Energy-Aware Prefetching for Parallel Disk Systems

Algorithms, Models, and Evaluation

Adam Manzanres[†], Xiaojun Ruan[†], Shu Yin[†], Mais Nijim[‡], WeiLuo[¥], and Xiao Qin[†]
[†]Department of Computer Science and Software Engineering
Auburn University
Auburn, AL USA
{acm0008, xzr0001, szy0004, xqin}@auburn.edu

[‡]School of Computing
University of Southern Mississisippi
Hattiesburg, MS USA
mais.nijim@usm.edu

[¥]China Ship Development and Design Center
Wuhan, Hubei China
free_xingezi@163.com

Abstract— **Parallel disk systems consume a significant amount of energy due to the large number of disks. To design economically attractive and environmentally friendly parallel disk systems, in this paper we design and evaluate an energy-aware prefetching strategy for parallel disk systems consisting of a small number of buffer disks and large number of data disks. Using buffer disks to temporarily handle requests for data disks, we can keep data disks in the low-power mode as long as possible. Our prefetching algorithm aims to group many small idle periods in data disks to form large idle periods, which in turn allow data disks to remain in the standby state to save energy. To achieve this goal, we utilize buffer disks to aggressively fetch popular data from regular data disks into buffer disks, thereby putting data disks into the standby state for longer time intervals. A centrepiece in the prefetcing mechanism is an energy-saving prediction model, based on which we implement the energy-saving calculation module that is invoked in the prefetching algorithm. We quantitatively compare our energy-aware prefetching mechanism against existing solutions, including the dynamic power management strategy. Experimental results confirm that the buffer-disk-based prefetching can significantly reduce energy consumption in parallel disk systems by up to 50 percent. In addition, we systematically investigate the energy efficiency impact that varying disk power parameters has on our prefetching algorithm.**

Keywords-storage systems; energy-efficiency;prefetching

I. INTRODUCTION

The number of large-scale parallel disk 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 disk systems include long running simulations [8], remote sensing applications [20] and biological sequence analysis [10], to name just a few. As the size of a parallel disk system grows, the energy consumed by the I/O system often becomes a large part of the total cost of ownership [16]. Reducing the energy costs of operating these large-scale disk I/O systems often becomes one of the most important design issues.

Several techniques proposed to conserve energy in disk systems include dynamic power management schemes [7], power-aware cache management strategies [23], software-directed power management techniques [18], redundancy techniques [15], data placement [16], and multi-speed settings [9]. However, the research on energy-efficient prefetching with buffer disks is still in its infancy. Therefore, it is imperative to develop new prefetching techniques to reduce the energy consumption in parallel disk systems while maintaining high performance.

Existing power management strategies can shorten the life cycle of disks if they are spun up and down too frequently, thereby degrading the reliability of the disks. To remedy this deficiency we propose a novel parallel disk architecture with buffer disks (see [21] for the details of the disk architecture) to reduce the number of power-state transitions of disks. Using buffer disks to temporarily 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. PRE-BUD 1 adds an extra disk performing as a buffer disk, whereas the PRE-BUD 2 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 [14]. 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 potential into buffer disks. To accurately prefetch data blocks, information concerning future disk requests is indispensable. PRE-BUD can deal with both the 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 [12] that can furnish a window of future disk requests. This work is also currently focuses on techniques to improve the energy efficiency of read requests. To implement writes in the BUD architecture data consistency must be addressed, but we reserve this for future work.

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 disk systems with buffer disks. The energy savings measured by the prediction model represents the importance and priority of prefetching blocks in a buffer disk to efficiently conserve energy in disk systems. 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. We also quantitatively compare PRE-BUD with existing techniques employed in parallel disk systems.

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

II. Related Work

A. *Strengths/Limitations of Previous Work*

Almost all energy efficient strategies rely on DPM techniques [1]. 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 [15]. Their techniques relied on changing prefetching and caching strategies within the Linux kernel. PB-LRU is another energy efficient cache management strategy [24]. 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 [4]. Energy efficient caching and prefetching in the context of mobile distributed systems have also been extensively studied [11] **Error! Reference source not found.**. These three research papers focus on mobile disk systems, whereas we focus on large 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.

Multi-speed/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 multi-speed disks [18] [22]. Energy efficient techniques have also relied on replacing high performance disks with low energy disks [2]. Mobile computing systems have also been recognized as platforms where disk energy should be conserved [4][13]. The mobile computing platforms use low power disks with smaller break-even times. The weakness of using multi-speed disks is that there are no commercial multi-speed 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 [6]. 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 [16]. 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 pre-fetch 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 requires metadata for the blocks in the buffer disk.

B. *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 prefetching strategy that tries to move the most popular data into a set of buffer disks without affecting the data layout of any of the data disks.

(2) Our prefetching strategy is unique in the fact that it prefetches blocks that will produce energy savings using information in the look-ahead window. Previous techniques prefetch blocks without consideration of the explicit energy savings the prefetched block can produce. Our strategies also have the added benefit of not requiring any changes to be made to the overall architecture of an existing disk system.

Corresponding Author. xqin@auburn.edu

TABLE I. NOTATION FOR THE DESCRIPTION OF THE PREFETCHING MODEL

Notation	Description
R	Current look-ahead. $r \in R$ is a reference in the look-ahead
block(r)	Block accessed in reference $r \in R$
disk(r)	Disk in which *block(r)* is residing
A	Subset of the look-ahead *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 $A^+ \cup G$

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.

III. ENERGY-EFFICIENT PREFETCHING STRATEGY

A. Prefetching Model

Before presenting the prefetching module of PRE-BUD, we first summarize the notation for the description of the prefetcher in Table 1. Fig. 1 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/Os while placing data disks in the standby state longer; that is, the buffer disk is likely to become a performance bottleneck. To properly address the bottleneck issue, 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

```
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 $T_{Disk}(r) \le T_0(r)$, 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 look-ahead 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 look-ahead, 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                        $\sum_{r' \in G^+} s(r') \le C_0$ /* 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 */
          }
```

Figure 1. The energy-efficient prefetching module.

achieve a shorter response time than 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 Fig. 1). 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 Fig. 1). Thus, block *b* is prefetched in Step 10 only when the following four conditions are met. First, a request *r* in the look-ahead is accessing the block, i.e., $\exists r \in R$: *block(r) = b*. Second, the block is not present in the buffer disk, i.e., $b \notin G$. Third, fetching the blocks and caching them into the buffer disk can improve energy efficiency, i.e., $E_s(b)>0$. Lastly, the block is residing in an active data disk, i.e., $disk(b) \in A$. Note that set A^+ (see Table 1) contains all the blocks that satisfy the above four 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 fetched 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 savings, 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 Fig. 1). 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 look-ahead. 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 until the last possible moment when the above 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 look-ahead 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

B. *Energy-Saving Calculation Model*

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 Fig. 1). The prediction model along with the calculation module are critical 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 predict 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

TABLE II. NOTATION FOR THE DESCRIPTION OF THE ENERGY-SAVING CALCULATION MODULE.

Notation	Description
R_j	A set of references accessing blocks in the *j*th data disk.
$R_{k,j} \subseteq R$	A set of references accessing the *k*th block $b_{k,j}$ in the *j*th data disk
$b_{k,j}$	The *k*th block in the *j*th data disk
T_{BE}	Break-even time. Minimum idle time required to compensate the cost of entering standby
T_{ij}	Active time period serving the *i*th request issued to the *j*th data disk
t_{ij}	Time spent serving the *i*th request issued to the *j*th data disk
α_{ij}	Time spent in the idle period prior to the *i*th request accessing a block in disk *j*
I_{ij}	An idle period prior to the *i*th request accessing a block in the *j*th data disk
n_j	The total number of requests (in the look-ahead) issued to the *j*th disk
Φ_j	A set of disk access activities for references in R_j,
$time(b_{k,j})$	Active time period to serve a request accessing block $b_{k,j}$.
$block(T_{ij})$	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 standby 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

from the buffer disk. Table 2 summarizes the notation for the description of the energy-saving calculation module. To analyze circumstances in 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 *j*th data disk. Thus, R_j – a subset of the look-ahead R – can be defined as $R_j = \{r \mid r \in R \wedge disk(r) = j\text{th data disk} \wedge \text{block}(r) = b_{k,j} \wedge b_{k,j} \notin G\}$.

Given a set $R_{k,j} \subseteq R$ of references accessing the *k*th block $b_{k,j}$ in the *j*th data disk, let us derive the energy saving $E_s(b_{k,j})$ achieved by fetching $b_{k,j}$ from the data disk into the buffer disk. $R_{k,j}$ is comprised of all the requests referencing a common block $b_{k,j}$ that is not present in the buffer disk; therefore, $R_{k,j}$ can be formally expressed as

$R_{k,j} = \{r \mid r \in R \wedge block(r) = b_{k,j} \wedge b_{k,j} \notin G\}$.

Given a reference list R_j and a block $b_{k,j}$, 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 $b_{k,j}$. First, we introduce two energy saving principles utilized by PRE-BUD.

Energy Saving Principle 1: To increase the length and number of idle periods larger than disk break-even time T_{BE}, which is the minimum disk standby time required to compensate 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

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 *i*th request issued to data disk *j*; for idle period I_{ij}, α_{ij} is the time spent in the idle period prior to the *i*th request accessing a block in the *j*th data disk, and n_j is the total number of requests issued to the *j*th disk. We denote $block(T_{ij})$ as a block accessed during the active period T_{ij}.

The first three cases demonstrate scenarios that apply the *energy saving principle 1* while the fourth case uses the *energy saving principle 2* to generate longer idle periods (i.e., longer than T_{BE}) by prefetching block(T_{ij}) to combine the *i*th and (*i+1*)th idle periods. Let us pay attention to the *i*th 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 *i*th and (*i+1*)th idle periods are equal to or smaller than the break-even time T_{BE}. Thus, we have $0 < \alpha_{ij} \le T_{BE}$, $0 < \alpha_{(i+1)j} \le T_{BE}$, and $\alpha_{ij} + t_{ij} + \alpha_{(i+1)j} > T_{BE}$.

Case 2: The *i*th 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 $0 < \alpha_{ij} \le T_{BE}$, $\alpha_{(i+1)j} > T_{BE}$, and $\alpha_{ij} + t_{ij} + \alpha_{(i+1)j} > T_{BE}$.

Case 3: The *i*th 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: $\alpha_{ij} > T_{BE}$, $0 < \alpha_{(i+1)j} \le T_{BE}$, and $\alpha_{ij} + t_{ij} + \alpha_{(i+1)j} > T_{BE}$.

Case 4: In this case, both α_{ij} and $\alpha_{(i+1)j}$ are larger than T_{BE}, meaning that the *j*th 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 $\alpha_{ij} + t_{ij} + \alpha_{(i+1)j} > T_{BE}$.

IV. Experimental Results

For our experimental results we implemented a parallel disk simulator in JAVA. The disk simulator implements the algorithm presented in Fig. 1 and the energy saving calculator in Fig. 2. The disk parameters used for our simulation results are given in Table 3.

The first set of experiments we conducted varied the hit rate and the data size of the requests. The hit rate in these experiments is defined as the percentage of all the requests that can be served by the buffer disk and the data size is defined as the data size of each request. We generated random disk requests and varied the inter-arrival delay of the requests. The inter-arrival rate must be fairly low to produce energy savings or disks will never be placed in the sleep state. If the inter-arrival rate is high all disks must be active to serve the requests. The results of the first set of experiments are summarized in Fig. 3.

There are two main observations we can draw from this figure, one being that as data size increases energy savings increases, and second, as the hit rate is increased energy savings increases. As the data size increases the time to serve the request increases. If multiple requests can be served from the buffer disk than the data disks have a greater opportunity to transition to the sleep state. Similarly as the hit rate increases the buffer disk serves a greater number of consecutive hits allowing data disks to sit idle for longer periods of time. The goal of our energy-efficient prefetcher is to increase the number and length of idle periods to allow a data disk to transition to the sleep state. This can be achieved by increasing the hit rate or increasing the data size of requests. This leads us to believe that many web and multimedia applications would be suitable for our energy saving techniques.

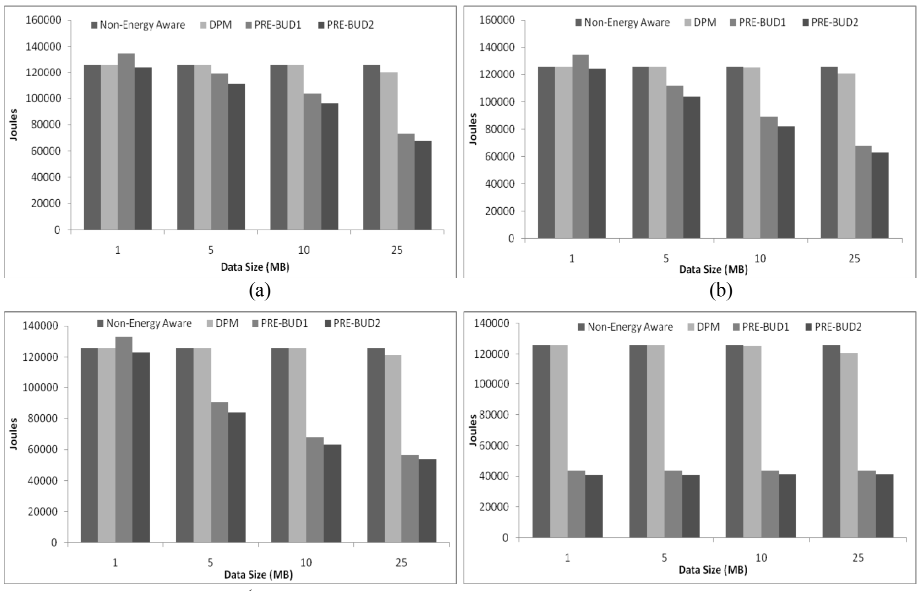
The second set of experiments conducted focuses on the impact that varying disk power parameters has on the energy savings. Fig. 4 varies the power characteristics of our simulated IBM36Z15 disk. For each figure we only vary one disk energy parameter. The number of disks was fixed at four and the data size is 25MB. From Fig. 4 we realize that

```
Input: block $b_{k,j}$, disk $j$, a set $\Phi_j$ 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 ($\alpha_{ij} + t_{ij} + \alpha_{(i+1)j} > T_{BE}$) {
4              if ($0 < \alpha_{ij} \le T_{BE}$) {
5                if ($0 < \alpha_{(i+1)j} \le T_{BE}$)
6                    $E_S(b_{ij}) = E_S(b_{ij}) + P_I \cdot (\alpha_{ij} + \alpha_{(i+1)j}) - P_S \cdot (\alpha_{ij} + t_{ij} + \alpha_{(i+1)} - T_U - T_D) - E_D - E_U$;
7                else $E_S(b_{ij}) = E_S(b_{ij}) + P_I \cdot \alpha_{ij} - P_S \cdot (\alpha_{ij} + t_{ij})$;
8          }
9          else {
10             if ($0 < \alpha_{(i+1)j} \le T_{BE}$) /* Case 3, see Eq. (4.8) */
11                 $E_S(b_{ij}) = E_S(b_{ij}) + P_I \cdot \alpha_{(i+1)j} - P_S \cdot (\alpha_{(i+1)j} + T_{ij})$;
12             else $E_S(b_{ij}) = E_S(b_{ij}) + E_D + E_U - P_S \cdot (T_D + T_U + t_{ij})$.
13         }
14       } /* end Cases 1-4 */
15      else $E_S(b_{ij}) = E_S(b_{ij}) - P_I \cdot t_{ij}$; /* Negative energy saving. */
16   } /* end for */
17   return $E_S(b_{ij}) - P_A \cdot \text{time}(b_{k,j})$
```

Figure 2. The energy-saving calculation module.

TABLE III. DISK PARAMETERS USED FOR SIMULATIONS

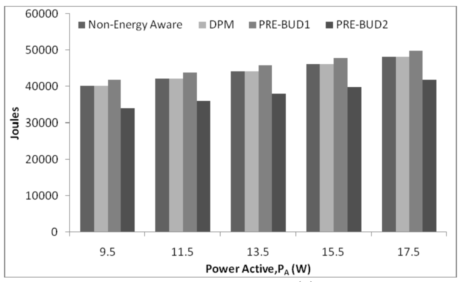
Transfer Rate	***P_A***	***P_I***	***P_S***	***T_D***	***T_U***	***E_D***	***E_U***
Simulation Parameters							
55 MB/S	13.5 W	10.2 W	2.5 W	1.5 S	10.9 S	13.0 J	135 J



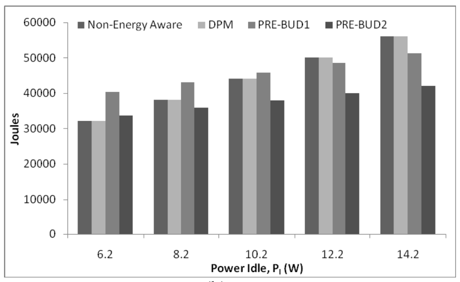
(c)

(d)

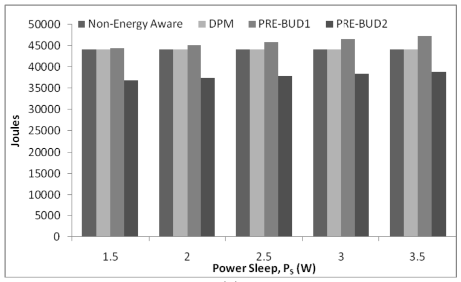
Figure 3. Total Energy Consumption of Disk System while Data Size is varied for four different values of hit rate: (a) 85 %, (b) 90 %, (c) 95 %, and (d) 100 % hit rate.



(a)



(b)



(c)

Figure 4. Total Energy consumption for various values of the following disk parameters: (a) power active, (b) power idle, and (c) power standby

lowering the Power Active, which is the energy consumed while the disk is in the active state, will decrease the energy consumption for all the strategies we compared. Lowering Power Active also impacts the relative energy savings that the PRE-BUD strategies are able to produce. If Power Active is 9.5W PRE-BUD2 saves 15.1 % energy over DPM. If it is increased to 17.5W PRE-BUD2 only saves 13% energy over DPM. Fig. 4 (a) is similar to Fig. 4 (b) but now we see that Power Idle has a greater impact on energy savings as compared to Power Active. If Power Idle, the energy consumed while the disk is idle, is very low PRE-BUD 2 has a negative impact, but if it's increased to 14.2 W PRE-BUD 2 now saves 25 % of energy as compared to DPM. The last set of experiments varied the Power Sleep parameter, which represents the energy consumed while the disk is in the sleep state, also has significant impact on PRE-BUD strategies. The percentage change in energy savings starts at 16.3% and drops to 11.7% with increasing Power Sleep.

The results illustrated in Fig. 4 indicate that parallel disks with low active power, high idle power, and low standby power can produce the best energy-savings benefit. This is because PRE-BUD allows disks to be spun down to the standby state during times they would be idle using DPM. The greater the discrepancy between idle and standby power, the more beneficial PRE-BUD becomes.

V. Conclusions and Future Work

The use of large-scale parallel disk systems continues to rise as the demand for information systems with large capacities grows. Parallel disk 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 disk system using a buffer disk to which frequently accessed data are prefetched.

We proposed two different buffer disk configurations. The first configuration added an extra disk to the parallel disk system, whereas the second one used an existing disk as the buffer disk. Placing popular data blocks in the buffer disk provides ample opportunities to increase idle periods in data disks, thereby facilitating long sleep times of disks. As sleep times are increased the disk is able to save energy over being in the idle state. Although the first prefetching strategy may consume more energy due to the energy overhead introduced by an extra disk, it does not compromise the capacity of the disk system. We implemented a simulator and compared our approaches against three existing approaches.

For the future research work we will investigate the scalability of the energy-efficient prefetching algorithm by adding more than one buffer disk to the disk system. The number of buffer disks will have to be increased as the scale of the disk system is increased. This will add the extra requirements parallel applications demand and our strategies will have to be modified to reflect these changes. There is also a need to support write operations in the BUD architecture. Furthermore, we plan to use traces from real-world applications to evaluate the performance of PRE-BUD. Last but not least, we will study the reliability impacts of buffer disks on parallel disk systems.

Acknowledgement

This work was made possible partially thanks to NSF awards CCF-0845257 (CAREER), CNS-0757778 (CSR), CCF-0742187 (CPA), CNS-0831502 (CyberTrust), OCI-0753305 (CI-TEAM), DUE-0837341 (CCLI), and DUE-0830831 (SFS), and an Intel gift number 2005-04-070 as well as an Auburn University startup grant.

References

[1] L. Benini, A. Bogliolo, and G. Michelini. "A Survey of Design Techniques for System Level Dynamic Power Management," *IEEE Trans. On Very Large Scale Integration (VLSI) Systems*, Vol. 8, No. 3, June 2000.

[2] E. Carrera, E. Pinheiro, and R. Bianchini. "Conserving Disk Energy in Network Servers," *Proc. Int'l Conf. Supercomp.*, pp.86-97, 2003.

[3] J. Chase and Ron Doyle. "Energy Management for Server Clusters," *Proc. the 8th Workshop Hot Topics Operating Sys.,* pp. 165, May 2001.

[4] F. Chen, S. Jiang, and W. Yu, "FlexFetch: A History-Aware Scheme for I/O Energy Saving in Mobile Computing," *Int'l. Conf. on Parallel Processing*, Sept. 2007.

[5] F. Chen, S. Jiang, and X. Zhang, "SmartSaver: Turning Flash Drive Into a Disk Energy Saver for Mobile Computers," *Int'l. Symp. on Low Power Electronics and Design*, Oct. 2006.

[6] D. Colarelli and D. Grunwald. Massive Arrays of Idle Disks for Storage Archives. *In Proceedings of Supercomputing*, November 2002.

[7] F. Douglis, P. Krishnan, and B. Marsh, "Thwarting the Power-Hunger Disk," *Proc. Winter USENIX Conf.*, pp.292-306, 1994.

[8] H. Eom and J.K. Hollingsworth. "Speed vs. accuracy in simulation for I/O-intensive applications," *Proc. Int'l Symp. Parallel and Distri. Processing Symp.*, pp. 315–322, May 2005.

[9] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Fanke, "DRPM: Dynamic Speed Control for Power Management in Server Class Disks," *Proc. Int'l Symp. Computer Architecture*, pp. 169-179, June 2003.

[10] J. Hawkins and M. Boden, M., "The applicability of recurrent neural networks for biological sequence analysis," *IEEE/ACM Trans. Comp. Biology and Bioinfo.*, vol. 2, no. 3, pp. 243 – 253, July-Sept. 2005..

[11] S. Huaping, M. Kuman, S. Das, Z. Wang. "Energy-Efficient Caching and Prefetching with Data Consistency in Mobile Distributed Systems," *Proc. Int'l Parallel and Distri. Proc. Symp.*, 2007.

[12] M. Kallahalla and P. Varman, "PC-OPT: Optimal Offline Prefetching and Caching for Parallel I/O Systems," *IEEE Trans. Computers*, vol. 51, no. 11, pp. 1333-1344, Nov. 2002.

[13] Y-J. Kim, K-T Kwon, and J. Kim, "Energy-efficient disk replacement and file placement techniques for mobile systems with hard disks," *ACM Special Interest Group on Applied Computing*, 2007.

[14] T.T. Kwan, R.E. McGrath, and D.A Reed, "NCSA's World Wide Web Server: Design and Performance," *Computer*, vol. 28, no. 11, pp. 68 – 74, Nov. 1995.

[15] A. E. Papathanasiou and M.L. Scott, "Energy Efficient Prefetching and Caching," *USENIX* 2004.

[16] E. Pinheiro and R. Bianchini, "Energy Conservation Techniques for Disk Array-Based Servers," *Proc. Int'l Conf. Supercomputing*, pp. 68-78, June 2004.

[17] E. Pinheiro, R. Bianchini, C. Dubnicki, "Exploiting Redundancy to Conserve Energy in Storage Systems," *Proc. Sigmetrics and Performance*, Saint Malo, France, June 2006.

[18] S.W. Son, M. Kandemir, "Energy Aware Pre-Fetching for Multi Speed Disks," *Proc. of the 3rd Conf. Comp. Frontiers*, pp. 105-114, May 2006.

[19] S.W. Son, M. Kandemir, and A. Choudhary, "Software-Directed Disk Power Management for Scientific Applications," *Proc. Int'l Symp. Parallel and Distr. Processing*, April, 2005.

[20] D.B. Trizna, "Microwave and HF Multi-Frequency Radars for Dual-Use Coastal Remote Sensing Applications," *Proc. MTS/IEEE OCEANS*, pp. 532 - 537, Sept. 2005.

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

[22] Q. Zhu, Z. Chen, L. Tan, Y. Zhor, K. Keeton, J. Wikes. "Hibernator Helping Disk Arrays Sleep Through The Winter," *Proc. ACM Symp. Operating Sys. Principles*, October. 2005.

[23] Q. Zhu, F. M. David, C. F. Devaaraj, Z. Li, Y. Zhou, and P. Cao, "Reducing Energy Consumption of Disk Storage Using Power-Aware Cache Management," *Proc. High-Performance Computer Arch.*, 2004.

[24] Q. Zhu, A. Shankar and Y. Zhou, "PB-LRU: A Self-Tuning Power Aware Storage Cache Replacement Algorithm for Conserving Disk Energy," *Int'l Conf. Supercomputing*, 2005.

[25] X. Zhuang and S. Pande. "Power-Efficient Prefetching via Bit-Differential Offset Assignment on Embedded Processors," *Proc. ACM SIGPLAN/SIGBED Conf. Languages, Compilers, and Tools for Embedded Sys.,* 2004.