simulations [Eom and Hollingsworth 2000], remote sensing applications [Trizna 2005] and biological sequence analysis [Hawkins and Bodn 2005]. As the size of a parallel I/O system grows, the energy consumed by the I/O system often becomes a large part of the total cost of ownership [Pinheiro and Bianchini 2004; Wang et al. 2008; Xie 2008]. Reducing the energy costs of operating these large-scale disk I/O systems often becomes one

This work was supported by the US National Science Foundation under Grants CCF-0742187, CNS-0757778, CNS-0831502, OCI-0753305, and DUE-0621307, and by Auburn University under a startup grant.
Authors' addresses: A. Manzanares, X. Qin, X. Ruan, and S. Yin, Auburn University, Department of Computer Science and Software Engineering, Shelby Center for Engineering Technology, Suite 3101, Auburn, AL 36849-5347, Corresponding author's email: xqin@auburn.edu.

DOI 10.1145/1970343.1970346 http://doi.acm.org/10.1145/1970343.1970346

of the most important design issues. It is known that disk systems can account for nearly 27% of the total energy consumption in a data center [Maximum Throughput, Inc., 2002]. Even worse, the push for disk I/O systems to have larger capacities and speedier response times have driven energy consumption rates upward.

Reducing energy consumption of computing platforms has become an increasingly hot research field. Green computing has recently been targeted by government agencies; efficiency requirements have been outlined in Jones [2006]. Large-scale parallel disks inevitably lead to high energy requirements of data-intensive computing systems due to scaling issues. Data centers typically consume anywhere between 75 W/ft^2 to 200 W/ft^2 and this may increase to 200–300 W/ft^2 in the near future [Moore 2002; Zong et al. 2007]. These large-scale computing systems not only have a large economical impact on companies and research institutes, but also produce a negative environmental impact. Data from the US Environmental Protection Agency indicates that generating 1 kWh of electricity in the United States results in an average of 1.55 pounds (lb) of carbon dioxide (CO2) emissions. With large-scale clusters requiring up to 40TWh of energy per year at a cost of over $4B it is easy to conclude that energy-efficient clusters can have huge economical and environmental impacts [Carrera et al. 2003].

Several techniques proposed to conserve energy in disk systems include dynamic power management schemes [Douglis et al. 1994; Li et al. 1994], power-aware cache management strategies [Zhu et al. 2004a], software-directed power management techniques [Son and Kandemir 2006], redundancy techniques [Papathanasiou and Scott 2004], data placement [Pinheiro and Bianchini 2004; Xie 2008], and multi-speed settings [Gurumurthi et al. 2003; Helmbold et al. 1998; Krishnan et al. 1995]. However, the research on energy-efficient prefetching for parallel I/O systems with buffer disks is still in its infancy. Therefore, it is imperative to develop new prefetching techniques to reduce energy consumption in buffer-disk-based parallel I/O systems while maintaining high performance.

Energy dissipation in parallel disks can be reduced by traditional power management strategies that turn idle disks into low-power modes or by directly shutting down idle disks. The traditional power management schemes can suffer great time and energy overheads that are induced by waking a disk up many times. Moreover, the existing power management strategies can shorten the lifecycle of disks if they are spun up and down frequently, thereby degrading the availability and reliability of the disk system. To remedy these two deficiencies, we proposed a novel parallel I/O architecture with buffer disks (see Zong et al. [2007] for the details of the disk architecture) to reduce the number of power-state transitions of disks and decrease the energy consumption of the disk system. Using buffer disks to temporally buffer the requests for data disks, one can keep data disks in the low-power state (e.g., standby mode) as long as possible. To fully utilize buffer disks while aggressively putting data disks into the low-power state, we design in this study an energy-aware prefetching strategy (PRE-BUD for short).

There are two buffer disk configurations for PRE-BUD. The first configuration adds an extra disk performing as a buffer disk, whereas the second configuration uses an existing disk in the I/O system as a buffer disk. The design of these two disk configurations relies on the fact that in a wide variety of data-intensive computing applications (e.g., web applications) a small percentage of the data is frequently accessed [Kwan et al. 1995]. The goal of this research is to move this small amount of frequently accessed data from data disks into buffer disks, thereby allowing data disks to switch into a low-power state for an increased period of time.

PRE-BUD has the goal of dynamically fetching data sets with the highest energy-savings into buffer disks. To accurately prefetch data blocks, information concerning future disk requests is indispensable. PRE-BUD can deal with both offline and online

situations. In the offline case, PRE-BUD is provided with a priori knowledge of the list of disk requests. In the online case, PRE-BUD employs the look-ahead technique [Kallahalla and Varman 2002] that can furnish a window of future disk requests.

This research offers the following contributions. First, we are among the first to examine how to prefetch data blocks with maximum potential energy savings into buffer disks, thereby reducing the number of power-state transitions and increasing the number of standby periods to improve energy efficiency. Second, we build a new energy-saving prediction model, based on which an energy-saving calculation module was implemented for parallel I/O systems with buffer disks. Energy savings measured by the prediction model represent the importance and priority of prefetching blocks in a buffer disk to efficiently conserve energy in the disk system. Third, we developed an energy-efficient prefetching algorithm in the context of two buffer disk configurations. A greedy prefetching module was implemented to fetch blocks that have the highest energy savings. Finally, we construct models to theoretically and experimentally analyze the energy efficiency and performance of PRE-BUD. We quantitatively compared PRE-BUD with three existing techniques employed in parallel I/O systems.

The rest of the article is organized as follows. Section 2 summarizes related work in the area of energy-efficient disk systems. Section 3 presents a motivational example. Section 4 presents a prefetching module and an energy-saving calculation module to facilitate the development of energy-efficient parallel disk systems with buffer disks. Section 5 analyzes the energy efficiency and performance of PRE-BUD. In Section 6, we experimentally compare PRE-BUD with existing approaches found in the literature. The conclusion of the article and future research directions are discussed in Section 7.

2. RELATED WORK

2.1. Strengths and Limitations of Related Work

Almost all energy efficient strategies rely on DPM techniques [Benini et al. 2000]. These techniques assume a disk will have several power states. Lower power states have lower performance, so the goal is to place a disk in a lower power state if there are large idle times. There are several different approaches to generate larger idle times for individual disks. There are also several approaches to prefetch data, although many techniques have focused on low power disks.

Memory Cache Techniques. Energy efficient prefetching was explored by Papathanasiou and Scott [2004]. Their techniques relied on changing prefetching and caching strategies within the Linux kernel. PB-LRU is another energy efficient cache management strategy [Zhu et al. 2004b]. This strategy focused on providing more opportunities for underlying disk power strategies to save energy. Flash drives have also been proposed for use as buffers for disk systems [Chen et al. 2007]. Energy efficient caching and prefetching in the context of mobile distributed systems has been studied [Shen et al. 2004; Zhuang and Pande 2004]. These three research papers focus on mobile disk systems, whereas we focus on large scale parallel disk systems. All the previously mentioned techniques are limited in the fact that caches, memory, and flash disk capacities are typically smaller than disk capacities. We propose strategies that use a disk as a cache to prefetch data into. The break-even times of disk drives are usually very high and prefetch data accuracy and size become a critical factor in energy conservation.

Multispeed/Low Power Disks. Many researchers have recognized the fact that large break-even times limit the effectiveness of energy efficient power management strategies. One approach to overcome large break-even times is to use multispeed disks [Son and Kandemir 2006; Zhu et al. 2005]. Energy-efficient techniques have also relied on replacing high-performance disks with low-energy disks [Carrera et al. 2003]. Mobile computing systems have also been recognized as platforms where disk energy should

be conserved [Chen et al. 2007; Kim et al. 2007]. The mobile computing platforms use low power disks with smaller break-even times. The weakness of using multispeed disks is that there are no commercial multispeed disks currently available. Low-power disk systems are an ideal candidate for energy savings, but they may not always be a feasible alternative. Our strategies will work with existing disk arrays and do not require any changes in the hardware.

Disk as Cache. MAID was the original paper to propose using a subset of disk drives as cache for a larger disk system [Colarelli and Grunwald 2002]. MAID designed mass storage systems with the performance goal of matching tape-drive systems. PDC was proposed to migrate sets of data to different disk locations [Pinheiro and Bianchini 2004]. The goal is to load the first disk with the most popular data, the second disk with the second most popular data, and continue this process for the remaining disks. The main difference between our work and MAID is that our caching policies are significantly different. MAID caches blocks that are stored in a LRU order. Our strategy attempts to analyze the request look-ahead window and prefetch any blocks that will be capable of reducing the total energy consumption of the disk system. PDC is a migratory strategy and can cause large energy overheads when a large amount of data must be moved within the disk system. PDC also requires the overhead of managing metadata for all of the blocks in the disk system, whereas our strategy only needs metadata for the blocks in the buffer disk.

2.2. Observations

With the previously mentioned limitations of energy efficient research, we propose a novel prefetching strategy. Our research differs from the previous research on the following key points.

(1) We develop a mathematical model to analyze the energy efficiency of our prefetching strategy. This mathematical model allows us to produce simulations that offer insights into the key disk parameters that effect energy-efficiency.
(2) We develop a prefetching strategy that tries to move popular data into a set of buffer disks without affecting the data layout of any of the data disks. We also perform simulations with parallel I/O intensive applications, which previous researchers have avoided.

Our strategies also have the added benefit of not requiring any changes to be made to the overall architecture of an existing disk system. Previous work has focused on redesigning a disk system or replacing existing disks to produce energy savings. Our strategy will either add extra disks or use the current disk system to produce energy savings under certain conditions.

3. MOTIVATIONAL EXAMPLE

For a simple motivational example that demonstrates the utility of the buffer disk architecture, we present a scenario that is depicted in Figure 1. Each horizontal bar represents the time a particular disk is busy or idle. Figure 1 presents requests for individual disks that are represented by the specific patterns presented in the legend. Idle periods for all of the disks are represented with the solid pattern. If we are using the IBM 36Z15 disk for disks A, B, and C, DPM techniques will not be able to save any energy. DPM requires a disk to have an idle period greater than the break-even time. For the IBM 36Z15 disk, the break-even time is 14.5 seconds. The largest idle-period for any of the disks presented in Figure 1 is 8 seconds. This means that DPM is unable to save any energy is this example, even though there are idle periods of 8 seconds. The total energy consumed by all of the disks to serve all of the requests is approximately

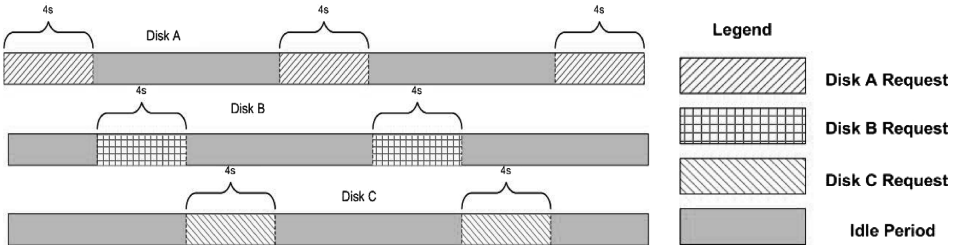

Fig. 1. Sample disk trace.

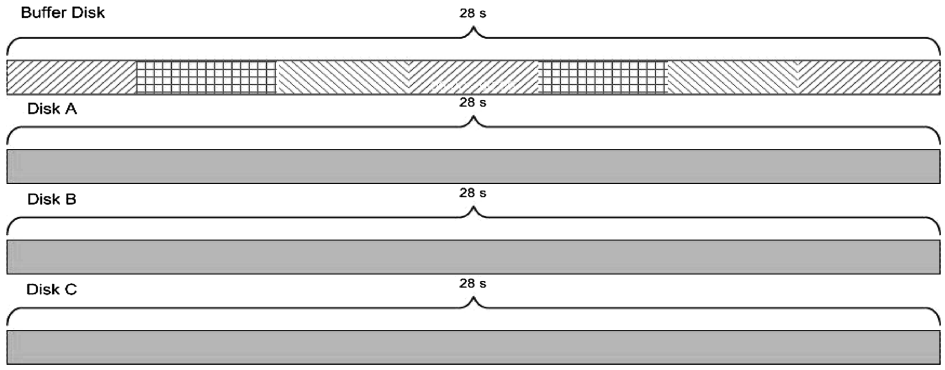

Fig. 2. Buffer disk added to architecture.

949.2 Joules. Each disk must remain in the idle state, which consumes 10.2 W, when they are not serving a request.

If we were able to prefetch the requested data from all three disks into a single disk, which is represented by Figure 2, we could have one single disk do the work of the three disks. Disks A, B, and C will be put into the sleep state and remain in the sleep state for the entire length of the trace.

Using a buffer disk allows one to trade many lightly loaded disks, for a smaller number of heavily loaded disks. The key to energy savings using a buffer disk is to accurately place frequently requested data into the buffer disk. This allows non-buffer disks to have larger idle-window sizes as compared to not using a buffer disk. If a request can be served from a buffer disk, the corresponding data disk for this particular request treats the time for the buffer disk to serve the disk request as an extra idle window. The key to energy savings with the buffer disk strategy is to have consecutive hits from the perspective of a single disk, so the disk can see a long continuous idle window. Adding an extra buffer disk represents one of our approaches, PRE-BUD1, to conserving energy in parallel storage systems. This approach will consume 804 J, which includes the energy required to prefetch the data from all three disks and to operate the buffer disk. Similarly, if you used Disk A to prefetch requested data from Disk B and Disk C, Disk A would now become a buffer disk. Disk A would remain active for 28 s, while Disk B and Disk C would sleep for 28 s. This preceding approach, PRE-BUD2, will consume 680 J. PRE-BUD1 is able to save 15.3% and PRE-BUD2 is able to save 28.4% energy over the DPM strategy. These numbers will go up if the trace presented in Figure 1 is repeated. This is because the requested blocks are already in the buffer disk and sleeping a disk is four times more energy efficient than leaving it in the idle state.

Input: a request *r*, parallel disk system with *m* disks
1 if *block(r)* is present in the buffer disk {
2 if *disk(r)* is active and $T_{Disk}(r) \leq T_0(r)$, where $T_{Disk}(r)$ and $T_0(r)$ are response time of *r* when serviced by disk(r) and the buffer disk, respectively
3 The request *r* is serviced by *disk(r)*;
4 else the request *r* is serviced by the buffer disk;
}
5 else { /* *block(r)* is not present in the buffer disk */
/* Initiate the prefetching phase */
6 if *disk(r)* is in the standby state /* spin up disk(r) when it is standby */
7 spin up *disk(r)*;
8 Compute the energy savings of references in $A \subseteq R$, where *A* is a subset of the lookahead *R*, and $\forall$ *r'*$\in$*A: disk(r')* is active;
9 Update the energy savings of blocks in the buffer disk;
10 Fetch blocks in $A^+ \cap G^+$;
11 Evicting the blocks in $G - G^+$ with the lowest energy savings as necessary, where *G* is the set of blocks present in the buffer disk;
A^+ is the set of blocks, such that if block $b \in A^+$, then *b* is referenced by a request in the lookahead, *b* is not present in the buffer disk, *disk(b)* is active (i.e., *disk(b)* $\in$ *A*), and the energy saving $E_s(b)$ of *b* is larger than 0;
G^+ is the set of blocks with the highest energy saving in $A^+ \cup G$,
11.a such that $\sum_{r' \in G^+} \lambda(r') \cdot t(r') < B_0$ /* Bandwidth constraint must be satisfied */
11.b $\sum_{r' \in G^+} s(r') \leq C_0$ /* Capacity constraint must be satisfied */
/* The request r is then serviced */
12 if *block(r)* has not been prefetched
13 The request *r* is serviced by *disk(r)*;
14 else return *block(r)*; /* *block(r)* was recently retrieved; no extra I/O is necessary */
}

Fig. 3. Algorithm PRE-BUD: The prefetching module.

4. PRE-BUD ENERGY-EFFICIENT PREFETCHING STRATEGY

In this section, we describe our energy-efficient prefetching strategy for parallel storage systems with buffer disks. Energy consumption in parallel disk systems can be reduced by placing idle disks into the standby state, which causes the idle disks to stop spinning completely. The fundamental goal of PRE-BUD is to improve energy efficiency of parallel disks through the following two energy-saving principles. First, by reducing the number of power state transitions one can decrease the energy overhead of spinning down the disks. Second, increasing the number and lengths of standby intervals can foster new opportunities to aggressively turn disks into the standby state. PRE-BUD implements these two energy saving principles using the concept of buffer disks, which contain frequently accessed data blocks that are prefetched and buffered. There are two buffer disk architectures: (1) adding buffer disks to the disk system, PRE-BUD1, and (2) using existing disks as the buffer disk(s), PRE-BUD2. The energy-efficient prefetching strategy, PRE-BUD, described in this article can be successfully applied to deal with the two approaches to the architecture. In this article, let us first focus on parallel disk systems with a single buffer disk. Then, in Section 7, we briefly discuss how to extend PRE-BUD to conserve energy in parallel disk systems with multiple buffer disks.

The PRE-BUD strategy is a greedy algorithm in the sense that blocks fetched into a buffer disk in each prefetching phase (see Steps 8–11 in Figure 3) are the ones that have the highest energy savings, which in turn attempts to maximize the energy efficiency of the parallel disk system. PRE-BUD has two key components: the prefetching module and the energy-saving calculation module. Given a parallel disk system with a buffer

Table I. Notation for the Description of the Prefetching Module

Notation	Description
R	Current lookahead. $r \in R$ is a reference in the lookahead
$block(r)$	Block accessed in reference $r \in R$
$disk(r)$	Disk in which $block(r)$ is residing
A	Subset of the lookahead R; for any r in A, $disk(r)$ is active, i.e., $\forall r \in A$: $disk(r)$ is active
G	A set of blocks present in the buffer disk
$E_s(b)$	Energy saving contributed by prefetching block b
A^+	For any $b \in A^+$, we have $disk(b) \in A$, $E_s(b) > 0, b \notin G, and \exists r \in R : block(r) = b$
G^+	The set of blocks with the highest energy savings in $A^+ \cup G$

disk, the prefetching module determines which blocks to fetch from any of the parallel disks to improve the energy efficiency of the entire disk system. If the buffer disk is full while more blocks have to be fetched, the prefetching module is tasked with deciding which blocks need to be evicted. The prefetching module relies on the second module to calculate and update the energy savings of referenced blocks in the current lookahead window and blocks present in the buffer disk. The energy savings estimate of a block in a data disk quantifies the energy consumption reduction produced by fetching the block into a buffer disk. On the other hand, the energy savings estimate of a block in the buffer disk reflects the energy savings value of caching the block instead of evicting it from the buffer disk. The prefetching and energy-saving calculation modules are detailed in Sections 4.1 and 4.2, respectively.

4.1. Prefetching Module

Before presenting the prefetching module of PRE-BUD, we first summarize the notation for the description of the prefetcher in Table I.

Figure 3 outlines the prefetching module in PRE-BUD. PRE-BUD is energy-efficient in nature, because a request for data in a disk currently in the standby mode will not have to be spun up to serve the request if the requested block is present in the buffer disk (see Step 4). Buffer-disk resident blocks allow standby data disks to stay in the low-power state for an increased period of time, as long as accessed blocks are present in the buffer disk. There is a side effect of making the buffer disk perform I/O operations while placing data disks in standby longer; that is, the buffer disk is likely to become a performance bottleneck. To address the bottleneck issue properly, we design the prefetcher in such a way that the load between the buffer and data disks is balanced, if the active data disk can achieve a shorter response time than the buffer disk we don't rely on the buffer disk (see step 2). In addition to load balancing, utilization control is introduced to prevent disk requests from experiencing unacceptably long response times. In light of the utilization control, the prefetching module ensures that the aggregated required I/O bandwidth is lower than the maximum bandwidth provided by the buffer disk (see Line 11.a in Figure 3).

To improve the energy efficiency of PRE-BUD, we force PRE-BUD to fetch blocks from data disks into the buffer disk on a demand basis (see Line 5 in Figure 3). Thus, block b is prefetched in Step 10 only when the following four conditions are met. First, a request r in the lookahead is accessing the block, that is, $\exists r \in R : block(r) = b$. Second, the block is not present in the buffer disk, that is, $b \notin G$. Third, fetching the blocks and caching them into the buffer disk can improve energy efficiency, that is, $Es(b) > 0$. Last, the block is residing in an active data disk, that is, $disk(b) \in A$. Note that set A^+ (see Table I) contains all the blocks that satisfy these criteria.

To maximize energy efficiency, we have to identify data-disk-resident blocks with the highest energy savings potential. This step is implemented by maintaining a set,

G^+, of blocks with the highest energy saving in $A^+ \cup G$. Thus, blocks in $A^+ \cap G^+$ are the candidate blocks to be prefetched in the prefetching phase. A tie of energy savings between a buffer-disk-resident block and a data-disk-resident block can be broken in favor of the buffer-disk-resident block. If two data-disk-resident blocks have the same energy saving, the tie is broken in favor of the block accessed earlier by a request in the look-ahead.

In the case that the buffer disk is full, blocks in $G - G^+$ must be evicted from the buffer disk (see Step 11 in Figure 3). This is because $G - G^+$ contains the blocks with the lowest energy savings. We assign zero to the energy savings of buffer-disk-resident blocks that will not be accessed by any requests in the lookahead. The buffer-disk-resident blocks without any contribution to energy conservation will be among the first to be evicted from the buffer disk, if a disk-resident block with high energy saving must be fetched when the buffer disk is full. Blocks that will not be accessed in the look-ahead are evicted in the least-recently-used order.

PRE-BUD can conserve more energy by the virtue of its on-demand manner, which defers prefetching decisions till the last possible moment when these two criteria are satisfied. Deferring the prefetching phase is beneficial, because (1) this phase needs to spin up a corresponding disk if it is in the standby state, and (2) late prefetching leads to a larger lookahead for better energy-aware prefetching decisions.

The prefetching module can be readily integrated with a disk scheduling mechanism, which is employed to independently optimize low-level disk access times in each individual disk. This integration is implemented by batching disk requests and offering each disk an opportunity to reschedule the requests to optimize low-level disk access performance.

4.2. Energy-Saving Calculation Module

We develop an energy-saving prediction model, based on which we implement the energy-saving calculation module invoked in Steps 8 and 9 in the prefetching module (see Figure 3). The prediction model along with the calculation module is indispensable for the prefetcher, because the energy savings of a block represents the importance and priority of placing the block in the buffer disk to reduce the energy consumption of the disk system. The energy-saving calculation module can illustrate the amount of energy conserved by fetching a block from a data disk into a buffer disk. It also calculates the utility of caching a buffer-disk-resident block rather than evicting it from the buffer disk. Table II summarizes the notation for the description of the energy-saving calculation module.

To analyze circumstances under which prefetching blocks can yield energy savings, we focus on a single referenced block stored in a data disk. Let $R_j \subseteq R$ be a set of references accessing blocks in the jth data disk. Thus, R_j is a subset of the lookahead R and can be defined as

$$R_j = \{r | r \in R \wedge disk(r) = j\text{th data disk} \wedge \text{block}(r) = b_{k,j} \wedge b_{k,j} \notin G\}.$$

Given a set $R_{kj} \subseteq R$ of references accessing the kth block b_{kj} in the jth data disk, let us derive the energy saving $E_s(b_{kj})$ achieved by fetching b_{kj} from the data disk into the buffer disk. R_{kj} is comprised of all the requests referencing a common block bkj that is not present in the buffer disk; therefore, R_{kj} can be formally expressed as $R_{k,j} = \{r \,|\, r \in R \wedge block(r) = b_{k,j} \wedge b_{k,j} \notin G\}$. Intuitively, energy savings $E_s(b_{kj})$ can be computed by considering the energy consumption incurred by each disk request in R_{kj}.

Given a reference list R_j and a block b_{kj}, in what follows, we identify four cases where a reference in R_j can contribute to positive energy savings by the virtue of prefetching block b_{kj}. First, we introduce two energy-saving principles utilized by PRE-BUD.

Table II. Notation for the Description of the Energy-Saving Calculation Module

Notation	Description
R_j	A set of references accessing blocks in the jth data disk
$R_{k,j} \subseteq R$	A set of references accessing the kth block $b_{k,j}$ in the jth data disk
$b_{k,j}$	The kth block in the jth data disk
T_{BE}	Break-even time. Minimum idle time required to compensate the cost of entering the standby state
T_{ij}	Active time period serving the ith request issued to the jth data disk
t_{ij}	Time spent serving the ith request issued to the jth data disk
α_{ij}	Time spent in the idle period prior to the ith request accessing a block in disk j
I_{ij}	An idle period prior to the ith request accessing a block in the jth data disk
n_j	The total number of requests (in the lookahead) issued to the jth disk
Φ_j	A set of disk access activities for references in R_j
$time(b_{kj})$	Active time period to serve a request accessing block b_{kj}
$block(T_{ij})$	A block accessed during the active period T_{ij}
T_D	Time to transition from active/idle to standby
T_U	Time to transition from standy to active mode
E_D	Energy overhead of transitioning from active/idle to standby
E_U	Energy overhead of transitioning from standby to active mode
P_A, P_I, P_S	Disk power in the active, idle, and standby mode

Energy Saving Principle 1. To increase the length and number of idle periods larger than the disk break-even time T_{BE}, which is the minimum disk standby time required to offset the cost of entering the standby state. This principle can be realized by combining two adjacent idle periods to form a single idle period that is larger than T_{BE}. PRE-BUD fetches, in advance, a block accessed between two adjacent idle periods, thereby possibly forming a larger inactivity time that allows the disk to enter the standby state to conserve energy.

Energy Saving Principle 2. To reduce the number of power-state transitions. The energy efficiency of a disk can be further improved by minimizing the energy cost of spinning up and down disks. Disk vendors can provide high-quality disks with low spin-up/down energy overheads; PRE-BUD aims to reduce the number of disk spin-up and spin-down while enlarging disk idle times. We implement this principle in PRE-BUD by combining two adjacent standby periods to eliminate unnecessary state transitions between the two standby periods.

Now we investigate cases which exploit the above energy saving principles to conserve energy in disks. Let $\Phi_j = \{I_{1j}, T_{1j}, I_{2j}, T_{2j}, \ldots I_{ij}, T_{ij}, \ldots I_{nj,j}, T_{nj,j}\}$ be a set of disk accesses for references in R_j, where for an active period T_{ij}, t_{ij} is the time spent serving the ith request issued to data disk j; for idle period I_{ij}, α_{ij} is the time spent in the idle period prior to the ith request accessing a block in the jth data disk, and n_j is the total number of requests issued to the jth disk. We denote $block(T_{ij})$ as a block accessed during the active period T_{ij}.

The following three cases demonstrate scenarios that apply energy-saving principle 1 to generate longer idle periods (i.e., longer than T_{BE}) by prefetching $block(T_{ij})$ to combine the ith and $(i+1)$th idle periods. Let us pay attention to the ith active period T_{ij} and the two periods I_{ij} and $I_{(i+1)j}$ (i.e., the ones adjacent to T_{ij}). Cases 1–3 share two common conditions—(1) both I_{ij} and $I_{(i+1)j}$ are larger than zero and (2) the summation of t_{ij}, α_{ij}, and $\alpha_{(i+1)j}$ is larger than the break-even time T_{BE}.

Case 1. Both the ith and $(i+1)$th idle periods are equal to or smaller than the break-even time T_{BE}. Thus, we have $0 < \alpha_{ij} \leq T_{BE}$, $0 < \alpha_{(i+1)j} \leq T_{BE}$, and $\alpha_{ij} + t_{ij} + \alpha_{(i+1)j} > T_{BE}$.

Case 2. The ith idle period is equal to or smaller than the break-even time T_{BE}; the $(i+1)$th idle period is larger than T_{BE}. Formally, we have $0 < \alpha_{ij} \leq T_{BE}$,$\alpha_{(i+1)j} > T_{BE}$, and $\alpha_{ij} + t_{ij} + \alpha_{(i+1)j} > T_{BE}$.

Case 3. The ith idle period is larger than T_{BE}; the $(i+1)$th idle period is equal to or smaller than T_{BE}. The conditions for Case 3 can be expressed as: $\alpha_{ij} > T_{BE}$,$0 < \alpha_{(i+1)j} \leq T_{BE}$, and $\alpha_{ij} + t_{ij} + \alpha_{(i+1)j} > T_{BE}$.

Now we calculate, in these three cases, the energy savings produced by fetching $block(T_{ij})$ from the jth data disk to the buffer disk. The calculation makes use of the following definitions.

—Let P_A, P_I, and P_S represent the disk power consumption in the active, idle, and standby modes. Let T_D and T_U be times to transition to the standby and active mode; let E_D and E_U be energy overhead to transition to standby and active.
—E_{WOP} denotes energy consumption of the periods t_{ij}, α_{ij}, and $\alpha_{(i+1)j}$ when PRE-BUD is not applied.
—In case of having $block(T_{ij})$ prefetched, E_{WPF} denotes energy consumption of the jth disk in the periods t_{ij}, α_{ij}, and $\alpha_{(i+1)j}$.
—E_{BUD} represents energy consumption of the buffer disk accessing the prefetched $block(Tij)$.
—For $block(b_{kj})$, active time spent serving a request accessing the block is denoted by $time(b_{k.j})$.

Energy savings, $E_S(block(T_{ij}))$, contributed by prefetching $block(T_{ij})$ can be written as:

$$E_S(\text{block}(T_{ij})) = E_{WOP} - (E_{WPF} + E_{BUD}). \tag{1}$$

Energy Savings, $E_S(block(T_{ij})$, *in Case* 1. For case 1, I_{ij} and $I_{(i+1)j}$ are equal to or smaller than T_{BE}. This condition implies that the jth disk is in the idle mode during I_{ij} and $I_{(i+1)j}$. Energy consumption experienced by the disk in active period T_{ij} is $P_A \cdot t_{ij}$. Hence, E_{WOP} in case 1 can be expressed as:

$$E_{WOP} = P_I \cdot (\alpha_{ij} + \alpha_{(i+1)j}) + P_A \cdot t_{ij}. \tag{2}$$

When $block(T_{ij})$ is prefetched, a large (i.e., larger than T_{BE}) idle period can be formed by combining the periods T_{ij}, I_{ij}, and $I(i+1)j$. Therefore, E_{WPF} can be computed as the energy consumption of the jth disk in the standby mode during T_{ij}, I_{ij}, and $I_{(i+1)j}$. Taking into account energy overhead of power state transitions, we can calculate E_{WPF} using the following equation:

$$E_{WPF} = P_S \cdot (\alpha_{ij} + t_{ij} + \alpha_{(i+1)j} - T_D - T_U) + E_D + E_U. \tag{3}$$

We assume that the buffer disk and data disks are identical; therefore, energy consumption E_{BUD} of the buffer disk accessing the prefetched $block(T_{ij})$ is

$$E_{BUD} = P_A \cdot t_{ij}. \tag{4}$$

$E_S(block(T_{ij}))$ in case 1 can be determined by substituting Eqs. (1)–(4) into Eq. (1). Hence, we have:

$$E_S(\text{block}(T_{ij})) = P_I \cdot (\alpha_{ij} + \alpha_{(i+1)j}) - P_S \cdot (\alpha_{ij} + t_{ij} + \alpha_{(i+1)} - T_U - T_D) - E_D - E_U. \tag{5}$$

Energy Saving, $E_S(block(T_{ij}))$, *in case* 2. The jth disk in this case is transitioned into standby during $I_{(i+1)j}$, since $I_{(i+1)j}$ is larger than T_{BE}. The energy consumption of the disk in $I_{(i+1)j}$ is expressed as (see the third term on the right-hand side of Eq. (6)). Thus, the energy consumption E_{WOP} of the disk in T_{ij}, Iij, and $I_{(i+1)j}$ is:

$$E_{WOP} = P_I \cdot \alpha_{ij} + P_A \cdot t_{ij} + (P_S \cdot (\alpha_{(i+1)j} - T_D - T_U) + E_D + E_U) \tag{6}$$

We derive $E_S(block(T_{ij}))$ in case 2 by substituting Eqs. (6), (3), and (4) for for E_{WOP}, E_{WPF}, and E_{BUD}. Thus, we have

$$E_S(\text{block}(T_{ij})) = P_I \cdot \alpha_{ij} - P_S \cdot (\alpha_{ij} + t_{ij}) \tag{7}$$

Energy Savings, $E_S(block(T_{ij}))$, *in Case* 3. The Energy saving $E_S(block(T_{ij}))$ in this case is very similar to that in case 2 except that the jth disk is transitioned into standby during I_{ij} rather than $I_{(i+1)j}$. Consequently, the energy saving $E_S(block(T_{ij}))$ in case 3 can be written as:

$$E_S(\text{block}(T_{ij})) = P_I \cdot \alpha_{(i+1)j} - P_S \cdot (\alpha_{(i+1)j} + T_{ij}) \tag{8}$$

Case 4. The case described here shows a scenario that applies energy saving principle 2 to reduce power-state transitions by prefetching $block(T_{ij})$ to combine two adjacent standby periods I_{ij} and $I_{(i+1)j}$.

Energy Saving, $E_S(block(T_{ij}))$ *in case* 4. In this case, both α_{ij} and $\alpha_{(i+1)j}$ are larger than T_{BE}, meaning that the jth disk can be standby in these two time intervals to conserve energy. Formally, we have $\alpha_{ij} > T_{BE}$,$\alpha_{(i+1)j} > T_{BE}$, and $\alpha_{ij} + t_{ij} + \alpha_{(i+1)j} > T_{BE}$. Thus, energy consumption E_{WOP} of the jth disk without a buffer disk is:

$$\begin{aligned} E_{WOP} = {} & P_A \cdot t_{ij} + (P_S \cdot (\alpha_{ij} - T_D - T_U) + E_D + E_U) \\ & + (P_S \cdot (\alpha_{(i+1)j} - T_D - T_U) + E_D + E_U), \end{aligned} \tag{9}$$

where the second and third term on the right-hand side of Eq. (9) are the energy consumed by the disk in standby periods I_{ij} and $I_{(i+1)j}$, respectively. With a buffer disk in place, the energy consumption E_{WPF} and E_{BUD} in this case are the same as in case 1 (see Eqs. (3) and (4)). Therefore, the energy savings, $ES(block(T_{ij}))$, in this case is derived from E_{WOP} (see Eq. (9)), E_{WPF}, and E_{BUD} as:

$$E_S(\text{block}(T_{ij})) = E_D + E_U - P_S \cdot (T_D + T_U + t_{ij}). \tag{10}$$

Case 5 summarizes scenarios where prefetching a block may have negative impacts on the energy efficiency.

Case 5. If the summation of t_{ij}, α_{ij}, and $\alpha_{(i+1)j}$ is smaller than or equal to T_{BE}, that is, $\alpha_{ij} + t_{ij} + \alpha_{(i+1)j} \leq T_{BE}$, then prefetching block bkj causes an negative impact on energy conservation.

Energy savings, $E_S(block(T_{ij}))$, *in Case* 5. Since $\alpha_{ij} + t_{ij} + \alpha_{(i+1)j} \leq T_{BE}$, the disk j stays in the idle mode during periods T_{ij}, I_{ij}, and $I_{(i+1)j}$. If the block b_{kj} is prefetched to the buffer disk, the energy consumption E_{WPF} of disk j in the three periods is:

$$E_{WPF} = P_I \cdot (\alpha_{ij} + t_{ij} + \alpha_{(i+1)j}). \tag{11}$$

The values of E_{WOP} and E_{BUD} are the same as those of case 1 (see Eq. (2)). Applying E_{WOP}, $EWPF$ and $EBUD$ to Eq. (1), we estimate the negative energy-saving impact $E_S(block(T_{ij}))$ as:

$$E_S(\text{block}(T_{ij})) = -P_I \cdot t_{ij}. \tag{12}$$

In light of these four cases, the set Φ_{kj} of disk activities for references accessing block b_{kj} in disk j can be partitioned into the following four disjoint subsets,

$$\Phi_{k,j} = \Phi_{k,j,1} \cup \Phi_{k,j,2} \cup \Phi_{k,j,3} \cup \Phi_{k,j,4} \cup \Phi_{k,j,5}, \tag{13}$$

where $\Phi_{k,j,1}$, $\Phi_{k,j,2}$, $\Phi_{k,j,3}$, $\Phi_{k,j,4}$ and $\Phi_{k,j,5}$ contain active time periods that respectively satisfy the conditions of the four energy-saving cases. The four subsets can be defined as:

$$\begin{aligned} \Phi_{k,j,1} = {} & T_{ij} | \text{block}(T_{ij}) = b_{k,j} \wedge 0 < \alpha_{ij} \leq T_{BE} \wedge 0 < \alpha_{(i+1)j} \\ \leq {} & T_{BE} \wedge \alpha_{ij} + t_{ij} + \alpha_{(i+1)j} > T_{BE} \end{aligned}$$

for case 1;

$$\begin{aligned}\Phi_{k,j,2} &= T_{ij}|\text{block}(T_{ij}) = b_{k,j} \wedge 0 < \alpha_{ij} \leq T_{BE} \wedge \alpha_{(i+1)j} \\ &> T_{BE} \wedge \alpha_{ij} + t_{ij} + \alpha_{(i+1)j} > T_{BE}\end{aligned}$$

for case 2;

$$\begin{aligned}\Phi_{k,j,3} &= T_{ij}|\text{block}(T_{ij}) = b_{k,j} \wedge \alpha_{ij} > T_{BE} \wedge 0 < \alpha_{(i+1)j} \\ &\leq T_{BE} \wedge \alpha_{ij} + t_{ij} + \alpha_{(i+1)j} > T_{BE}\end{aligned}$$

for case 3;

$$\begin{aligned}\Phi_{k,j,4} &= T_{ij}|\text{block}(T_{ij}) = b_{k,j} \wedge \alpha_{ij} > T_{BE} \wedge \alpha_{(i+1)j} \\ &> T_{BE} \wedge \alpha_{ij} + t_{ij} + \alpha_{(i+1)j} > T_{BE}\end{aligned}$$

for case 4; and $\Phi_{k,j,5} = \{T_{ij}|\text{block}(T_{ij}) = b_{k,j} \wedge \alpha_{ij} + t_{ij} + \alpha_{(i+1)j} \leq T_{BE}\}$ for case 5.

Now we are positioned to show the derivation of energy savings, $E_S(b_{kj})$, yielded by fetching block b_{kj} from disk j to the buffer disk. Thus, $E_S(b_{kj})$ can be derived from Eqs. (5), (7), (8), (10), (11), and Eq. (12), where the last item on the right-hand side is the energy overhead of fetching b_{kj} from disk j to the buffer disk.

$$\begin{aligned}E_S(b_{k,j}) &= \sum_{T_{ij} \in \Phi_{k,j}} (E_S(\text{block}(T_{ij}))) \\ &= \sum_{T_{ij} \in \Phi_{k,j,1}} (P_I \cdot (\alpha_{ij} + \alpha_{(i+1)j}) - P_S \cdot (\alpha_{ij} + t_{ij} + \alpha_{(i+1)} - T_U - T_D) - E_D - E_U) \\ &\quad + \sum_{T_{ij} \in \Phi_{k,j,2}} (P_I \cdot \alpha_{ij} - P_S \cdot (\alpha_{ij} + t_{ij})) + \sum_{T_{ij} \in \Phi_{k,j,3}} (P_I \cdot \alpha_{(i+1)j} - P_S \cdot (\alpha_{(i+1)j} + t_{ij})) \\ &\quad + \sum_{T_{ij} \in \Phi_{k,j,4}} (E_D + E_U - P_S \cdot (T_D + T_U + t_{ij}).) - P_I \cdot \sum_{T_{ij} \in \Phi_{k,j,5}} t_{ij} - P_A \cdot \text{time}(b_{k,j}).\end{aligned} \tag{14}$$

Given the kth block bkj residing in disk j, the algorithm used to compute the energy savings of prefetching block b_{kj} from the data disk to the buffer disk is described in Figure 4. All of the energy-saving cases are handled explicitly from Steps 3 through 14; whereas Step 15 addresses the issue of negative energy savings. The time complexity of the energy-saving calculation module is low, because the time complexity of this routine for each block is $O(nj)$, where nj is the number of requests in the lookahead corresponding to the jth disk. After the block b_{kj} is fetched to the buffer disk, = $\{I_{1j}, T_{1j}, I_{2j}, T_{2j}, \ldots I_{ij}, Tij, \ldots I_{nj,j}T_{nj,j}\}$ the set j of disk access activities for references in R_j must be updated by deleting any $T_{ij} \in \Phi_j$ accessing b_{kj}, that is, $block(T_{ij}) = b_{kj}$.

5. ANALYSIS OF PRE-BUD

In this section, we analyze the energy efficiency and performance of PRE-BUD. We start the analysis by showing the energy consumption of a full-power baseline system without turning any disks into the standby state. Next, we analyze the energy consumption of a parallel disk system with the dynamic power management (DPM) technique. Last, our analysis will be focused on the energy consumption and response time of parallel disk systems with PRE-BUD.

5.1. A Full-Power Baseline System

In this section, we describe an energy consumption model, which is built to quantitatively calculate energy consumption of a modeled parallel disk systems. We model

```
Input: block b_{k,j}, disk j, a set Φ_j of disk access activities; Output: E_S(b_{k,j})
1  Initialize E_S(b_{k,j}) to 0;
2    for (i = 1 to n_j) {
3      if ( α_{ij} + t_{ij} + α_{(i+1)j} > T_BE ) {  /* Cases 1-4 */
4        if ( 0 < α_{ij} ≤ T_BE ) {
5          if ( 0 < α_{(i+1)j} ≤ T_BE )  /* Case 1, see Eq. (4.5) */
6            E_S(b_ij) = E_S(b_ij) + P_I · (α_ij + α_{(i+1)j}) − P_S · (α_ij + t_ij + α_{(i+1)} − T_U − T_D) − E_D − E_U ;
7          else E_S(b_ij) = E_S(b_ij) + P_I · α_ij − P_S · (α_ij + t_ij) ;/* Case 2, see Eq. (4.7) */
8        }
9        else {
10         if ( 0 < α_{(i+1)j} ≤ T_BE ) /* Case 3, see Eq. (4.8) */
11           E_S(b_ij) = E_S(b_ij) + P_I · α_{(i+1)j} − P_S · (α_{(i+1)j} + T_ij) ;
12         else E_S(b_ij) = E_S(b_ij) + E_D + E_U − P_S · (T_D + T_U + t_ij). /* Case 4, see Eq. (4.10) */
13       }
14     } /* end Cases 1-4 */
15     else E_S(b_ij) = E_S(b_ij) − P_I · t_ij ; /* Negative energy saving. Case 5, see Eq. (4.12) */
16   } /* end for */
17   return E_S(b_ij) − P_A · time(b_{k,j})
```

Fig. 4. Algorithm PRE-BUD: The energy-saving calculation module.

the power of a parallel disk system with m disks as a vector $P = (P_1, P_2, \ldots, P_m)$. The power P_i of the ith disk is represented by three parameters, that is, $P_i = (P_{A,i}, P_{I,i}, P_{S,i})$, where $P_{A,i}$, $P_{I,i}$, and $P_{S,i}$ are the power of the ith disk when it is in the active, idle, and standby state, respectively. Let $e_{j,i}$ be the energy consumption to serve the jth request served by the ith disk. We denote the energy consumption rate of the disk when it is active by $P_{A,i}$ and the energy consumption $e_{j,i}$ can be written as

$$e_{j,i} = x_{j,i} \cdot P_{A,i} \cdot t_{j,i} = x_{j,i} \cdot P_{A,i} \cdot \left(t_{SK,j,i} + t_{RT,j,i} + \frac{s_j}{B_i} \right), \tag{15}$$

where $t_{j,i}$ is the service time of request j on disk i. $t_{j,i}$ is the summation of $t_{SK,j,i}$, $t_{RT,j,i}$, and s_j/B_i, which are the seek time and rotational latency of the request, and the data transfer time depending on the data size s_j and the transfer rate B_i of the disk. Element $x_{j,i}$ is "1" if request j is responded by the ith disk and is "0", otherwise. Since each request can be served by only one disk, we have $\sum_{i=1}^{m} x_{j,i} = 1$.

Given a reference string R, we can compute the energy E_A consumed by serving all requests as

$$\begin{aligned} E_A(P, R) &= \sum_{i=1}^{m} \sum_{j=1}^{n} e_{j,i} = \sum_{i=1}^{m} \sum_{j=1}^{n} (x_{j,i} \cdot P_{A,i} \cdot t_{j,i}) \\ &= \sum_{i=1}^{m} \sum_{j=1}^{n} \left(x_{j,i} \cdot P_{A,i} \cdot \left(t_{SK,j,i} + t_{RT,j,i} + \frac{s_j}{B_i} \right) \right). \end{aligned} \tag{16}$$

We define f_j as the completion time of request r_i. in the reference string. Then, we obtain the analytical formula for the energy consumed when disks are idle:

$$E_I(P, R) = \sum_{i=1}^{m} (P_{I,i} \cdot T_{I,i}), \tag{17}$$

where $T_{I,i}$ is the time interval when the ith disk is idle. $T_{I,i}$ can be derived from the total disk I/O processing time and completion time of the last request served by the disk. Thus, we have

$$T_{I,i} = \max_{j=1}^{n}(x_{j,i} \cdot f_j) - \sum_{j=1}^{n}\left(x_{j,i} \cdot \left(t_{SK,j,i} + t_{RT,j,i} + \frac{s_j}{B_i}\right)\right), \tag{18}$$

where the first term on the right-hand side of Eq. (18) is the summation of I/O processing times and disk idle times, and the second term is the total I/O time. The total energy consumption E_{NEC} of a parallel disk system without placing any disk into the standby state is derived from Eqs. (16) and (17) as

$$\begin{aligned} E_{NEC}(P, R) &= E_A(P, R) + E_I(P, R) \\ &= \sum_{i=1}^{m}\sum_{j=1}^{n} e_{j,i} + \sum_{i=1}^{m}(P_{I,i} \cdot T_{I,i}). \end{aligned} \tag{19}$$

5.2. Dynamic Power Management

Energy in disks systems can be efficiently reduced by employing the dynamic power management (DPM) strategy, which places disks into standby when they are idle. To analyze the energy efficiency of PRE-BUD, it is important and intriguing to model energy consumption in a DPM-based parallel disk system. If there is an idle time of the ith disk that is larger than the break-even time $T_{BE,i}$, then energy conservation can be achieved by putting the disk into the standby state. Otherwise, the energy penalty to transition between the high-power and low-power state is unable to be offset by the energy conserved. Let $P_{TR,i}$ be the power of state transitions in the ith disk. Let $P_{AS,i}$ and $P_{SA,i}$ denote additional power introduced by transitions from active to standby, and vice-versa. $P_{TR,i}$ can be derived from $P_{AS,i}$ and $P_{SA,i}$ as

$$P_{TR,i} = P_{AS,i} + P_{SA,i} = \frac{T_{AS,i} \cdot P_{AS,i} + T_{SA,i} \cdot P_{SA,i}}{T_{AS,i} + T_{SA,i}}, \tag{20}$$

where the numerator is the energy consumption caused by a pair of transitions and the denominator is the transition time. In light of Eq. (20), one can calculate the break-even time $T_{BE,i}$ as

$$T_{BE,i} = \begin{cases} (T_{AS,i} + T_{SA,i}) \cdot \left(1 + \frac{P_{TR,i} - P_{A,i}}{P_{A,i} - P_{S,i}}\right) & \text{if } P_{TR,i} > P_{A,i} \\ T_{AS,i} + T_{SA,i} & \textbf{otherwise} \end{cases} \tag{21}$$

In what follows, we make use of $T_{BE,i}$ to quantify the energy consumption of a parallel disk system when the DPM technique is employed. Suppose the number of idle time intervals in a disk i is N_i; a sequence of idle periods in the disk can be expressed as $(t_{I,i,1}, t_{I,i,2,} \ldots, t_{I,i,Ni})$, where $t_{I,i,k}$ represents the length of the kth idle period in the sequence. Let $\widehat{E}_I(P, R)$ be the energy consumed when disks are idle. The expression of $\widehat{E}_I(P, R)$ is given as

$$\begin{aligned} \hat{E}_I(P, R) &= \sum_{i=1}^{m}(P_{I,i} \cdot \hat{T}_{I,i}) \\ &= \sum_{i=1}^{m}\left(P_{I,i} \cdot \sum_{k=1}^{N_i}(y_{k,i} \cdot t_{I,i,k})\right), \end{aligned} \tag{22}$$

where $\hat{T}_{I,i}$ is the summation of small idle time intervals that are unable to compensate the cost of transitioning to the standby state. $\hat{T}_{I,i}$ can be derived from a step function $y_{k,i}$, where $y_{k,i}$ is "1" if the idle interval is smaller than or equal to the break-even time. Otherwise, $y_{k,i}$ is "0". Using the step function $y_{k,i}$, we can express $\hat{T}_{I,i}$ in Eq. (22) as

$$\hat{T}_{I,i} = \sum_{k=1}^{N_i} (y_{k,i} \cdot t_{I,i,k}).$$

The energy consumption of the parallel disk system when the disks are in the standby state can be expressed as

$$\begin{aligned} E_S(P,R) &= \sum_{i=1}^{m} (P_{S,i} \cdot T_{S,i}) \\ &= \sum_{i=1}^{m} \left(P_{S,i} \cdot \sum_{k=1}^{N_i} (\bar{y}_{k,i} \cdot (t_{I,i,k} - T_{BE,i})) \right), \end{aligned} \tag{23}$$

where $T_{S,i}$ is the time period when disk i is in the standby state. Similar to $\hat{T}_{I,i}$, $T_{S,i}$ is derived from a step function $\bar{y}_{k,i}$, where $\bar{y}_{k,i}$ is "1" if the idle interval is larger than $T_{BE,i}$, and is "0", otherwise. With the step function $\bar{y}_{k,i}$, we can model $T_{S,i}$ in Eq. (23) as

$$T_{S,i} = \sum_{k=1}^{N_i} (\bar{y}_{k,i} \cdot (t_{I,i,k} - T_{BE,i})).$$

Similarly, in this article, we obtain the formula for the energy consumption of disk power-state transitions

$$E_{TR}(P,R) = \sum_{i=1}^{m} (P_{TR,i} \cdot T_{TR,i}), \tag{24}$$

where $P_{TR,i}$ is determined by Eq. (20), $T_{TR,i}$ is the time interval when disk i is transitioning from one power state into another. $T_{TR,i}$ can be derived from $T_{BE,i}$. Hence, we obtain

$$T_{TR,i} = \sum_{k=1}^{N_i} (\bar{y}_{k,i} \cdot T_{BE,i}).$$

The energy consumption E_{DPM} of the parallel disk system with the DPM technique is the summation of the energy incurred by the disks when they are in the active, idle, standby, and transition states. Thus, E_{DPM} can be derived from Eqs. (16), (22), (23), and (24) as

$$E_{DPM}(P,R) = E_A(P,R) + \hat{E}_I(P,R) + E_S(P,R) + E_{TR}(P,R). \tag{25}$$

5.3. Derivation of Energy Efficiency for PRE-BUD

Now we analyze the energy efficiency of the PRE-BUD strategy. We only analyze the energy consumption of a parallel I/O system with PRE-BUD where an extra disk is added to the system as a buffer disk. PRE-BUD using an existing disk can be modeled similarly and can be derived from the models presented in this section.

First of all, we analyze the energy overhead E_{PF} introduced by prefetching the popular data blocks from data disks to the buffer disk. Let $D = (D_1, D_2, \ldots, D_q)$ be a set of data blocks retrieved by reference string R. We make use of a predicate $\alpha_{j,i,k}$, which

asserts that request r_i is accessing data block k on disk i, to partition the reference string in a way that requests accessing the same kth block on disk i can be grouped into the one set $R_{k,i}$. Thus, we have

$$R_{k,i} = \{r_j \in R \mid x_{j,i} = 1 \wedge \alpha_{j,i,k} = TRUE\}. \tag{26}$$

The sizes of all the requests in $R_{k,i}$ are identical. For simplicity, we denote the size of requests in $R_{k,i}$ as $s_{k,i}$. The following property must be satisfied:

$$\forall 1 \le j \le n,\ 1 \le k \le q,\ r_j \in R_{k,i} :\ s_j = s_{k,i}. \tag{27}$$

In most cases, it is impossible for a buffer disk to cache all the popular data sets. Therefore, we introduce the following step function to distinguish data blocks prefetched from data disks to the buffer disk:

$$z_{k,i} = \begin{cases} 1 \textbf{ if block } k \textbf{ on disk } i \textbf{ is prefetched,} \\ 0 \textbf{ otherwise}. \end{cases} \tag{28}$$

Energy consumption E_{PF} caused by prefetching contains two components: energy consumption $E_{R,PF}$ of reading frequently accessed data blocks from the data disks and energy consumption $E_{W,PF}$ of placing the data blocks to the buffer disk. Thus, E_{PF} is quantified as follows:

$$\begin{aligned} E_{PF}(P, D) &= E_{R,PF}(P, D) + E_{W,PF}(P, D) \\ &= \sum_{i=1}^{m}\sum_{k=1}^{q}\left(z_{k,i} \cdot P_{A,i} \cdot \left(t_{SK,k,i} + t_{RT,k,i} + \frac{s_{k,i}}{B_{R,i}}\right)\right) \\ &\quad + \sum_{i=1}^{m}\sum_{k=1}^{q}\left(z_{k,i} \cdot P_{A,0} \cdot \left(t_{SK,k,0} + t_{RT,k,0} + \frac{s_{k,i}}{B_{W,0}}\right)\right), \end{aligned} \tag{29}$$

where $P_{A,0}$ is the power of the buffer disk in the active state, $B_{R,i}$ is the read transfer rate of data disk i, and $B_{W,0}$ is the write transfer rate of the buffer disk. Next, let us derive expressions to calculate the energy consumption E_0 in the buffer disk. E_0 is the summation of active, idle, and sleep state energy consumption totals of the buffer disk, and power state transition overheads. Thus,

$$E_0 = E_{A,0} + E_{I,0} + E_{S,0} + E_{TR,0} \tag{30}$$

where $E_{A,0}$, $E_{I,0}$, and $E_{S,0}$ are the active, idle, and sleep state energy consumption totals of the buffer disk. $E_{TR,0}$ is the energy overhead for power state transitions. In what follows, we direct our attention to the analytical formulas of $E_{A,0}$, $E_{I,0}$, and $E_{S,0}$. Given a set D of accessed data blocks, we model energy $E_{A,0}$ of the buffer disk when it is active as

$$\begin{aligned} E_{A,0}(D) &= \sum_{i=1}^{m}\sum_{k=1}^{q}(z_{k,i} \cdot P_{A,0} \cdot T_{A,0}) \\ &= \sum_{i=1}^{m}\sum_{k=1}^{q}\left(z_{k,i} \cdot P_{A,0} \cdot \sum_{r_j \in R_{k,i}}\left(t_{SK,j,0} + t_{RT,j,0} + \frac{s_{k,i}}{B_{R,0}}\right)\right), \end{aligned} \tag{31}$$

where $T_{A,0}$ is the time period when the buffer disk is in the active state. $T_{A,0}$ is the accumulated service times of requests processed by the buffer disk.

Let $IS = (t_{I,1}, t_{I,2}, \ldots, t_{I,N0})$ be a sequence of idle periods in the buffer disk. Eq. (32) quantifies energy consumption $E_{I,0}$ of the buffer disk when it is sitting idle.

$$E_{I,0}(IS) = P_{I,0} \cdot \hat{T}_{I,0} = P_{I,0} \cdot \sum_{t_{I,k} \in IS}(y_{k,0} \cdot t_{I,k}), \tag{32}$$

where $\hat{T}_{I,0}$ is the summation of small idle time intervals that are unable to compensate the cost of transitioning to the sleep state. $y_{k,i}$ is a step function used in Eq. (22).

Energy consumption $E_{S,0}$ in Eq. (30) is expressed as

$$\begin{aligned} E_{S,0}(IS) &= P_{S,0} \cdot T_{S,0} \\ &= P_{S,0} \cdot \sum_{t_{I,k} \in IS} (\bar{y}_{k,0} \cdot (t_{I,k} - T_{BE,0})), \end{aligned} \tag{33}$$

where $T_{S,0}$ is the total time when the buffer disk is in the sleep mode. $T_{S,0}$ is derived from the break-even time given by Eq. (21) and the step function is used in Eq. (23). The energy overhead $E_{TR,0}$ for power state transitions is expressed as follows:

$$\begin{aligned} E_{TR,0}(IS) &= P_{TR,0} \cdot T_{TR,0} \\ &= P_{TR,0} \cdot \sum_{t_{I,k} \in IS} (\bar{y}_{k,0} \cdot T_{BE,i}). \end{aligned} \tag{34}$$

Energy consumption E_D of the data disks with the dynamic power management technique can be determined by applying Eq. (25).

Now we are in a position to obtain the energy consumption total of the parallel I/O system, $E_{PRE-BUD}$, with an extra buffer disk from Eqs. (25), (29), and (30). Thus,

$$E_{PRE-BUD} = E_{PF}(P, D) + E_0(D, IS) + E_D(P, R). \tag{35}$$

5.4. Derivation of Response Time for PRE-BUD

Now we are in a position to derive the response time approximation of the PRE-BUD architecture. By definition, the response time of a disk request is the interval between its arrival time and finish time. The response time can be calculated as a sum of a disk requests wait time and I/O service time. Let $D_0 = \{d_1, \ldots, d_k, \ldots, d_{l0}\}$ be a set of data blocks prefetched to a buffer disk. Throughout this section, the subscript 0 is used to represent the buffer disk. Let λ_k and t_k represent the access rate and I/O service time of the kth data block in D_0. Let ρ_0 and Λ_0 be the utilization and aggregate utilization of the buffer disk. Thus, we have

$$\rho_0 = \sum_{d_k \in D_0} (\lambda_k \cdot t_k) \textbf{ and } \Lambda_0 = \sum_{d_k \in D_0} \lambda_k. \tag{36}$$

The mean service time $\bar{S}_0$ and mean-square service time $\bar{S}_0^2$ of disk accesses to the buffer disk are given as

$$\bar{S}_0 = \sum_{d_0 \in D_0} \left(\frac{\lambda_k}{\Lambda_0} \cdot t_k \right) = \frac{1}{\Lambda_0} \cdot \sum_{d_0 \in D_0} (\lambda_k \cdot t_k), \tag{37}$$

$$\bar{S}_0^2 = \sum_{d_0 \in D_0} \left(\frac{\lambda_k}{\Lambda_0} \cdot t_k^2 \right) = \frac{1}{\Lambda_0} \cdot \sum_{d_0 \in D_0} \left(\lambda_k \cdot t_k^2 \right), \tag{38}$$

where λ_k / Λ_0 is the probability of access to data block d_k in the buffer disk.

We model each disk in a parallel disk system as a single M/G/1 queue, which has exponentially distributed interarrival times and an arbitrary distribution for service times of disk requests. Consequently, we can obtain the mean response time $\bar{T}_0$ of accesses to the buffer disk from Eqs. (36), (37), and (38) as

$$\bar{T}_0 = \bar{S}_0 + \frac{\Lambda_0 \cdot \bar{S}_0^2}{2 \cdot (1 - \rho_0)}. \tag{39}$$

In what follows, let us derive mean response time $\bar{T}_j$ of accesses to disk j. We denote $D_j(1 \leq j \leq m)$ as a set of data blocks stored in the jth disk. Let $D_j^{PF} \subseteq D_j(1 \leq j \leq m)$ be a set of data blocks in D_j prefetched to a buffer disk. Similarly, let $D'_j \subseteq D_j(1 \leq j \leq m)$ be a set of data blocks that has not been prefetched. For the jth disk, we have $D_j = D_j^{PF} \cup D'_j$. Let ρ_j and Λ_j represent the utilization and aggregate utilization of the buffer disk. ρ_j and Λ_j can be expressed as:

$$\rho_j = \sum_{d_k \in D'_j} (\lambda_k \cdot t_k) \text{ and } \Lambda_j = \sum_{d_k \in D'_j} \lambda_k. \tag{40}$$

The mean and mean-square service times (i.e., $\bar{S}_j$ and $\bar{S}_j^2$) of disk accesses to disk j are given as

$$\bar{S}_j = \sum_{d_0 \in D'_j} \left(\frac{\lambda_k}{\Lambda_j} \cdot t_k \right) = \frac{1}{\Lambda_j} \cdot \sum_{d_0 \in D'_j} (\lambda_k \cdot t_k) \tag{41}$$

$$\bar{S}_j^2 = \sum_{d_0 \in D'_j} \left(\frac{\lambda_k}{\Lambda_j} \cdot t_k^2 \right) = \frac{1}{\Lambda_j} \cdot \sum_{d_0 \in D'_j} \left(\lambda_k \cdot t_k^2 \right). \tag{42}$$

We can derive the mean response time $\bar{T}_j$ of accesses to data disk j from these equations as

$$\bar{T}_j = \bar{S}_j + \frac{\Lambda_j \cdot \bar{S}_j^2}{2 \cdot (1 - \rho_j)} \tag{43}$$

Therefore, the overall mean response time of a parallel disk system with a buffer disk is written as follows, where $\Lambda = \sum_{j=0}^{m} \Lambda_j$ is the aggregate access rate of the parallel disk system.

$$\bar{T} = \frac{1}{\Lambda} \cdot \sum_{j=0}^{m} (\Lambda_j \cdot \bar{T}_j) \tag{44}$$

6. EXPERIMENTAL RESULTS

In this section, we present our experimental results for the proposed PRE-BUD energy efficient prefetching approach for parallel disk systems. First, we provide information about our simulation environment and parameters that were varied for our experiments. Next, we compare PRE-BUD with PDC and DPM—two well-known energy conservation techniques for parallel disks [Zhu et al. 2005]. Then, we study the impacts of various system parameters on energy efficiency and the performance of parallel disks.

6.1. Experiment Setup

Extensive experiments were conducted with a disk simulator based on the mathematical models presented in Sections 4 and 5. Our disk model (see Table III) is based on the IBM Ultrastar 36Z15, which has been widely used in data-intensive environments [Kwan et al. 1995]. Our simulator was implemented in JAVA, allowing us to quickly and easily change various system parameters. Both synthetic and real-world traces are used to evaluate PRE-BUD.

For comparison purposes, we consider a parallel I/O system (referred to as Non-Energy Aware) where disks are operating in a standard mode without employing any energy-saving techniques. In other words, disks are in the busy state while serving

Table III. Disk Parameters (IBM Ultrastar 36Z15)

Parameter	Value	Parameter	Value
Transfer Rate	55 MB/S	Spin Down Time:T_D	1.5 s
Active Power:P_A	13.5 W	Spin Up Time:T_U	10.9 s
Idle Power:P_I	10.2 W	Spin Down Energy:E_D	13.0 J
Standby Power:P_S	2.5 W	Spin Up Energy:E_U	135 J

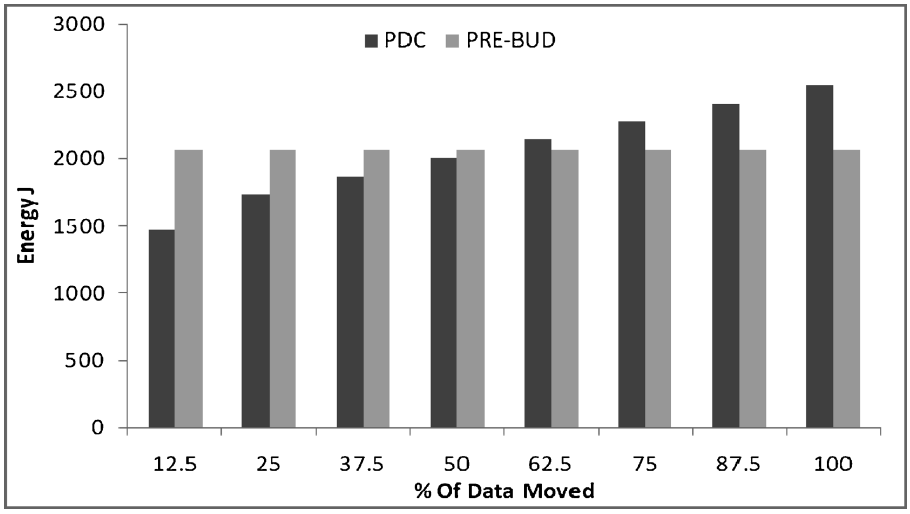

Fig. 5. PDC and PRE-BUD comparison.

requests, and are in the idle state when not serving a request. Two PRE-BUD configurations are evaluated; the first configuration PRE-BUD1 adds an extra disk to be used as the buffer disk and the second configuration called PRE-BUD2 designates an existing disk as the buffer disk. Recall from the motivational example that PRE-BUD1 requires the extra energy overhead of not only prefetching the data, but also to operate the buffer disk. This can lead to certain situations where PRE-BUD1 can consume more energy than non-energy aware approaches and our experimental results demonstrate this fact. Note that the term "hit rate" used throughout this section is defined as the percentage of requests that can be served by the buffer disk. One of the goals of our experiments is to identify the parameters that are crucial to energy efficient disk storage systems.

6.2. Comparison of PRE-BUD and PDC

Figure 5 shows the energy efficiency comparison results of our PRE-BUD strategy and the PDC [Pinheiro and Bianchini 2004] energy saving technique. PDC attempts to move popular data across the disks, such that the first disk has the most popular data, while the second disk has the second most popular set of data and so forth. We fixed the data size to be 275 MB and the hit rate is 95% for PRE-BUD. Since the data could potentially be anywhere in the disk system, the PDC strategy causes data to be moved within the disk system.

Figure 5 shows that PRE-BUD is more energy efficient than PDC if PDC has to move a large amount of data within the storage system. PDC may have a much higher initial energy penalty when a large amount of data must be moved within the storage system. PRE-BUD has a fixed amount of buffer disk space; for this example, it is fixed at 10% of the total data in the storage system. PRE-BUD can be adaptively tuned to

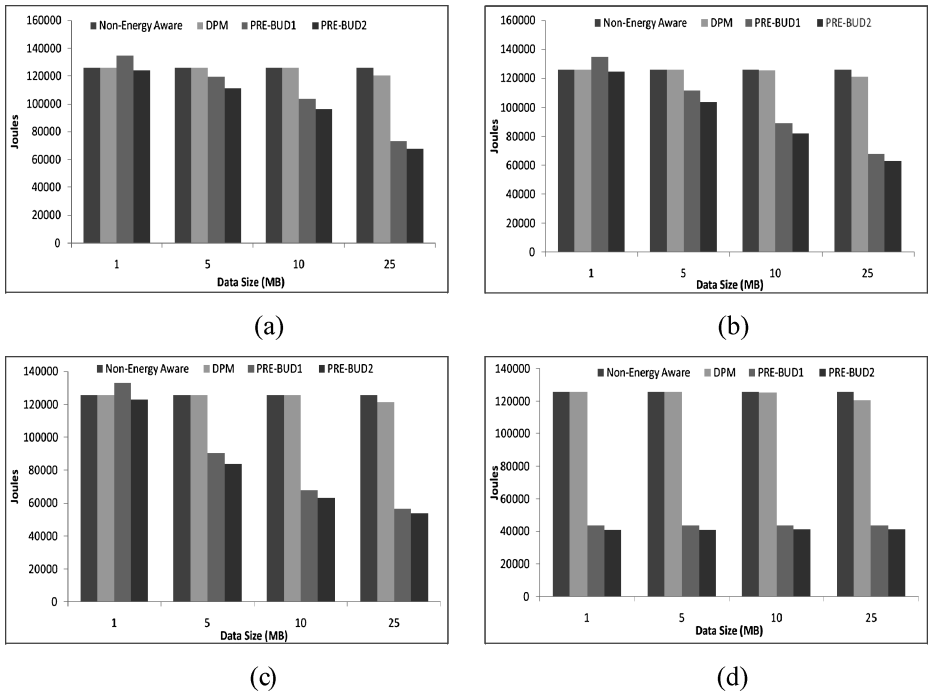

Fig. 6. Total energy consumption of disk system while data size is varied for four different values of the hit rate: (a) 85%, (b) 90%, (c) 95%, and (d) 100%.

find the particular amount of buffer disk capacity that will yield the largest amount of savings. PRE-BUD only needs to move blocks that can provide energy savings. In contrast, PDC makes no guarantees about the energy impact of moving data within the storage system. PDC does not adapt as quickly as our PRE-BUD strategy to changing workload conditions. The lookahead window we employ can amortize the expense of moving frequently accessed data into the buffer disk. PDC attempts to move frequently accessed data at one time, which can cause large overheads when the workload of the parallel disk system changes frequently.

6.3. Impact of Data Size

The second set of experiments focused on evaluating the impact that the data size of the requests has on the energy savings of DPM and PRE-BUD. For these set of experiments we fixed the number of disks at 12. The hit rate of the buffer disk is varied from 85% to 100%. Figure 6 reveals that the data size has a huge impact on the energy efficiency of DPM and our PRE-BUD strategy when the hit rate is lower than 100%. If the hit rate is 100% for the buffer disk, data disks can sleep for a long period of time regardless of the data size.

The results depicted in Figure 6 indicate that our PRE-BUD strategy performs best with data-intensive applications that request large files. Thus, multimedia storage systems would be a perfect candidate for the PRE-BUD energy saving strategy. The data size has such a large impact on energy savings because of the break even time, TBE, which is 14.5 seconds for the chosen disk model. Large data sizes take a longer time to serve; consecutive buffer hits for a large data size meet the break-even time. Conversely, small data sizes produce little or no energy efficiency gains. These experimental results confirm that the data size together with the hit rate combine to produce

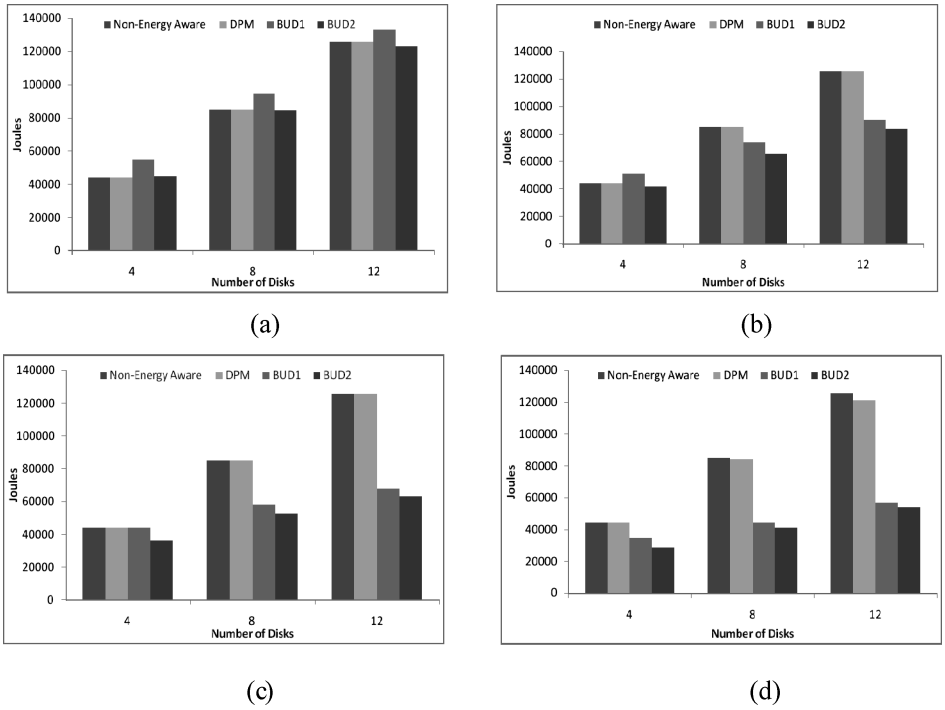

Fig. 7. Total energy consumption of disk system while the number of data disks is varied. Data size is fixed at: (a) 1MB, (b) 5MB, (c) 10MB, and (d) 25MB.

a probability of meeting TBE, which is the break-even time, with higher hit rates and large data sizes being the ideal combination for energy savings. PRE-BUD1 consumes more energy than DPM when the data size is 1MB or smaller. This is because PRE-BUD1 adds an extra disk to the disk system, and with a small data size energy efficient opportunities to put a disk to sleep are rare. This set of experiments leads us to the conclusion that large data sizes are conducive to energy efficiency in PRE-BUD.

6.4. Impact of Number of Data Disks

Now we evaluate the impact of varying the ratio of data disks to buffer disks. The number of buffer disks is fixed at 1; the number of data disks is set to 4, 8, and 12. The hit rate is fixed at 95% and the data size is varied from 1MB to 25MB. Not surprisingly, we discover from Figure 7 that as we increase the number of data disks per buffer disk, the energy savings becomes more pronounced for PRE-BUD. This energy efficiency trend is expected because increasing the number of disks makes each individual disk less heavily loaded.

The buffer disk simply prefetches blocks that can produce energy savings; lightly loaded disks are more likely to be switched into the standby mode to conserve energy. PRE-BUD, of course, has to prefetch a smaller amount of data from each disk to achieve this high energy efficiency. If the number of data disks is increased, we must be sure that the performance is not negatively impacted. When more data disks are added into a parallel disk system, the buffer disk is more likely to become the performance bottleneck. Moreover, Figure 7 shows that a large data size makes PRE-BUD more energy efficient. This result is consistent with that plotted in Figure 6.

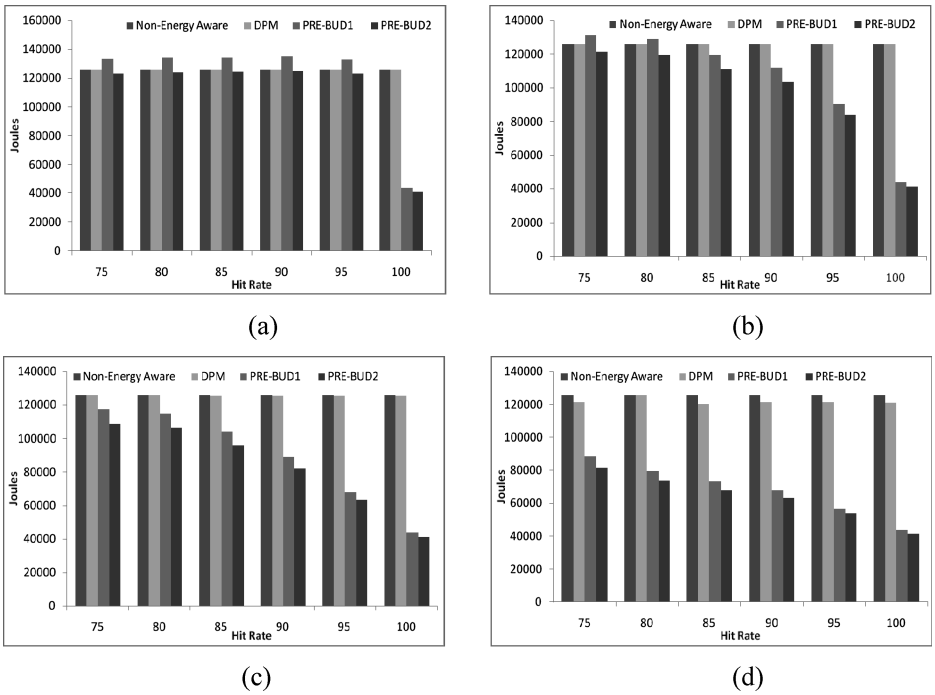

Fig. 8. Total energy consumption for different hit rate values where the data size is fixed at: (a) 1MB, (b) 5MB, (c) 10MB, and (d) 25MB.

6.5. Impact of Hit Rate

In this set of experiments, we chose to investigate the impact the buffer disk hit rate has on the energy efficiency of the parallel disk system. Again, the data size is varied from 1 to 25MB. The number of data disks is set to 12. We observe from Figure 8 that higher hit rates enable PRE-BUD to save more energy in the parallel disk system. This is expected because with a high hit rate, we heavily load the buffer disk while allowing data disks to be transitioned to the standby state. A low hit rate means a data disk must be frequently spun up to serve requests, incurring energy penalties. The longer a disk can stay in the standby state, the more energy efficient a parallel disk will be. Note that hit rates of 100% are not realistically achievable if the disk requests require all disks to be active. A 100% hit rate can only be accomplished if the overall load on the entire disk system is fairly light. It has been documented that some parallel workloads are heavily skewed towards a small percentage of the workload, thereby making 80% hit rates feasible. With the varying data sizes, we notice that the energy savings becomes more significant for larger data sizes. Having a larger data size is similar to increasing the hit rate of buffer disks operating on smaller data sizes.

6.6. Impact of Interarrival Delays

In these experiments, we study the impact that the inter-arrival rates of the requests have on the energy savings of PRE-BUD. Figure 9 shows the energy consumption totals of the disk system with four different values of the interarrival delay. The number of disks was fixed at 12, the data size was fixed at 1MB, and the hit rate was varied from 85% to 100%. When there is no interarrival delay, DPM will not yield any energy savings. This is because there are no idle-windows large enough for disks to spin down. PRE-BUD1 ends up consuming more energy than DPM in this case,

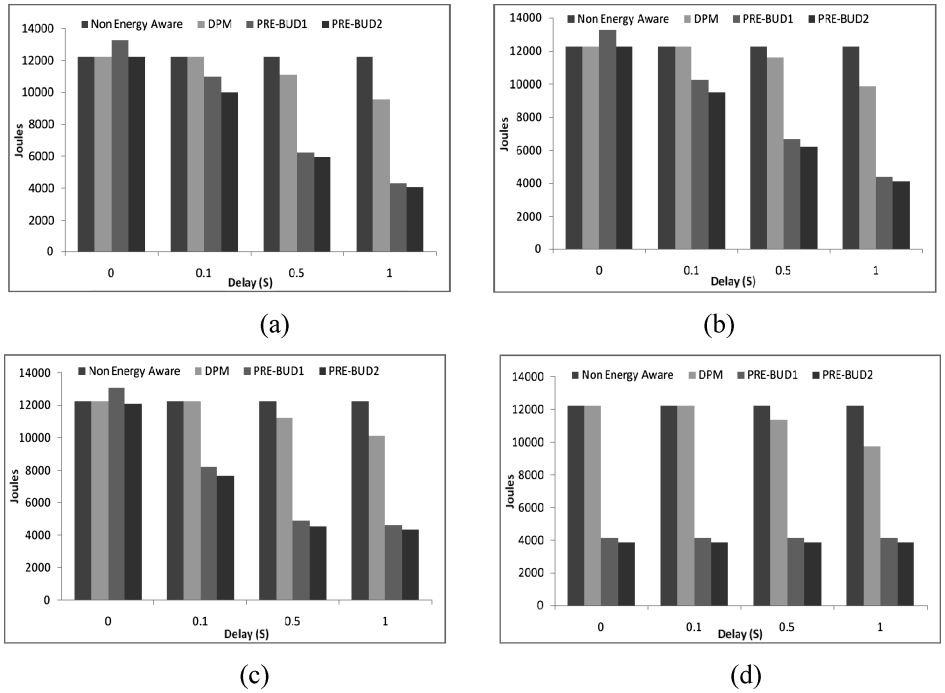

Fig. 9. Total energy consumption for different delay values where the hit rate is (a) 85%, (b) 90%, (c) 95%, and (d) 100%.

because PRE-BUD1 adds the overhead of an extra disk and the energy required to prefetch the data. PRE-BUD2 is the most energy efficient, since there is no need to add an extra disk. If the interarrival delay is 100 ms, we have a similar situation, except that PRE-BUD1 is now able to produce a small amount of energy savings.

When the interarrival delay becomes 500 ms, DPM begins to produce energy savings. However, such energy savings pales in comparison to PRE-BUD. When the delay is increased to 1 Sec., the results look similar to the results for a 500 ms delay. Although DPM in this case can result in more energy savings, PRE-BUD1 and PRE-BUD2 significantly outperform DPM in terms of energy efficiency. These results fit our intuition about the behavior of the PRE-BUD approach. DPM needs large idle times between consecutive requests to achieve energy savings, heavily depending on the break even time of a particular hard drive. PRE-BUD is more energy efficient than DPM, because PRE-BUD proactively provides data disks with larger idle windows by redirecting requests to the buffer disk.

6.7. Power State Transitions

In this section of our article, we investigate the relationship between the number of power state transitions and energy efficiency. Figure 10 depicts the number of power state transitions triggered by DPM, PRE-BUD1, and PRE-BUD2 when the data size and hit rate are varied. The number of state transitions caused by DPM is zero when the data size is smaller than or equal to 25MB. There is no power state transitions for small data sizes, because no idle time periods of data disks are long enough for DPM to justify transitioning to the standby state. When the data size is larger than 25MB, the number of power state transitions quickly rises with increasing data sizes. If DPM triggers transitions, it is able to improve the energy efficiency of the disk system.

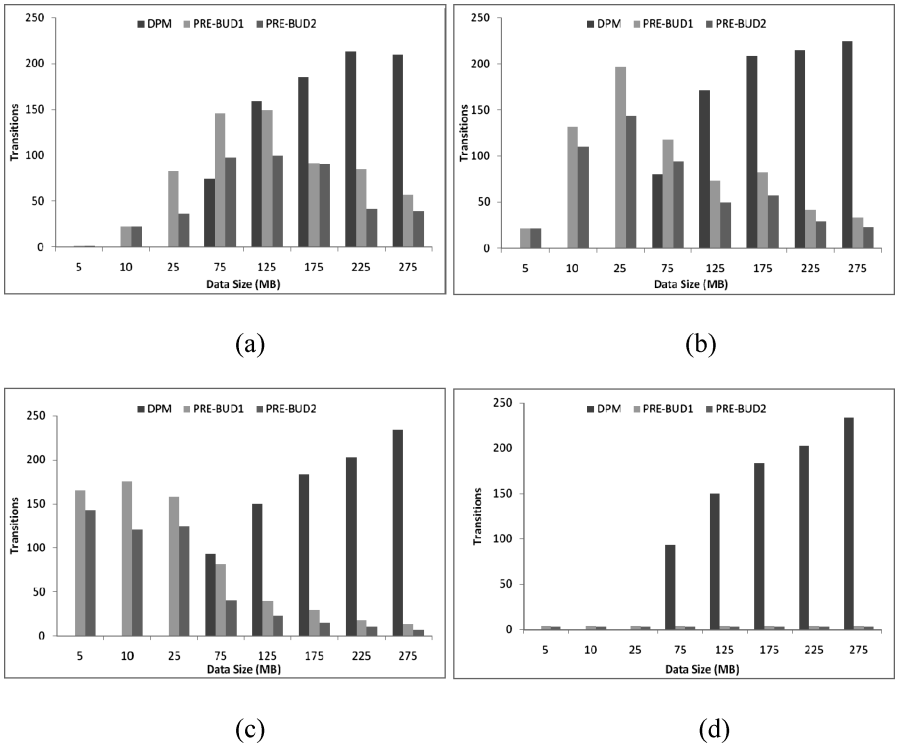

Fig. 10. Total disk state transitions for different data sizes where the hit rate is: (a) 85%, (b) 90%, (c) 95%, and (d) 100%.

Interestingly, the number of transitions for PRE-BUD slowly increases at first and then starts dropping when the data size is larger than 125MB. The transition number increases when data size is small because many small idle periods in data disks are merged by PRE-BUD creating new opportunities for data disks to sleep. Since the small idle intervals tend to be spread out data disks experience many power state transitions. The buffer disk reduces the number of transitions for data disks when the data size is large, because a buffer disk generates larger and fewer idle time periods in data disks. A few very large idle time periods lead to a small number of transitions.

One of the problems with DPM is that it will transition a disk many times, which may decrease the reliability of the disk. Unlike DPM, PRE-BUD can improve the reliability of the disk system by lowering the number of transitions when data sizes of requests are very large. As such, PRE-BUD is conducive to improving both energy efficiency and reliability for data-intensive applications with large data requests.

6.8. Impact of Disk Power Characteristics

To examine the effect that manipulating disk power characteristics has on PRE-BUD, we varied the active power, idle power, and standby power, for three separate experiments, respectively. The number of data disks is fixed at 4 and the data size is 25MB.

Figure 11(a) shows that for all the four schemes, increasing the active power of a disk results in a continuous increase of energy consumption across the four different strategies. Results plotted in Figure 11(a) indicate that PRE-BUD is more energy efficient for parallel disks with low active power. For example, if the active power is 9.5W, PRE-BUD2 saves 15.1% of the energy consumption total over DPM. If the active

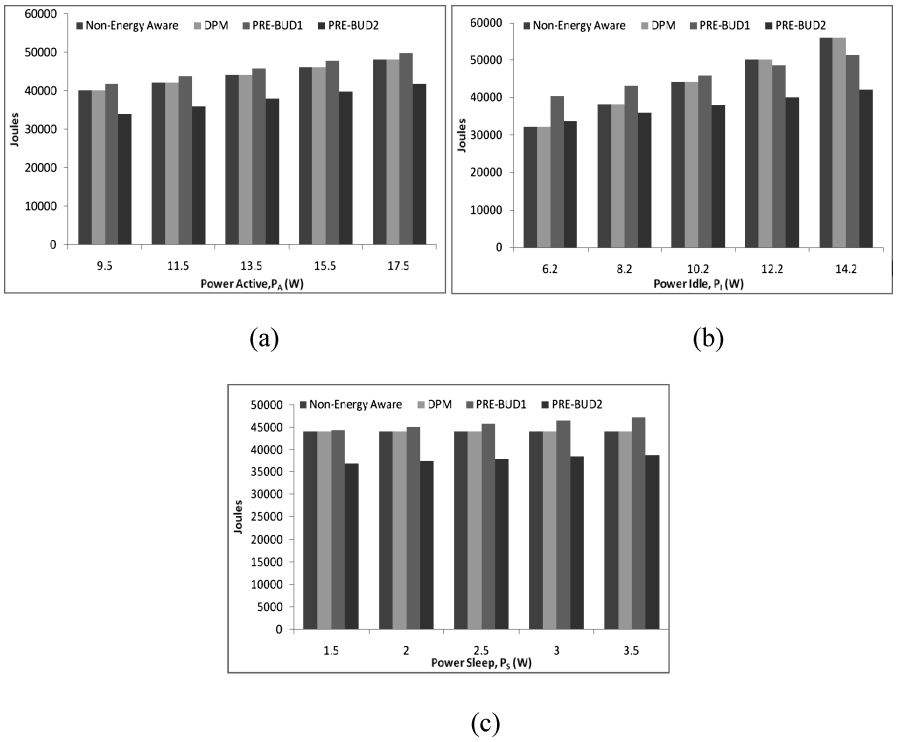

Fig. 11. Total energy consumption for various values of the following disk parameters: (a) power active, (b) power idle, and (c) power standby.

power is increased to 17.5W, then PRE-BUD2 improves energy efficiency over DPM by only 13.0%. Figure 11(b) shows the impact of varying the idle power parameter of a disk has on the energy efficiency of PRE-BUD. Compared with active power, idle power has a greater impact on the energy savings achieved by PRE-BUD. If the idle power is very low, PRE-BUD2 has a negative impact. If the idle power is increased to 14.2 W, PRE-BUD2 can save energy over DPM by 25%. Figure 11(c) shows that standby power also has a significant impact on PRE-BUD. Specifically, the energy savings starts at 16.3% and drops to 11.7% with increasing standby power.

These results, illustrated in Figure 11, indicate that parallel disks with low active power, high idle power, and low standby power can produce the best energy-saving benefit. This is because PRE-BUD allows disks to be spun down in standby during times they would be idle using DPM. The greater the discrepancy between idle and standby power, the more beneficial PRE-BUD becomes. Lowering active power also makes PRE-BUD more energy efficient because the amount of energy consumed prefetching and serving requests can be reduced.

Throughout our experiments, it was realized that the main factor limiting the energy savings potential of PRE-BUD is the large break-even times of disks. A large break-even time of a disk reduces opportunities for DPM to conserve energy if there are a large number of idle periods that are smaller than the break-even time. PRE-BUD alleviates this problem of DPM by combining idle periods to form large idle windows. Unfortunately, PRE-BUD inevitably reaches a critical point where energy savings are no longer possible. To further improve energy efficiency of PRE-BUD, we have to rely

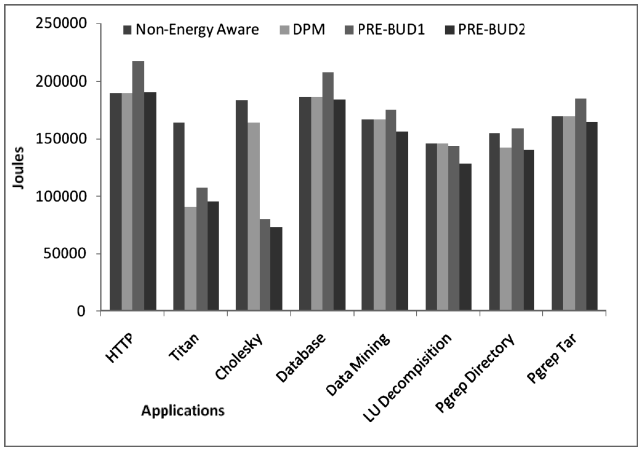

Fig. 12. Total energy consumed for real world traces.

on disks that are able to quickly transition among power states—one of the dominating factors in energy savings for disks.

6.9. Real-World Applications

To validate our results based on synthetic traces, we evaluated eight real-world application traces. The applications are parallel in nature; thus, all of the applications used eight disks, with the Titan and HTTP application being the exceptions and only used seven disks. Note that results plotted in Figure 12 generally represented the worst case for PRE-BUD. Figure 12 shows that PRE-BUD1 consumes more energy than DPM for most applications except for the Cholesky and LU Decomposition applications. When applications are very I/O-intensive, adding an extra disk leaves no opportunity to conserve energy. Figure 12 also shows PRE-BUD2 noticeably improves energy efficiency over DPM for most applications. The results confirm that PRE-BUD can generally produce energy savings under both low and high disk workloads, even though the energy savings is relatively small for high workloads.

A surprising exception is the Titan application, because DPM is more energy efficient than PRE-BUD. In the Titan trace, there is one large gap between all of the consecutive requests, allowing DPM an opportunity to put all of the disks into the standby state for a long period of time. PRE-BUD, on the other hand, keeps the buffer disk active all the time to minimize the negative impact on performance. In this special case, the active buffer disk makes PRE-BUD less energy efficient than DPM. The energy efficiency of PRE-BUD can be further improved by aggressively transitioning the buffer disk to the standby state if it is sitting idle. This situation leads us to believe that a prediction-based algorithm may be necessary to determine situations where DPM may be more energy-efficient than prefetching strategies. We reserve this for future work because prediction-based schemes require a detailed analysis of workloads and the prediction mechanism.

6.10. Response Time Analysis

In Table IV, we present our response time analysis results for the PRE-BUD strategy. We used four different traces, which had a designated set of popular data that varied in size and overall percentage of the entire trace. We also varied the number of data disks that each buffer disk is responsible for prefetching data from. From the table, we see that the first three traces have similar response time results for each number

Table IV. Response Time Analysis

PRE-BUD Response Time Degradation	5 Disks	10 Disks	15 Disks	20 Disks
10% of Data Accessed in 90% of Trace	6 ms	16 ms	26 ms	36 ms
20% of Data Accessed in 80% of Trace	6 ms	16 ms	26 ms	36 ms
30% of Data Accessed in 70% of Trace	6 ms	16 ms	26 ms	36 ms
40% of Data Accessed in 60% of Trace	32 ms	47 ms	62 ms	79 ms

of data disks used for the experiments. This tells us that, our PRE-BUD strategy is capable of balancing the load and producing energy savings with a minimal impact on the response time of the parallel disk system. For the last trace, in which 40% of the data is accessed 60% in the trace, we see that our response time degradation is significantly higher when compared to the other traces. This result is expected because the workload does not have an easily identifiable subset of data that can be prefetched to produce energy savings. PRE-BUD relies on the fact that some parallel application I/O operations are heavily skewed towards a small subset of data. From all of the results presented in Table IV, we realize that the PRE-BUD strategy produces relatively small response-time degradations. This means our strategy will work for applications that can tolerate response degradations and is not suitable for real-time applications.

7. CONCLUSION

The use of large-scale parallel I/O systems continues to rise as the demand for information systems with large capacities grows. Parallel disk I/O systems combine smaller disks to achieve large capacities. A challenging problem is that large-scale disk systems can be extremely energy inefficient. The energy consumption rates are rising as disks become faster and disk systems are scaled up. The goal of this study is to improve the energy efficiency of a parallel I/O system using a buffer disk to which frequently accessed data are prefetched.

In this article, we develop an energy-efficient prefetching algorithm (PRE-BUD) for parallel I/O systems with buffer disks. Two buffer disk configurations considered in our study are (1) adding an extra buffer disk to accommodate prefetched data and (2) utilizing an existing disk as the buffer disk. Prefetching data blocks in the buffer disk provides ample opportunities to increase idle periods in data disks, thereby facilitating long standby times of disks. Although the first buffer disk configuration may consume more energy due to the energy overhead introduced by an extra disk, it does not compromise the capacity of the disk system. The second buffer disk configuration lowers the capacity of the parallel disk system, but it is more cost-effective and energy-efficient than the first one. Compared with existing energy saving strategies for parallel I/O systems, PRE-BUD exhibits the following appealing features: (1) it is conducive to achieving substantial energy savings for both large and small read requests, (2) it is able to positively impact the reliability of parallel disk systems by the virtue of reducing the number of power state transitions, (3) it prefetches data into a buffer disk without affecting the data layout of any data disks, (4) it does not require any changes to be made to the overall architecture of an existing parallel I/O system, and (5) it does not involve complicated metadata management for large-scale parallel I/O systems.

There are four possible future research directions we have identified for the PRE-BUD strategies. First, we will improve the scalability of PRE-BUD by adding more than one buffer disk to the parallel I/O system. This can be implemented by considering a buffer disk controller that manages various buffer disks each responsible for a set of data disks. In this work, we investigate the relationship between buffer disks and data disks, to improve the parallelism of PRE-BUD we need to investigate the relationship between a buffer disk controller and the buffer disks. The number of buffer disks will have to be increased as the scale of the disk system is increased. Second, PRE-BUD will

be integrated with the dynamic speed control or DRPM [Gurumurthi et al. 2003] for parallel disks. Third, we will quantitatively study the reliability impacts of PRE-BUD on parallel I/O systems. Last, we plan to investigate a prediction-based algorithm to dynamically determine when to use DPM or our PRE-BUD strategies to produce the largest energy-efficiency gain.

REFERENCES

BENINI, L., BOGLIOLO, A., AND DE MICHELI, G. 2000. A survey of design techniques for system-level dynamic power management. *IEEE Trans. VLSI Syst. 8*, 299–316.

CARRERA, E. V., PINHEIRO, E., AND BIANCHINI, R. 2003. Conserving disk energy in network servers. In *Proceedings of the 17th International Conference on Supercomputing*. 86–97.

CHEN, F., JIANG, S., SHI, W., AND YU, W. 2007. Flexfetch: A history-aware scheme for I/O energy saving in mobile computing. In *Proceedings of the International Conference on Parallel Processing (ICPP'07)*. IEEE Computer Society, 10.

COLARELLI, D. AND GRUNWALD, D. 2002. Massive arrays of idle disks for storage archives. In *Proceedings of the ACM/IEEE Conference on Supercomputing (Supercomputing'02)*. IEEE Computer Society Press, Los Alamitos, CA, 1–11.

DOUGLIS, F., KRISHNAN, P., AND MARSH, B. 1994. Thwarting the power-hungry disk. In *Proceedings of the Winter USENIX Conference*. 293–306.

EOM, H. AND HOLLINGSWORTH, J. K. 2000. Speed vs. accuracy in simulation for I/O-intensive applications. In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE Computer Society Press, 315–322.

GURUMURTHI, S., SIVASUBRAMANIAM, A., KANDEMIR, M., AND FRANKE, H. 2003. DRPM: Dynamic speed control for power management in server class disks. In *Proceedings of the 30th Annual International Symposium on Computer Architecture*. 169–179.

HAWKINS, J., AND BODN, M. 2005. The applicability of recurrent neural networks for biological sequence analysis. *IEEE/ACM Trans. Computat. Biol. Bioinf. 2*, 243–253.

HELMBOLD, D., LONG, D., SCONYERS, T., AND SHERROD, B. 1998. Adaptive disk spin-down for mobile computers. *Mobile Netw. Appl. 5*, 2000.

JONES, E. 2006. EPA announces new computer efficiency requirements. EPA Anouncement.

KALLAHALLA, M. AND VARMAN, P. J. 2002. PC-OPT: Optimal offline prefetching and caching for parallel I/O systems. *IEEE Trans. Comput. 51,* 11, 1333–1344.

KIM, Y.-J., KWON, K.-T., AND KIM, J. 2007. Energy-efficient disk replacement and file placement techniques for mobile systems with hard disks. In *Proceedings of the ACM Symposium on Applied Computing (SAC'07)*. ACM, New York, 693–698.

KRISHNAN, P., LONG, P. M., AND VITTER, J. S. 1995. Adaptive disk spindown via optimal rent-to-buy in probabilistic environments. Tech. rep., Duke University.

KWAN, T. T., MCGRATH, R. E., AND REED, D. A. 1995. NCSA's world wide web server: Design and performance. *IEEE Comput. 28*, 68–74.

LI, K., KUMPF, R., HORTON, P., AND ANDERSON, T. 1994. A quantitative analysis of disk drive power management in portable computers. In *Proceedings of the Winter USENIX Conference*. 279–291.

MAXIMUM THROUGHPUT, INC. 2002. Power, heat, and sledgehammer. White Paper.

MOORE B. 2002. Taking the data center power and cooling challenge. *Energy User News*.

PAPATHANASIOU, A. E. AND SCOTT, M. L. 2004. Energy efficient prefetching and caching. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference (ATEC'04)*. USENIX Association, Berkeley, CA, USA, 22–22.

PINHEIRO, E. AND BIANCHINI, R. 2004. Energy conservation techniques for disk array-based servers. In *Proceedings of the 18th Annual International Conference on Supercomputing (ICS'04)*. ACM, New York, 68–78.

SHEN, H., KUMAR, M., DAS, S., AND WANG, Z. 2004. Energy-efficient caching and prefetching with data consistency in mobile distributed systems. In *Proceedings of the 18th International Symposium on Parallel and Distributed Processing*. 67.

SON, S. W. AND KANDEMIR, M. T. 2006. Energy-aware data prefetching for multi-speed disks. In *Proceedings of the Conference on Computing Frontiers*. 105–114.

TRIZNA, D. 2005. Microwave and hf multi-frequency radars for dual-use coastal remote sensing applications. In *Proceedings of MTS/IEEE (OCEANS'05). 1*, 532–537.

WANG, J., ZHU, H., AND LI, D. 2008. ERAID: Conserving energy in conventional disk-based raid system. *IEEE Trans. Comput. 57,* 3, 359–374.

XIE, T. 2008. SEA: A striping-based energy-aware strategy for data placement in raid-structured storage systems. *IEEE Trans. Comput. 57,* 6, 748–761.

ZHU, Q., CHEN, Z., TAN, L., ZHOU, Y., KEETON, K., AND WILKES, J. 2005. Hibernator: Helping disk arrays sleep through the winter. *SIGOPS Oper. Syst. Rev. 39,* 5, 177–190.

ZHU, Q., DAVID, F. M., DEVARAJ, C. F., LI, Z., ZHOU, Y., AND CAO, P. 2004a. Reducing energy consumption of disk storage using power-aware cache management. In *Proceedings of the 10th International Symposium on High Performance Computer Architecture*. IEEE Computer Society, 118.

ZHU, Q., SHANKAR, A., AND ZHOU, Y. 2004b. PB-LRU: A self-tuning power aware storage cache replacement algorithm for conserving disk energy. In *Proceedings of the 18th Annual International Conference on Supercomputing (ICS'04)*. ACM, New York, 79–88.

ZHUANG, X. AND PANDE, S. 2004. Power-efficient prefetching via bit-differential offset assignment on embedded processors. In *Proceedings of the ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES'04)*. ACM, New York, 67–77.

ZONG, Z., BRIGGS, M., O'CONNOR, N., AND QIN, X. 2007. An energy-efficient framework for large-scale parallel storage systems. In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDS'07)*. 1–7.

Received April 2010; revised October 2010; accepted November 2010