Energy Efficient Storage Management Cooperated with Large Data Intensive Applications

Norifumi Nishikawa [#1], Miyuki Nakano [#2], Masaru Kitsuregawa [#3]

[#]*Institute of Industrial Science, The University of Tokyo*
4-6-1 Komaba, Meguro-ku, Tokyo, Japan
[1]norifumi@tkl.iis.u-tokyo.ac.jp
[2]miyuki@tkl.iis.u-tokyo.ac.jp
[3]kitsure@tkl.iis.u-tokyo.ac.jp

***Abstract*—Power, especially that consumed for storing data, and cooling costs for datacenters have increased rapidly. The main applications running at datacenters are data intensive applications such as large file servers or database systems. Recently, power management of the data intensive applications has been emphasized in the literature. Such reports discuss the importance of power savings. However, these reports lack research on power management models for the efficient use of data intensive applications' I/O behaviors. This paper proposes a novel energy efficient storage management system that monitors both application- and device-level I/O patterns at run time, and uses not only the device-level I/O pattern but also application-level patterns. First, the design of the proposed model combined with such large data intensive applications will be shown. The key features of the model are i) classifying application-level I/O into four patterns using run-time access behaviors such as the length of idle time and read/write frequency, and ii) adopting an appropriate power-saving method-based on these application level I/O patterns. Next, the proposed method is quantitatively evaluated with typical data intensive applications such as file servers, OLTP, and DSS. It is shown that energy efficient storage management is effective in achieving large power savings compared with traditional approaches while an application is running.**

I. Introduction

The amount of digital data produced by humans is increasing every day. Based on the IDC report [1], the amount of information and content created and stored digitally will grow to over 7 zettabytes by 2015. This explosion of digital data must be managed and used with data intensive applications such as sensor data archives, search engines, and customer management systems (OLTP). As well, the increased data must be stored somewhere, which implies that storage capacity must also increase. The report [2] says that the storage capacity shipment in 2014 will be 7 times as large as that in 2009.

Today, the energy consumption rate of IT equipment in data centers has increased greatly[3]. A report [4] notes that the storage energy consumption rate for all IT equipment will increase more and more in the short term because the amount of digital data is increasing quickly. For example, the paper [5] reports that the storage power consumed for large online transaction processing systems (OLTP, data intensive applications) consumes more than 70% of the total power of all IT equipment. Thus, power reduction for storage is strongly required in large datacenters.

Many storage power saving methods have been proposed. In order to use a disk's power control functions, some typical methods extend the duration of the idle period by controlling the I/O interval in storage devices [6], [7], [8], [9], and some extend the idle period by replacing data between disks [10], [11], [12], [13], [14], [15], [16].

Many applications run at datacenters today. I/O behaviors of applications are quite different in different applications. For example, TPC-C issues random I/O to master data tables, and TPC-H issues sequential read commands mainly to large transaction tables. These I/O behaviors of data intensive applications will become good hints for power saving of storages.

However, previous storage power saving methods use only storage level I/O behaviors and do not utilize application level I/O behaviors effectively. Thus these previous methods based on storage level I/O behaviors may improperly apply storage power saving function at run time even though a length of I/O intervals is not enough to obtain the power saving effect. Therefore, previous power saving methods may cause run time performance degradation of applications. By considering the application I/O behaviors, we may avoid applying the power saving function at improper timing and it is likely that we achieve high power saving effects without high performance degradation of applications compared with previous works.

In this paper, we propose a novel power saving method that utilizes application level I/O behaviors to storage power saving. The goal is to construct an energy efficient storage management system combined with data intensive applications. The proposed storage management chooses an appropriate power-saving method-based on the characteristics of the application's I/O behavior. First, the design of the energy efficient storage management model is proposed. As well, the method for combining storage power management with I/O behaviors of large data intensive applications is explained. Features of the proposed model are i) saving storage power during running applications, ii) classifying application-level logical I/Os into four patterns by monitoring the run time access behaviors of applications, such as the length of idle time and read/write frequency, and iii) adopting an appropriate power-saving method-based on the logical I/O patterns. Next, the proposed method is quantitatively evaluated with typical data intensive applications such as file server, OLTP, and DSS.

The I/O trace for multiple production enterprise workloads from Microsoft Research [13] (MSR) is called the File Server workload, I/O trace generated by TPC-C benchmark [17] is the OLTP workload, and I/O trace generated by TPC-H benchmark [18] is the DSS workload. These I/O traces are replayed using a trace replay tool [19] and the power consumption for enterprise storage is actually measured. The proposed method is compared with two previous methods, popular data concentration [11] and dynamic data reorganization [15]. The evaluation results confirm the potential of the proposed method for actual applications such as File servers, OLTP, and DSS. Further, the advantages of the proposed method are discussed with statistics of I/O trace data, and the advantage of the application I/O behavior usage for runtime power saving of storage systems is considered.

Section II presents the design of the energy efficient storage management system combined with large data intensive applications. Section III shows the monitoring system. Section IV shows the power management functions. Section V describes the runtime power saving methods. Section VI describes the I/O characteristics of File Servers, OLTP and DSS. Section VII examines the experimental results. Finally, Section VIII presents related work, and Section IX concludes the paper.

II. Design of an Energy Efficient Storage Management System Combined with Large Data Intensive Applications

Today, much IT equipment provides energy saving functions such as a processor's Halt or a DVFS function. As well as processors, enterprise storage has some power saving functions, such as power on and off of disk enclosures, in addition to simple power management of HDDs. However, the power saving function for the storage units cannot always effectively achieve high energy saving. For example, if there is a gain in power saving by using the power on and off function, at least one I/O interval must be sufficiently longer than the period for turning the storage off and on.

Data intensive applications running at large datacenters have their own I/O behaviors as described at section I. If these application-specific I/O behaviors are used to save storage energy consumption, it may be possible to reduce runtime storage energy consumption more efficiently compared with previous work.

In order to use the applications' I/O behaviors to obtain storage energy savings, the concept of *logical I/O patterns* is introduced. The logical I/O pattern is a classified and patterned applications' I/O behaviors and is information suitable for choosing a storage energy saving function. The logical I/O pattern can be classified into four patterns. The first logical I/O pattern is a pattern that has no I/O issued from the applications during the monitoring period, which means that a power saving function can easily be adapted. The second pattern enlarges the read I/O intervals using a storage cache. The third logical I/O patterns enlarges write I/O intervals by delaying the write I/O of data within the storage cache. The last I/O pattern is a pattern to distinguish I/O behaviors to which one cannot apply the storage energy saving functions. By classifying the application's I/O behaviors based on the logical I/O patterns, it is possible to extract energy saving-enabled I/O behaviors for running applications and easily achieve high energy saving for runtime energy consumption of storage units.

A. Storage Model

The storage power management targets so-called enterprise storage units, which are large storage systems mainly used at data centers. These enterprise storage units consist of various components such as RAID controllers with a large battery back-up cache and some disk enclosures that contain multiple HDDs. Components, such as the whole storage system, disk enclosures, and HDDs are considered the power saving units. This paper focuses on disk enclosures as the power saving unit. If the whole storage unit is powered off, it will need to stop all application-related calls to the storage. Furthermore, it takes a long time to turn the whole large storage back on again. Thus, the power control of the whole storage is very inefficient for run-time power saving. Next, we consider a HDD as a power saving unit. Almost all enterprise storages configure RAID on multiple HDDs in a disk enclosure. I/Os issued to a disk enclosure are divided into I/Os to HDDs, which implies that HDD-level I/O behaviors are similar to those of the disk enclosure level. The timing of turning on and off HDDs in a disk enclosure will be the same timing as turning on and off the disk enclosure. Therefore, it is proposed that the efficiency of power saving at the HDD level is equivalent to that at the disk enclosure level.

Enterprise storages also have a large battery back-up cache. Energy efficient storage management uses the storage cache in order to enlarge I/O intervals of the disk enclosures. There are two approaches for enlarging I/O intervals using a storage cache: enlarging read I/O intervals by preloading data into the cache before they are read from applications, and enlarging write I/O intervals by keeping write I/Os in the cache and delaying write I/Os to disk enclosures (write delay). The proposed method uses both approaches *preload* and *write delay* for energy saving in storage units.

B. Power Management Mode and Break-Even Time

1) Power Management Mode: A disk enclosure can have three power modes: Active, Idle, and Power off.

- Active: A disk enclosure is powered-on and I/Os are executed. The power consumption of this mode is the highest of the three modes.
- Idle: A disk enclosure is powered on but no I/O is executed. The power consumption of this mode is lower than that of the active mode.
- Power off: A disk enclosure is turned off. If I/Os are issued to a powered off disk enclosure, it takes a fixed time to turn on the disk enclosure. It also requires additional energy to turn a disk enclosure on.

2) Break-Even Time: The break-even time is an indicator of storage power management. Once a disk enclosure is turned off, it requires some energy to be turned on again. On the

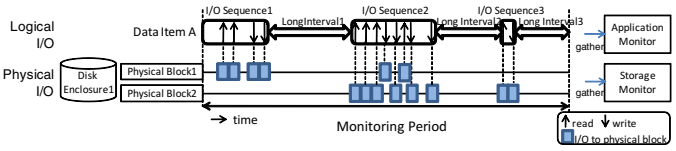

Fig. 1. Long Intervals and I/O Sequences

other hand, it consumes some energy to remain in idle mode waiting for an I/O request. There exists an energy trade-off between power off and idle modes. If there is enough of a time interval before the next I/O request, power off mode may be energy efficient compared with idle mode. However, if there is not enough of a time interval, the power off mode consumes more energy than that of remaining in idle mode, because the energy of making a disk enclosure power on is much larger than that consumed by idle mode. In order to reduce power consumption of disk enclosures in power off mode, the I/O intervals of a disk enclosure must be longer than the break-even time.

C. Data Item and Logical I/O Pattern

As described above, applications have their own I/O behaviors. In order to achieve high energy reduction, a power saving method should be carefully selected and applied to the storage units by considering the differences in the application's I/O behaviors. In order to identify I/O behaviors of applications, the idea of *data item* and *logical I/O pattern* are introduced.

1) Data Item: A data item is a fragment of an application's *data* on one disk enclosure. A unit of data differs depending on the type of application. For DBMS applications such as OLTP or DSS, a unit of data may be a table or a database index. As for applications that run on file servers, a unit of data may be a file. If some parts of the data lie on multiple disk enclosures, we treat these partitions as different data items. By splitting data into data items, we can identify I/O behavior for the applications' data on the disk enclosures.

2) Logical I/O Pattern: A logical I/O pattern means a patterned I/O behavior of applications, and is used for choosing a power saving function. In order to identify a logical I/O pattern, we introduce the concepts of *Long Interval* and *I/O Sequence*. A Long Interval is an I/O interval that is longer than the break-even time. An I/O Sequence consists of a sequence of some read/write I/Os to a data item and an I/O interval(s) shorter than the break-even time. Fig. 1 shows an example of Long Intervals and I/O Sequences for data item A and its physical I/O. Data item A has three Long Intervals and three I/O Sequences. Long Interval #3 ends at the end of a monitoring period. I/O Sequence #1 starts at the beginning of the monitoring period. As shown in Fig. 1, we can find that the Data Item A is on a Disk Enclosure #1 by combining its logical and physical I/O.

It is inefficient to tailor an appropriate power management based on each application's I/O characteristics even though we can observe the application-level I/O behaviors. Thus, four logical I/O patterns are introduced to identify data items so that it is possible to easily execute better power management for each data item regardless of application type.

Definitions of logical I/O patterns are as follows.

- **I/O Pattern P0**: No I/Os are issued to the data item during a monitoring period. This pattern has only a single Long Interval and no I/O Sequence.
- **I/O Pattern P1**: The I/O behavior of the data item has at least one Long Interval and at least one I/O Sequence, and the total number of reads in I/O Sequences is larger than 50% of the total number of I/Os. P1 data items are read frequently, thus they are candidates for preloading into a storage cache.
- **I/O Pattern P2**: The I/O behavior of the data item has at least one Long Interval and at least one I/O Sequence, and the total number of reads in the I/O Sequences is less than 50% of the total number of I/Os in the I/O Sequences. P2 data items are candidates for enlarging write I/O intervals by delaying write I/Os to disk enclosures.
- **I/O Pattern P3**: The I/O behavior of the data item has only one I/O Sequence, that is, all I/O intervals of the data item are shorter than the break-even time. P3 data items are candidates that are not applied to a turn-off function because the P3 data items have no long intervals.

The logical I/O patterns are classified into four patterns. The power saving methods of enterprise storage are i) turning off disk enclosures, ii) preloading data items, and iii) delaying write I/O of data items. This means that it is sufficient to consider how these three power-saving methods are applied to each disk enclosure. Therefore, introduce three patterns, P0, P1, and P2. The last I/O pattern P3 is considered as a mark of data items that are not suitable for energy savings. By considering only four logical I/O patterns, we can cover all of data items in order to choose an appropriate power saving methods for each data item. Thus, it is proposed that four Logical I/O patterns are enough to combine the application's I/O behaviors with storage-level I/O behavior and utilize the application's I/O behaviors to power saving of storages.

D. Energy Efficient Storage Power Management Model

Fig. 2 shows the proposed energy efficient storage power management model. A block diagram on the left of Fig. 2 shows the system components and the storage management modules. The application monitor component watches application I/O behaviors and stores them as I/O trace records in its repository. The storage monitor component traces the I/O behavior of disk enclosures, and stores them as I/O trace records in its repository. The power management component gathers these stored I/O traces. The power management component also associates I/O trace records with each data item and determines an I/O pattern for each data item. After determining the I/O pattern, the power management decides the data items' placement, and adopts an appropriate power saving method for the disk enclosures. It notifies the power saving method to the run-time power-saving component. The run-time power-saving component turns the disk enclosures on and off. It also executes the power saving method using a storage cache.

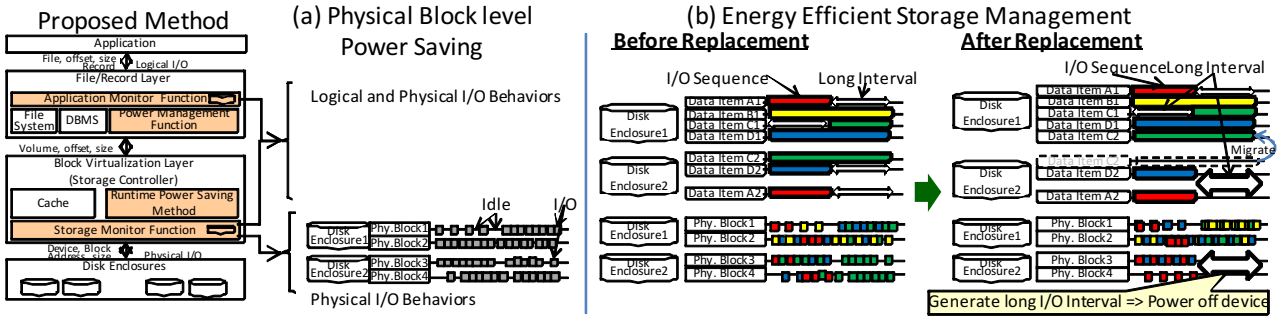

Fig. 2. Application-Collaborative Storage Power Management Model

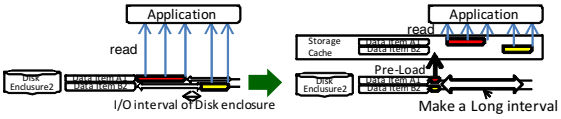

Fig. 3. Effectiveness of Preload

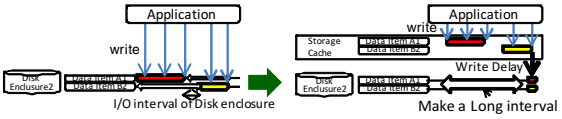

Fig. 4. Effectiveness of Write Delay

Fig. 2(a) shows the traditional physical block-based power saving. The application-level I/O behavior and physical-level ones are not combined in the conventional method. Thus, it is impossible to distinguish I/Os to data items on the same physical block by analyzing only the physical-level I/O behaviors, which means that it is difficult to choose an appropriate power-saving method for each application. The energy efficient storage management (Fig. 2(b)) combines both logical I/O behaviors and physical I/O behaviors. The left-hand side of Fig. 2(b) shows the relationship between I/O patterns of the data items and the I/Os to the physical block. The right-hand side of Fig. 2(b) shows the power-saving potential of the proposed storage power management system.

As shown in the left-hand side of Fig.2(b), it is possible to find the data items A1, C1, D2, and A2 that contain long intervals. Furthermore, it is possible to separate disk enclosures for each data item using the mapping information between data items and their physical blocks. Therefore, the data items can be replaced by the disk enclosures in order to make I/O intervals longer than the break-even time. As shown in the right-hand side of Fig. 2(b), the power-saving potential of disk enclosure #2 can be increased by migrating the C2 data item onto disk enclosure #1. If we do not use logical I/O patterns, we cannot find long intervals in physical blocks #1 to #4, so that we may not turn off the disk enclosures.

E. Power Saving Method

As shown in Fig. 2(b), the model analyses the logical I/O patterns and adopts an appropriate power saving method to disk enclosures based on the statistics of logical I/O patterns. The proposed system provides two types of power saving functions: data placement control and I/O interval control using the storage cache.

1) Data Placement Control: In order to apply power off to a disk enclosure, I/O intervals to some disk enclosures must be longer than the break-even time. To achieve this, the proposed data placement method puts P3 data items into the same disk enclosures, and makes I/O intervals of the *other* disk enclosures longer than the break-even time.

2) I/O Interval Control using Storage Cache: It is desired to turn off disk enclosures that store only P0, P1, and P2 data items for a long time. Since I/O timing of these data items is different, the I/O intervals of the disk enclosure becomes shorter than the I/O intervals of each data item. Enterprise storages have large battery back-up caches in a RAID controller [20]. Usage of the large cache is considered in order to enlarge I/O intervals.

First, consider *preload*. The preload function reads P1 data items into a storage cache before the data items are read from the applications and keeps them in the storage cache. After these data items are stored in the storage cache, the applications read the data items from the storage cache, and no read I/Os are issued to disk enclosures. The latency for reading P1 data items into the storage cache is usually much shorter than the execution time of the applications. Thus, this creates Long Intervals in the disk enclosures. As shown in Fig. 3, the read I/O intervals of the disk enclosure can be increased by preloading data items A1 and B2 into the storage cache.

Next, consider *write delay*. The write delay function keeps the updated parts of P2 data items in the storage cache, and writes them to disk enclosures in one go. Usually, the latency for writing the data from the storage cache onto the disk enclosures is shorter than the time that the applications are running. Therefore, a Long Interval is created in the disk enclosures. As shown in Fig. 4, the write I/O intervals of disk enclosures can be increased by delaying write I/O of data items A1 and A2. The storage cache is backed-up with a battery, thus the ACID feature of DBMS is guaranteed.

III. Monitoring System

This section describes the monitoring functions of the proposed energy efficient storage management system. As

shown in Fig. 2, the proposed management system has two monitoring functions (Application Monitor and Storage Monitor) in order to use both application and storage-level I/O behaviors for energy saving. The Application Monitor lies on the File/Record Layer. The Storage Monitor lies on the Block Virtualization Layer in Fig2.

A. Application Monitor

An Application Monitor collects two types of information: logical mapping information and logical I/O trace.

- **Logical Mapping Information**: Logical mapping information contains a relationship between data and volumes. The volumes are provided to the file/record layer by block virtualization.
- **Logical I/O Trace**: The logical I/O trace contains a timestamp of when the I/O is issued, a data identifier, I/O address (offset) of the data, I/O size, and I/O type (read or write).

Logical mapping information is created, updated, or deleted when data are created, expanded, shrunk, or deleted. The mapping information is stored in the repository in the application monitor. A logical I/O trace is captured when I/O is issued from the application and stored into memory in the application monitor. If the memory becomes full, the I/O trace is stored in the repository of the monitor. The logical I/O trace information is equivalent to that which the DBMS collects. Therefore, the overhead of collecting logical I/O trace is small.

B. Storage Monitor

A Storage Monitor collects physical mapping information, physical I/O trace, power status of the disk enclosures, and power consumption of the disk enclosures.

- **Physical Mapping Information**: Physical mapping information contains a relationship between an offset of a volume and block address of a disk enclosure.
- **Physical I/O Trace**: The monitor collects I/O traces that the block virtualization layer issues to the disk enclosures. The physical I/O trace contains a timestamp, a name of a disk enclosure, a block address, and I/O type (read or write).
- **Power Status of the Storage Device**: Power status of a disk enclosure contains the name of a disk enclosure, a timestamp of when the power of the disk enclosure is on and off, and a power status (on and off) of the disk enclosure.
- **Power Consumption of the Storage Device**: The disk enclosure's power consumption information contains the name of the disk enclosure, a timestamp of when power consumption of the disk enclosure is collected, and power consumption of the disk enclosure.

IV. Power Management Function

A. Overview

The energy efficient storage management system monitors application and storage I/O behaviors for a fixed period. At the end of the monitoring period, the system invokes the power management function.

The power management function determines Logical I/O patterns for the data items. Then, the function separates the disk enclosures into hot ones and cold ones using the Logical I/O patterns of the data items. Hot disk enclosures mainly store P3 data items. However, the method intends to keep the initial data placement in order to avoid data migration overhead. Thus, a hot disk enclosure may store all I/O patterns of data items. Cold disk enclosures store data items of type P0, P1, and P2. Based on Logical I/O patterns stored in cold disk enclosures, an appropriate power-saving strategy can be selected for each disk enclosure. Next, the function determines the placements of data items based on their Logical I/O patterns. Using Logical I/O patterns, the function may have a chance to create cold disk enclosures even though all of the disk enclosures store P3 data items as described below.

The function tries to enlarge the I/O intervals of cold disk enclosures by applying write delay and preload functions to data items in the cold disk enclosures. Cold disk enclosures are accessed as little as possible. Thus, high energy savings are expected by increasing the I/O intervals for cold disk enclosures compared with trying to increase the I/O intervals of hot disk enclosures. The function applies a write delay function first and then applies a preload function. This is because the efficiency of the write delay function may be higher than that of the preload function. A storage cache is non-volatile in enterprise storage. Thus the enterprise storage can decide a timing of write I/O to disk enclosures independent from a timing of application's write I/O. On the other hand, the prediction of read timing is difficult because it depends on the run-time status of the applications.

Finally, the function sets the storage units to apply their power-off function to only the cold disk enclosures. The function also calculates the length of the monitoring period based on the distribution of the lengths of the I/O intervals of the latest monitoring period in order to choose an appropriate period for power-saving.

Algorithm 1 shows an outline of the function.

Algorithm 1 Power Management Function

```
Start application monitor and storage monitor;
while applications are running do
    Wait until monitoring is finished;
    Determine Logical I/O pattern of data items;
    Determine Hot and Cold Disk Enclosures;
    Determine Data Placement;
    Determine Write Delay for Applicable Data Item;
    Determine Preload for Applicable Data Item;
    Determine the Power Control Method for the Disk Enclosures;
    Determine the length of the next monitoring period;
end while
```

B. Determine Logical I/O Pattern

The I/O pattern determination function extracts Logical I/O patterns of data items as follows:

- **Step1. Find Long Interval**: First, this function splits a logical I/O trace into a unit of data item. Then checks if the logical I/O trace contains I/O intervals longer than the break-even time. If so, this step marks these I/O intervals as Long Intervals. If no I/Os are issued to the data item, then this step also marks the I/O interval (its length is equal to the monitoring period) as a Long Interval.
- **Step2. Find I/O Sequence**: Next, this function extracts I/O sequences from rest parts of the I/O trace.
- **Step3. Determine the I/O Pattern of the Data Item**: If no I/Os are issued to the data item, the I/O pattern of the data item is $P0$. If the data item has no Long Intervals, then the I/O pattern of the data item is $P3$. For the remaining data items, the I/O pattern determination function counts the numbers of read and write I/Os of these data items. If more than half of the I/Os are read I/Os, then this function determines a Logical I/O pattern of the data items as $P1$; otherwise it is $P2$.

C. Determining Hot and Cold Disk Enclosures

Next, the power management function determines hot and cold disk enclosures. The function chooses hot disk enclosures based on $P3$ data items. $P3$ data items are accessed frequently, so a degradation of I/O performance for $P3$ data items causes performance degradation in an application. The function selects the disk enclosures satisfying the following conditions: i) The total served IOPS of the selected disk enclosures is larger than the sum of IOPS of $P3$ data items, and ii) The size of the selected hot disk enclosures is larger than the sum of $P3$ data items' size. The cost of data item migration among disk enclosures is high. Therefore, the function selects the hot disk enclosures from the largest order of size of $P3$ data items in the disk enclosure in order to reduce the cost of data item migration. The system determines hot and cold disk enclosures as follows:

- **Step1. Calculate Maximum IOPS of P3 data items**: The maximum IOPS I_{max} is calculated as follows: $I_{max} = max(\sum_{i=0}^{n-1} I_{it})$, where n is the number of P3 data items and I_{it} is the IOPS of P3 data item i at time t.
- **Step2. Calculate the Number of Hot Disk Enclosures**: The number of hot disk enclosures N_{hot} is calculated as follows: $N_{hot} = max(\lceil I_{max}/O \rceil, \lceil (\sum s_i)/S \rceil)$, where O is the maximum IOPS that a disk enclosure can serve, S is the size of a disk enclosure, and s_i is the size of the data item of type $P3$. N_{hot} is selected so that the hot disk enclosures can serve the required IOPS for all P3 data items and can store all P3 data items.
- **Step3. Choose Hot and Cold Disk Enclosures**: First, Step 3 sorts the disk enclosures in descending order by the total size of P3 data items in each disk enclosure. Then, this step selects the top N_{hot} disk enclosures as hot disk enclosures. The other disk enclosures are cold disk enclosures. If N_{hot} is larger than the number of disk enclosures, all of the disk enclosures are selected as hot disk enclosures. By selecting the top N_{hot} disk enclosures as hot disk enclosures, we can reduce the size of P3 data items to be moved from cold disk enclosures to hot disk enclosures.

D. Determining Data Placement

For the determination of the placement of data items, two data placement algorithms are provided for P3 and for P0, P1, and P2, respectively.

Algorithm 2 shows a data placement algorithm for P3 data items. In the algorithm 2, first consider the placement of P3 data items stored on cold disk enclosures. The P3 data items in the cold disk enclosures are migrated to a hot disk enclosure that satisfies the following conditions in order to balance a load among hot disk enclosures: i) the capacity of the served IOPS of the hot disk enclosure is larger than the maximum IOPS of the migrated P3 data item, ii) the capacity of the served IOPS of the hot disk enclosure is the largest, iii) the free space of the hot disk enclosure is larger than the size of the migrated P3 data item. The algorithm does not move P3 data items in hot disk enclosures. If all of the hot disk enclosures satisfy only condition i) and ii), then the algorithm tries to migrate P0, P1, and P2 data items in the hot disk enclosure to cold disk enclosures using algorithm 3 in order to make free space on the hot disk enclosures. If the algorithm cannot find any hot disk enclosures, then the algorithm increase N_{hot} and retries steps from a selection of hot disk enclosres.

Algorithm 2 Calculate Data Placement for P3 Data Items

```
M ←P3 data items in cold disk enclosures;
Sort elements of M by IOPS/size in descending order;
i = 0;
for number of elements in M do
    d ← M[i];
    s ← hot disk enclosure which average IOPS is minimum;
    if d.averageIOPS+s.averageIOPS< O and d.size+s.usedSize< S then
        Set s as a target disk enclosure of d;
    else if d.averageIOPS+s.averageIOPS >= O then
        Increase N_hot and retry this algorithm;
    else if d.size+s.usedSize >= S then
        s ← hot disk enclosure which average IOPS is next minimum and
        retly;
    end if
    increment i;
end for
```

Algorithm 3 shows a data placement algorithm for P0, P1, and P2 data items. The P0, P1, or P2 data item on a hot disk enclosure is migrated to a cold disk enclosure when the hot disk enclosure does not have enough space to store P3 data items. The target cold disk enclosure is selected so that it satisfies the following conditions: i) the IOPS is the largest of all cold disk enclosures, ii) the capacity of served IOPS is larger than the IOPS of the migrated P0, P1, and P2 data items, and iii) the available free space is larger than the migrated P0, P1, and P2 data items.

E. Determining Which Data Item Should be Write Delayed

Enterprise storage units decide a write I/O timing to their disk enclosures. This write behavior usually delays write I/Os

Algorithm 3 Calculate Data Placement for P0, P1, P2 Data Item

```
M ←P1 and P2 data items in the hot disk enclosures;
i = 0;
for length of M do
    d ← M[i];
    s ← a cold disk enclosure that its l_max is maximum;
    if d.size < S - s.usedSize and l_max+d.iops < O then
        Set s as a target disk enclosure of d;
    else
        s ← a disk enclosure which l_max is next larger and retry;
    end if
    increment i;
end for
```

and may enlarge the write I/O intervals to disk enclosures. However, the storage's write function does not recognize the applications' data items and delays all updated data. This write behavior consumes cache space for write-delay, since P3 data items are updated at a high frequency, and shortens the write I/O intervals of cold disk enclosures. Therefore, we introduce a new write-delay function. Our write-delay function selects all P2 data items in the cold disk enclosures as candidates to apply the write delay function because more than 50% of the I/Os of P2 data items are write I/Os. Applying the write delay function to P2 data items may increase a length of write I/O intervals. If a storage cache has enough space after applying a write delay function to P2 data items, some of the P1 data items that have more write I/Os than others in cold disk enclosures are selected.

F. Determining Which Data Items Should be Preloaded

Enterprise storages also prefetch data that are read sequentially. However, the purpose of the prefetch function is to reduce read I/O response time and does not enlarge the read I/O intervals. To enlarge the read I/O intervals, the proposed management system selects P1 data items in the cold disk enclosures as candidates for applying a preload function. The management system sorts P1 data items based on the number of read I/O per data size in descending order, and it then selects P1 data items until the size of selected P1 data items reaches the cache space assigned for the preload function.

A storage cache may become a bottleneck because the preload function loads many P2 data items into the cache. However, the internal bandwidth of enterprise storage units exceeds 100 GB/s. Thus, the overhead of loading data is small.

G. Determining the Power Control Method for Disk Enclosures

After determining the preload applicable data items, the proposed management system determines the power control method for the disk enclosures. The management system configures the power-off function that is applied to only the cold disk enclosures.

H. Determining the Length for the Next Monitoring Period

Finally, the algorithm determines the length of the next monitoring period. The management system determines the length of the monitoring period based on the average lengths of all data items' Long Interval. The calculation is $I_{new} = average(I_{cur}) \times \alpha$, where, I_{cur} is all the measured long intervals at the current monitoring period, and I_{new} is the next monitoring period.

A parameter α, greater than 1, is introduced to increase the monitoring period when the actual I/O intervals are longer than the monitoring period. Without α, the power management function runs and consumes CPU cycles at the end of the monitoring period even though I/O intervals are larger than the monitoring period.

V. Runtime Power-Saving Method

A. Movement of Data Items

After the end of the power management function, the runtime power-saving method migrates data items between the disk enclosures based on the data placement determined in subsection IV-D. The run-time power-saving method controls data transfer I/O throughputs so as to not influence the applications' performance. The function migrates P0, P1, and P2 data items from hot disk enclosures to cold ones first. This is because the migration of these data items creates the free space required for P3 data items' migration. Data items are moved one by one, in order of M in Algorithm 2 and 3.

B. Control of Write-Delay

The run-time power-saving method instructs a storage controller to keep updated blocks of *write delay applied* data items into a cache assigned for write-delay. Then the method enlarges a *dirty block rate* from default setting in order to enlarge write I/O intervals. The dirty block rate is a storage parameter that decides a maximum rate of updated blocks in the storage cache. If the number of updated blocks in the storage cache reaches to the number of blocks calcurated from dirty block rate, then the storage flushes these updated blocks into disk enclosures at one time.

The run-time power-saving method also indicates to write updated data items onto disk enclosures when the *write delay applied* data items are changed.

C. Preload of Data Items

First, the run-time power-saving method removes the data items that are not a target of preload from the storage cache. Then, the method loads the newly selected data items as the target of the preload function into the storage cache. The method keeps data items that are already preloaded into the cache.

D. I/O Pattern Change

The run-time method executes the power management function immediately if one of the following conditions is satisfied: i) the I/O interval of the hot disk enclosures becomes longer than the break-even time, and ii) the number of powering on of the cold disk enclosure exceeds m times from the end of previous monitoring period t_e and current time t_c, where $m = 2 \times (t_c - t_e)/l_b$, l_b is the length of the break-even time. Executing the power management function immediately, the

method has an energy saving potential even if the I/O patterns change suddenly during the monitoring period.

VI. I/O Characteristics of Data Intensive Applications

For decreasing the energy consumption of the disk enclosures, the logical I/O pattern of data intensive applications is required. An investigation of three data intensive applications, File Server, OLTP, and DSS, is performed.

A. Experimental System

Fig. 5 portrays the system configuration used in the logical I/O behavior measurements. A server has 24 Intel Xeon X5670 processors and a memory size of 48 GBytes. The OS is a 64-bit version of Red Hat Enterprise Linux Server release 5.4. The storage unit is a Hitachi Adaptive Modular Storage 2500. The storage unit has 10 disk enclosures. A single disk enclosure contains 15 HDDs. The HDD of the storage is a 750-GBytes 7200 rpm SATA. A RAID 6 is configured on HDDs in a disk enclosure. The capacity of one unit is 11.25 TB. The total capacity of the disk enclosure is about 112.5 TB (both before constructing RAID). The storage cache capacity is 2.0 GB. The server and the storage is connected by one 2-Gbit fiber channel cable.

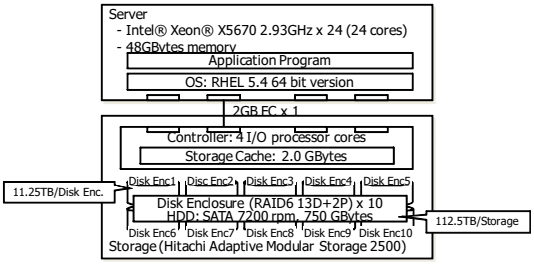

Fig. 5. Experimental System

B. Application Configuration and Load Generation

Table I shows the configuration of data intensive applications that are investigated for their logical I/O pattern.

In order to generate the load for File Server, we used the I/O trace of Microsoft Research described at [13]. On the other hand, we actually executed TPC-C and TPC-H on the experimental system (Fig. 5) as the load of OLTP and DSS respectively. The measurement period is from the start to the end of each application. These actual execution results are referred as throughput / response time without power saving in the next section.

C. I/O Data Patterns

Fig. 6 shows the measurement results of the Logical I/O patterns for data intensive applications. As shown in this figure, 89.6% of File Server data items are P1, and 9.9% of data items are P3. Almost no data items fit pattern P2. For TPC-C, 76.2% of data items are P3, and 23.3% of data items are P1. Almost no data items fit pattern P2. Unlike TPC-C, 61.5% of TPC-H data items are P1, and 38.5% of data items

TABLE I
Configuration of the Data Intensive Applications

Application	Data Size	Workload	Cache Size
File Server (MSR Trace) (6 hr)	19,800,000 records	Replay I/O trace using trace reply tool [19] Duration: 6 hr Create 36 volumes on 12 disk enclosures, and assign each volume in MSR trace to volumes in alphabetical order of the volume names.	2 GB (Storage)
OLTP (TPC-C)	500 GB	# of warehouse: 5000 # of threads: 1000 Think time: 0 Duration: 1.8 hr Put log to 1 Storage Device Put DB to 9 Storage Devices (hash distribution)	25 GB (DBMS) 2 GB (Storage)
DSS (TPC-H)	100 GB	SF=100 Run Q1 to 22 sequentially Duration: 6 hr Put log and work files to 1 Storage Device Put DB to 8 Storage Devices (hash distribution)	5 GB (DBMS) 2 GB (Storage)

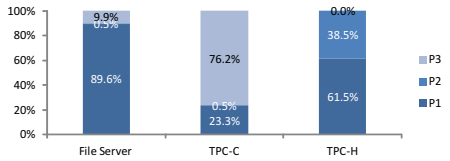

Fig. 6. Logical I/O Patterns for data intensive Applications

are P2. There are no P3 data. There are no P0 data items, since the measurement period is until the completion of the application and all data items are accessed at least once. This result shows that there is a need to change the power saving strategy for each application.

The stability of the I/O patterns was also considered. It was found that the I/O patterns of all applications are stable during the running of the application.

VII. Evaluation

This section describes the evaluation method and results for the proposed method to show its effectiveness.

A. Methodology

1) Comparison Targets: The proposed method is compared with the physical I/O behavior-based power-saving method and a logical I/O behavior-based power-saving method.

- **Dynamic Data Reorganization (DDR)**: DDR [15] is a physical I/O behavior-based data migration for saving power consumption for disk enclosures.
- **Popular Data Concentration (PDC)**: PDC [11] is a logical I/O behavior-based data migration for saving power consumption for disk enclosures. The unit of data is a file, not a data item.

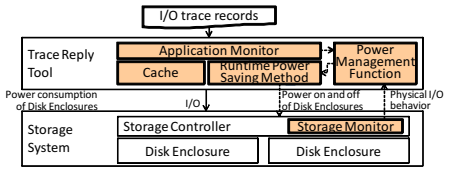

Fig. 7. Trace Replay Tool with Power Saving Method

2) Trace Replay Tool with a Power-Saving Method: The proposed power-saving method was implemented on a *blktrace* trace replay tool [19]. Fig. 7 shows a block diagram of the implementation. Our trace replay tool issues I/O for moving data items, preload data items, and flushing delayed write I/Os.

3) Test bed: The experimental system described in Fig. 5 was used as the test bed. A power meter was attached to the storage unit to measure the actual power consumption of the storage unit.

4) Measurement Items: The power consumption, I/O response time, and I/O throughput of the storage units were measured. The power consumption includes the power consumption of data item migration, preloading data items, delaying the write of data items, and powering on and off the disk enclosures. The I/O response time and I/O throughput were measured using the application monitor in the trace replay tool. The I/O response time and throughput include a delay caused by migrating data items, preloading data items, delaying write of data items, and powering on of disk enclosures.

5) Calculation of Transaction Throughput and Query Response: Our trace reply tool measures I/O response time and throughput, but the tool cannot measure an application performance. Therefore we calculate a transaction throughput of TPC-C and a query response time of TPC-H respectively. We also verify the calculated transaction throughput and query response time by comparing them with an actual transaction throughput and a query response time without power saving respectively.

The transaction throughput of TPC-C t was calculated as $t = t_{orig} \times (r/r_{orig})$, and the query response of TPC-H q was calculated as $q = q_{orig} \times (sum(r)/sum(r_{orig}))$. Here, t_{orig} is the transaction throughput of TPC-C measured as *without power saving*, q_{orig} is the query response time of TPC-H measured as *without power saving*, and r_{orig} is the average read I/O response time measured as *without power saving*.

B. Parameter Settings

Table II shows the parameter values for the proposed method. Break-even time, maximum IOPS, size of disk enclosures, and storage cache size are actual values of the storage of the test bed. The wait time for powering off a disk enclosure (Spin down time-out) is equal to the break-even time.

The storage cache size of the test bed is 2 GB. The trace replay tool recognizes that 500MB of the cache space was allocated to our preload function, and another 500MB of the cache space was allocated to our write delay function. The dirty block rate is enlarged to 50%.

TABLE II
PARAMETER VALUES FOR EVALUATION

Parameter	Value
Break-even Time	52 sec
Spin down Time-out	52 sec (Equal to Break-even Time)
Maximum IOPS of Disk Enclosure	900 (Random I/O) 2800 (Sequential I/O)
Size of Volumes on Disk Enclosure	1.7 TB
Storage Cache Size	2 GB
Cache Size for Write Delay	500 MB
Cache Size for Preload	500 MB
Dirty Block Rate for Write Delay Cache	50%
Coefficient of Monitoring Period (α)	1.2
Monitoring period for our method	520 sec
Monitoring period for PDC	30 min
TargetTH of DDR	450 IOPS

The initial value of the monitoring period was set to 520 sec, which is ten times the break-even time. After replacing I/O trace, the method changed the monitoring period dynamically as described in Section IV. The monitoring period of PDC was set to 30 min, which was described in [11].

The TargetTH of DDR was set to 450 IOPS, where TargetTH is used an indicator in DDR. TargetTH means the maximum IOPS of disk enclosures when an application has to satisfy the specified I/O throughput or I/O response time. Half of the maximum random IOPS of the disk enclosure was selected. The monitoring period of the DDR is the same value as described in paper [15].

C. Workload

For the evaluation, application-level I/O traces were obtained as the analysis of I/O behaviors of the data intensive applications described in Section VI. These I/O traces are replayed by our trace replay tool. The measurement interval of each application is equal to the duration of each application in Table I.

D. Results

1) File Server: Fig. 8 shows the average power consumption of the storage controller and disk enclosures used by the File Server application. As shown in the figure, the proposed method decreases power consumption of the disk enclosures from 2977.9 W to 2209.2 W, which is a decrease of 25.8%. PDC reduces the power consumption of disk enclosures to 2873.9 W, a decrease of 3.5%, and DDR reduces to 2869.7 W, a decrease of 3.6%.

Fig. 9 shows the average I/O response time measured at the application monitor. The I/O response time includes a wait time of powering on the disk enclosures and a degradation of response time caused by data migration. As shown in the figure, the average I/O response time of the proposed method is 17.1 ms. The average I/O response time of PDC is 22.6 ms, and DDR is 27.0 ms. The I/O response time of the proposed method is shorter than the I/O response time *without power saving*. This is because the proposed method preloads P1 data

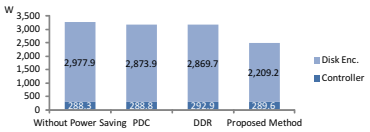

Fig. 8. Power Consumption for File Server

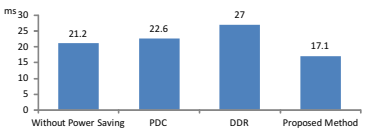

Fig. 9. Average I/O Response Time for File Server

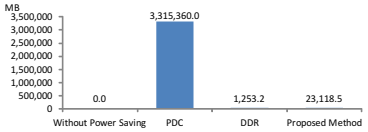

Fig. 10. Migrated Data Size for File Server

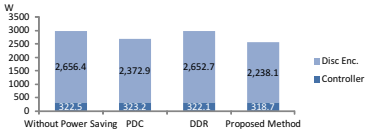

Fig. 11. Power Consumption for TPC-C

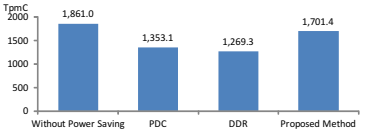

Fig. 12. Transaction Throughput for TPC-C

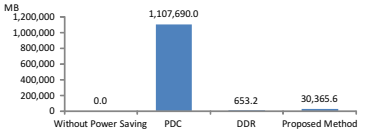

Fig. 13. Migrated Data Size for TPC-C

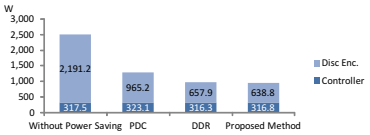

Fig. 14. Power Consumption for TPC-H

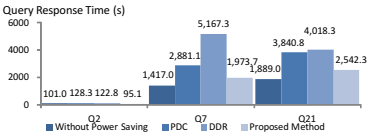

Fig. 15. Query Response Time for TPC-H

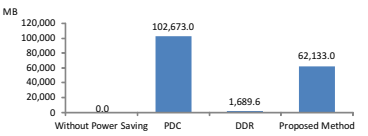

Fig. 16. Migrated Data Size for TPC-H

items, thus the number of read I/Os is reduced compared with *no power saving*.

Fig. 10 shows the total migrated data size during evaluation. As shown in the figure, the migrated data size for the proposed method is 23.1 GB. The method could reduce the migrated data size because it transfered only P3 data items from the cold disk enclosures to hot disk enclosures. The migrated data size of PDC exceeds 3 TB. This is because PDC also moves hot data between hot disk enclosures and cold data between cold disk enclosures. The migrated data size of DDR is 1.3 GB. The migrated data size of DDR is minimal compared to other methods. This is because the IOPS of all disk enclosures is higher than the LowTH of DDR (225, half of TargetTH). Therefore, DDR could not find any cold disk enclosures. Here, LowTH is an indicator of the disk enclosures in DDR. LowTH means the minimum IOPS of the disk enclosures that are considered as cold ones in DDR.

The duration of the work load was 6 hours. The number of data placement determinations for PDC is 11 and approximately 91,000 for DDR. The number for the proposed method is 5. The I/Os for File Server traces include many long I/O intervals, and the sequence distribution of these long intervals is not varied as time passes, which implies that a longer monitoring period is applicable. Thus, the proposed method recalculates the next monitoring period to be longer than the previous one. Therefore, our method can reduce the CPU cycles for determining data placement.

2) TPC-C: Fig. 11 shows the average power consumption of the storage controller and disk enclosures used by the TPC-C application. As shown in the figure, the proposed method decreases power consumption of the disk enclosures from 2656.4 W to 2238.1 W, a decrease of 15.7%. PDC reduces power consumption of the disk enclosures to 2873.9 W, a decrease of 10.7%, while DDR could not reduce the power consumption of the disk enclosures.

Fig. 12 shows the average transaction throughput for each method. The transaction throughput of the proposed method is 1701.4 tpmC, a 8.5% decrease. Transaction throughputs of PDC and DDR also decrease, and their degradation rate is higher than that of the proposed method. The reason for the observed difference is that the proposed method uses the preload function. Thus, the read response time of the method is shorter than that of other methods.

Fig. 13 shows the total transferred data size. As shown in the figure, the migrated data size in PDC exceeds 1 TB, and the migrated data size of DDR is a minimum. The reasons are the same as for the File Server case.

The duration of the work load was 1.8 hours. The number of data placement determinations with the proposed method was 7, 3 for PDC, and approximately 90,000 for DDR. The number of data placement determinations is higher than PDC, but the proposed method reduces the total migrated data size because it migrates only P3 data items from cold disk enclosures to hot disk enclosures.

3) TPC-H: Fig. 14 shows the average storage power consumption of storage controllers and the disk enclosures used by the TPC-H application. As shown in the figure, all methods can reduce power consumption of the disk enclosures by more than 50%. The proposed method decreases power consumption of the disk enclosures from 2191.2 W to 638.8 W, a decrease of 70.8%. PDC reduces power consumption of the disk enclosures to 965.2 W, a decrease of 55.9%, while DDR reduces power consumption to 657.9W, a decrease of 69.9%.

Fig. 15 shows the query response of Q2, 7, and 21. The query response times become worse for all methods, but the proposed method's query response is faster than those of

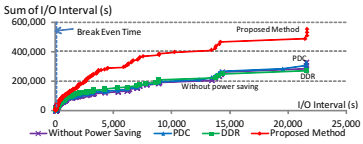

Fig. 17. I/O Intervals for File Server

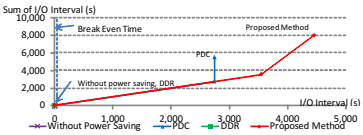

Fig. 18. I/O Intervals for TPC-C

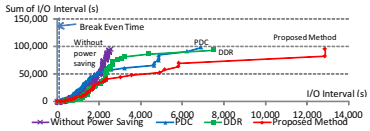

Fig. 19. I/O Intervals for TPC-H

PDC and DDR, because the proposed method uses a preload function; thus the response times of read I/O is faster than those of PDC and DDR. The query response time of DDR is approximately 3 times longer than that of the proposed method. This is because DDR migrates little data. TPC-H data are stripped and stored in all of the disk enclosures in our test bed, so that the number of times the disk enclosures are turned on is increased compared with that of other methods.

Fig. 16 shows the total migrated data size during evaluation. As shown in the figure, the proposed method and PDC migrate many data compared with DDR. This is because the hot data in cold disk enclosures are migrated to hot disk enclosures. The migrated data size of DDR is small, since DDR only migrates physical blocks in cold disk enclosures to hot disk enclosures when the physical blocks in cold disk enclosures are accessed. As described above, the TPC-H data are stored in all of the disk enclosures in the test bed. Thus, all of the disk enclosures are accessed during a query execution and become hot ones, and there are no cold disk enclosures. Thus DDR could not migrate physical blocks.

The duration of this work load was 6 hours. The number of data placement determinations in the proposed method is 10, 8 for PDC, and approximately 205,000 for DDR. The number of data placement determinations of the proposed method is higher than that of PDC, but the proposed method reduces the total migrated data size because it migrates only P3 data items onto cold disk enclosures.

E. Analysis

The proposed method can achieve a runtime power saving without sacrificing performance as shown above. However, it is still unclear to clarify the effect of the Logical I/O pattern which has only four patterns. The proposed method places data items into disk enclosures based on the Logical I/O pattern. The key point is the I/O intervals of disk enclosures. Therefore, we compared the tendency of I/O intervals. Fig. 17,18, and 19 show the relationship between I/O intervals and the total lengths of I/O intervals longer than the break-even time of the File Server, TPC-C, and TPC-H, respectively. The x-axis shows the length of the I/O intervals, while the y-axis shows the cumulative length of the I/O intervals.

As shown in Fig. 17, for File Server, the maximum length of the I/O intervals is approximately the same for all methods. However, the total length of I/O intervals in the proposed method is approximately twice as long as that compared with other methods. This result shows that the proposed power-saving method can reduce power consumption compared to other methods.

As shown in Fig. 18, for TPC-C, the I/O intervals of the method are longer than those of PDC and DDR. There are no I/O intervals longer than the break-even time in DDR. Our proposed method uses preloading and delaying writes using the storage cache, therefore the proposed method can use the application's I/O behaviors efficiently even though the workload is a very busy OLTP.

As shown in Fig. 19, for TPC-H, the I/O intervals of the proposed method are also longer than those of PDC and DDR. PDC and DDR could enlarge the I/O intervals. However, the proposed method can enlarge I/O intervals much longer than PDC and DDR. The reason is the same as that for TPC-C.

VIII. Related Work

A. I/O Interval Control

I/O Interval Control controls the I/O timing of an application to increase the probability that a disk drive is in a power-saving mode. A feature of this approach is to increase the idle period using hierarchical memory architecture such as cache memory [6], [7], [8], with varying application of I/O issue timing [9]. The features of these methods are to enlarge the I/O interval using a memory hierarchy as a storage cache: if the I/O request is read, then the method loads data into the cache, and if the data are written to the cache, then these data are bulk written to the disk drives.

The OLTP issues a random read. Therefore, it is difficult to prefetch data onto the storage cache only using disk enclosure-level I/O behaviors. On the other hand, the write to the disk enclosure is issued asynchronously to the transaction. Therefore, the write interval can be changed for these methods. However, the OLTP issues hundreds to thousands of I/Os per second. Therefore, the generation of an I/O interval that leads to power savings in the disk enclosure is difficult.

DSS issues a sequential read. Therefore, the storage can prefetch data into the storage cache. However, these methods do not use the application's I/O behavior and cannot decide on an appropriate size to prefetch. Therefore, the effect of power-saving by applying only this method is not so good.

B. Data Placement Control

Data Placement Control intends to reduce the power consumption of disk drives by controlling the data placement on disk drives. The concept of this approach is to concentrate frequently accessed data onto a few disk drives, then to move the status of other disk drives to standby or to sleep mode [10], [11], [12], [15], [16]. These approaches control physical block placement using physical I/O behaviors or control a file placement using logical I/O behaviors. The physical I/O

behavior-based approaches cannot find a power saving potential obtained by combining physical and logical I/O behaviors. In a large storage environment, a logical I/O behavior-based approach cannot recognize disk enclosures (units of power on and off) because these storage units are highly virtualized. Therefore, logical I/O-based data placement may not work well.

C. Application-Level Power Saving

A paper in the literature [21] describes traditional hardware-based power-saving methods as part of the solution and that data management software is important for power conservation for large data centers. The paper points out that algorithms or system configurations for high performance are not optimal from the viewpoint of power-saving. The paper also describes that tuning for power-saving or resource consolidation of DBMS is necessary. The paper [21] did not discuss power-saving efficiency or the impact on performance of DBMS.

Another paper [22] shows the relationship between performance and power consumption of TPC-C when varying the disk enclosure configuration (number of disks and media), power control function of CPU, or memory size. However, as described above, the report [22] avoids discussion of power-saving after the data center is implemented.

D. Energy Saving for Solid State Disks

Another paper [16], [23] describes storage power saving using Solid State Disks (SSDs). Power consumption of SSDs is much smaller than that of HDDs. Since our proposed approach utlizes the application's I/O behaviors, and therefore, it can be applied easily to SSD storage.

IX. Conclusions

This paper presents a power-saving framework that can provide the ability to use the application's I/O characteristics based on a novel application-collaborative storage power-saving model. The power-saving-enabled I/O patterns are defined to be of four types that combine storage-level I/O behaviors with application-level ones. An investigation of the appropriate power-saving method for each data item was performed. The I/O behaviors and energy consumption of multiple enterprise workloads running at an actual data center, such as File Server, OLTP, and DSS, was examined. The results were compared with conventional storage power-saving methods, Popular Data Concentration (PDC) and Dynamic Data Reorganization (DDR). For all applications, the power reduction of the proposed method is higher than or equal to PDC and DDR results. Furthermore, the application performance with our method is not degraded as much as compared with that of no power savings. The results show that the proposed method of application-collaborative power-saving method makes a large contribution to the power-saving of the data center.

Future work will improve and complete the implementation of the power-saving system in an actual data center with multiple energy saving methods. As well, the proposed methods will be applied to petabyte-scale databases to examine the effectiveness of the system on different configurations.

References

[1] F. Gens, "Idc predictions 2011: Welcome to the new mainstream," IDC White Paper #225878, 2010.

[2] R. L. V. Natalya Yezhkova, "Worldwide enterprise storage systems 2010-2014 forecast update," IDC White Paper # 226223, 2010.

[3] A. G. Yoder, "Green storage technologies, capex and opex," Storage Networking World 2011 Spring, 2011.

[4] S. Worth, "Green stroage - the big picture," Storage Networking World 2011 Spring, 2011.

[5] M. Poess and R. O. Nambiar, "Energy cost, the key challenge of today's data centers: a power consumption analysis of tpc-c results," in *VLDB '08 Proceedings*. ACM Press, 2008, pp. 1229–1240.

[6] A. Papathanasiou and M. Scott, "Energy efficient prefetching and caching," in *Proc. of USENIX 2004 Annual Technical Conference*. USENIX Association Berkeley, 2004, pp. 255–268.

[7] D. Li and J. Wang, "Eeraid: Power efficient redundant and inexpensive disk arrays," *Proc. 11th Workshop on ACM SIGOPS European Workshop*, pp. 174–180, 2004.

[8] X. Yao and J. Wang, "Rimac: A novel redundancy based hierarchical cache architecture for power efficient," in *High Performance Storage System Proc. 2006 EuroSys Conference*, 2006, pp. 249–262.

[9] T. Heath, E. Pinheiro, J. Hom, U. Kremer, and R. Bianchini, "Application transformations for power and performance-aware device management," in *11th International Conference on Parallel Architectures and Compilation Techniques*, 2002, pp. 121–130.

[10] D. Colarelli and D. Grunwald, "Massive arrays of idle disks for storage archives," in *Supercomputing, ACM /IEEE 2002 Conference*, 2002, pp. 47–57.

[11] E. Pinheiro and R. Bianchini, "Energy conservation techniques for disk array based servers," in *Proc. 18th Annual International Conference on Supercomputing*. ACM, 2004, pp. 68–78.

[12] O. Weddle, C., Q. M., J., and A. Wang, "Paraid: A gear-shifting power-aware raid," in *5th USENIX Conference on File and Storage*. USENIX Association, 2007, pp. 245–267.

[13] D. Narayanan, A. Donnelly, and A. Rowstron, "Write off-loading: Practical power management for enterprise storage," in *6th USENIX Conference on File and Storage Technologies*. USENIX Association, 2008, pp. 253–267.

[14] K. Verma, A., L. R., Useche, and R. Rangaswami, "Srcmap: Energy proportional storage using dynamic consolidation," in *8th USENIX Conference on File and Storage Technologies*. USENIX Association, 2010, pp. 267–280.

[15] E. Otoo, D. Rotem, and S.-c. Tsao, "Dynamic Data Reorganization for Energy Savings," in *Proceedings of the 22nd international conference on Scientific and statistical database management*, 2010, pp. 322–341.

[16] J. Guerra, H. Pucha, J. Glider, W. Belluomini, and R. Rangaswami, "Cost effective storage using extent based dynamic tiering," in *9th USENIX Conference on File and Storage Technologies*. USENIX Association, 2011.

[17] "Tpc-c, an online transaction processing benchmark," Transaction Processing Performance Council, 2010. [Online]. Available: http://www.tpc.org/tpcc/

[18] "Tpc benchmark^{TM}h standard specification revision 2.14.2," Transaction Processing Performance Council, 2011. [Online]. Available: http://www.tpc.org/tpch/

[19] A. Brunelle, "btrecord and btreplay user guide," 2010. [Online]. Available: http://www.cse.unsw.edu.au/ aaronc/iosched/doc/btreplay.html

[20] "Hitachi adaptive modular storage 2500 datasheet," Hitachi Data Systems, 2011. [Online]. Available: http://www.hds.com/assets/pdf/hitachi-datasheet-ams2500.pdf

[21] S. Harizopoulos, M. Shah, J. Meza, and P. Ranganathan, "Energy efficiency: The new holy grail of data management systems research," in *4th Biennial Conf. on Innovative Data Systems*, 2009, pp. 112–123.

[22] M. Poess and R. Nambiar, "Tuning servers, storage and database for power efficient data warehouse," in *26th IEEE International Conf. on Data Engineering*. IEEE Computer Society, 2010, pp. 1006–1017.

[23] S. M. Snyder, S. Chen, P. K. Chrysanthis, and A. Labrinidis, "Qmd: exploiting flash for energy efficient disk arrays," in *DaMoN*, S. Harizopoulos and Q. Luo, Eds. ACM, 2011, pp. 41–49.