論文 / 著書情報
Article / Book Information

Title	Effects on Performance and Energy Reductionby File Relocation based on File-access Correlations
Author	Masaru IRITANI, Haruo YOKOTA
Journal/Book name	Proc. of the 1st Workshop on Enegy Data Management (EnDM2012), , ,
Issue date	2012, 3
DOI	http://dx.doi.org/10.1145/2320765.2320794
Copyright	Copyright (c)2012 Association for Computing Machinery
Set statement	(c) ACM, 2012. This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in Proceedings of the 2012, Proc. of the 1st Workshop on Enegy Data Management (EnDM2012), , , , http://dx.doi.org/10.1145/2320765.2320794
Note	このファイルは著者（最終）版です。 This file is author (final) version.

Effects on Performance and Energy Reduction by File Relocation based on File-access Correlations

Masaru IRITANI
Tokyo Institute of Technology
Ookayama 2-12-1, Meguro
Tokyo, Japan
iritani@de.cs.titech.ac.jp

Haruo YOKOTA
Tokyo Institute of Technology
Ookayama 2-12-1, Meguro
Tokyo, Japan
yokota@cs.titech.ac.jp

ABSTRACT

Many recent advanced applications of information systems require large amounts of data to be stored in multiple hard disk drives (HDDs). The placement of data in these drives is quite influential in reducing energy consumption while retaining the performance of the system. Several approaches have been proposed in which frequently accessed files are concentrated into a limited number of drives to enable spin-down of the other drives. However, the placement of infrequently accessed data in these drives is also significant because energy consumption during spin-up to access these data cannot be ignored, particularly when files tending to be used together are placed on many different drives. In this paper, we propose a novel method named PLECO (Placement of files for Latency and Energy Consumption Optimization), which aims to place correlated files into the same drive to reduce power consumption and improve the performance of the system. We simulated and evaluated the proposed method, and these results indicate that our proposal can reduce both the energy consumption and the access latency by up to 32% and 92%, respectively, compared with a baseline system.

General Terms

Energy Reduction

Keywords

File system, File correlation mining, Power saving

1. INTRODUCTION

In recent years, as the amount of data handled by advanced applications of information systems has been increasing rapidly, the energy consumed in storing these data has also become large. For example, EPA (U.S. Environmental Protection Agency) [1] reported that the number of installed enterprise hard disk drives (HDDs) in the U.S. in 2010 would be more than 3.7 times the number used in 2004. They also estimated that the energy consumption for enterprise storage devices had tripled between 2000 and 2006. Therefore, the reduction of energy consumption by storage systems is one of the most important issues for information systems, particularly when handling large amounts of data, such as in data centers or file servers.

To reduce the energy consumption of a storage system comprising multiple HDDs, several approaches based on the skewed distribution of file-access frequency have already been proposed. These approaches divide the HDDs in a system into two groups, and place frequently accessed files on HDDs in one group by migrating or copying files from HDDs in the other group [7]. By such a placement, the number of accesses to HDDs in the latter group become infrequent, with many of them having the chance to spin-down (stop the rotation of the spindle motor), which is an effective means of power reduction.

However, these approaches have two shortcomings. When a file stored in an inactive HDD needs to be accessed, a spin-up process is required to make the read or write operation available on the HDD. It can take a long time (15–20 ms) for the rotation to become stable and enable the operation to be completed. Moreover, compared with normal disk accesses, a HDD consumes large amounts of electricity when it spins up. Therefore, frequent spin-downs and spin-ups incur long access-latency times and large energy consumption.

If a number of files accessed at the same time are scattered around multiple HDDs in the inactive group, these shortcomings will become even more serious because the simultaneous spin-ups will raise the peak power consumption of the storage system. On the other hand, if those files tending to be accessed at the same time are placed in the same HDD, the number of spin-ups can be reduced. Such placement is effective, not only for reducing the total power consumption of the storage system but also for improving its performance by omitting the access latency originating from irrelevant spin-ups.

In this paper, we propose a novel method for migrating those files tending to be accessed at the same time into the same HDD to achieve better energy consumption and performance of the storage system. For this purpose, it is important to identify files accessed at the same time. We introduce the concept of a task to represent a group of files used to accomplish some particular goal, such as writing a report or preparing presentation slides. This implies that a task will contain correlated files based on the behavior of its users. The proposed method mines file-access logs to discover tasks, and tries to place the files belonging to the same

EnDM 2012, March 30, Berlin, Germany.

task on the same HDD in the inactive group.

We evaluated the proposed method via simulation. The simulation results indicate that our proposal can reduce both the energy consumption and the access latency by up to 32% and 92%, respectively, compared with a baseline data-placement system.

The remainder of this paper is organized as follows. Section 2 briefly introduces related work to provide a background. As the basis of our proposal, Section 3 explains some assumptions made about storage systems. In Section 4, we propose a method for placing correlated files on the same HDD by mining file-access logs. Section 5 evaluates the proposed method in comparison with some baseline methods, and reports on evaluation results, indicating that the proposed method is effective for improving system performance and reducing energy consumption in storage systems. Section 6 contains the conclusions of this paper and points to future work.

2. RELATED WORK

2.1 Energy-conscious Storage Systems

There are a number of methods for reducing the energy consumption of a storage system comprising multiple HDDs. Among them, the Massive Array of Idle Disks (MAID) [7] method is well known. MAID employs a number of HDDs called cache disks that rotate continuously and store recently accessed data. This concentrates most accesses to them, by analogy with cache memory. RAPoSDA [15] enhances the concept of MAID by including the primary-backup configuration and considering the rotational states of each HDD to make the system more reliable and to reduce energy consumption further by utilizing the backups. Whereas MAID and RAPoSDA adopt least-recently-used (LRU) placement, in this paper, we propose a dedicated data- placement strategy by analyzing file-access histories.

A framework that uses reinforcement learning to optimize file accesses is proposed. Vengerov [18] proposed a method that observes the number of file input/output (I/O) requests in the queue at each tier of a hierarchical storage system, and tunes parameters for estimating the effect of migration using reinforcement learning. However, this was aimed at minimizing the file-access times and did not consider energy consumption.

Power-saving storage systems based on RAID [14] technology have been also proposed. PARAID [19] introduces the concept of changing gears, which varies the number of active HDDs to make energy consumption proportional to performance. GRAID [13] places an emphasis on ensuring reliability and power saving based on RAID10 disk arrays. Using information in the disk which is added into the normal disk to store logs, the system only needs to update the mirror disks periodically. Therefore the system can spin-down all the mirror disks to a low-power mode for most of the time and save energy. EERAID [12] is another power-saving method that is focused on the RAID controller layer. EERAID reduces the power consumption using dynamic I/O scheduling and a cache-management policy. Our approach can also be applied to RAID-based storage systems, but focuses on file placement within storage systems, rather than on block accesses.

Distributed storage systems employing the concept of power proportionality through leveraging cluster-based data placement with data replication, such as Sierra [17] and Rabbit [4], have been proposed. These methods achieve a proportional relation between power consumption and system performance by dividing all storage nodes into groups and deciding which groups actively serve certain workloads. Our approach does not aim for power proportionality.

Other approaches to reducing power consumption in a storage system involve changing the rotational speed of HDDs. DRPM [10] controls the rotational speed of a HDD dynamically, depending on how heavy its workload is. However, HDDs capable of dynamic control of their rotational speed are not yet available commercially. Therefore, this method currently cannot be implemented in real-world service. Hybernator [23] leverages multispeed HDDs. Like DRPM, a multispeed disk can change its rotational speed, although the changes are stepwise and not continuous. While keeping the rotational speed low makes it possible to reduce energy consumption by the HDD, changing speed itself consumes more energy than the other states. Therefore, changing the speed frequently might have a bad effect on the energy consumption. This method adopts a coarse-grain response when it observes a workload, to suppress the number of extra speed changes.

Whereas methods such as DRPM and Hybernator require special hardware devices to control energy consumption dynamically, we assume the use of ordinary, commercially available HDDs for the proposed method. However, our method could also be applied to any storage system that aims to reduce energy consumption by switching between states in HDDs.

2.2 Log Mining to Find File Correlations

In our approach, we employ correlations between files to guide file placement in a storage system to reduce its energy consumption. To derive the file correlation, there are several approaches to mining access logs.

To group files into virtual directories corresponding to user tasks, the FI and SUGOI methods [20] apply data-mining techniques to file-access logs. They were originally proposed to support file searches in file systems, but we adopted this approach in this paper for energy-consumption purposes. The details of these methods are described in Section 4.

FARMER [21] is another mining method for finding correlations between files from file-access logs. It collects attributes of accessed files such as timestamps, file paths, and users, and calculates the co-occurrence of attributes between files.

XAPC [22] observes query patterns issued for an XML tree structure to reduce the energy consumption of HDDs by utilizing the relationships between XML data. It applies a data-mining algorithm to uncover relationships between parts of the data structure in the XML tree. It then migrates these XML data into multiple HDDs based on the relationships between query patterns. It involves data placement based on correlations, but focuses on logs of queries, not file accesses.

Several methods aim to derive relationships between files based on access sequences to predict access patterns and to prefetch files. The baseline SUCCESSOR method logs subsequent accesses for each file. Griffioen et al. [9] proposed a method based on a probability graph that keeps multiple successors for each file in the form of weighted edges. Noah [3] is an extension of the SUCCESSOR method that aims to

filter out noise patterns by updating the successor to each file only if the new successor can be regarded as reliable. The Finite Multi-Order Context model [11] has been proposed for file-access pattern prediction. It originates from the contextual prefix tree (TRIE) of the PPM text compression algorithm [6]. It tracks the access sequences in a TRIE whose multiple partitions are limited in size.

3. ASSUMPTIONS

In this section, we describe the assumptions about storage systems that we use in applying our method.

3.1 States of a HDD

We assume that target storage systems comprise multiple HDDs for file storage. Most commercially available HDDs have three states, namely "ACTIVE", "IDLE", and "STANDBY".

The ACTIVE state is the state in which data in the HDD are being accessed. Because its spindle motor, actuator, and controller are all working in this state, this state consumes more power than the other two states. After completing the data access, the HDD changes its state to IDLE. The HDD keeps its platters rotating, but activates neither its actuator nor its read/write controller. Therefore, power consumption of a HDD in the IDLE state is less than that in the ACTIVE state.

A HDD can stop its platter rotation to change its state to the STANDBY state. In the STANDBY state, the HDD stops rotating and moves its disk head away from the platters. This state requires much less power than the IDLE state. Therefore, we can reduce the energy consumption of HDDs if we keep most of them in the STANDBY state as much as possible.

However, the state transition from STANDBY to IDLE, or vice versa, which involves spin-up or spin-down of its spindle motor, takes a relatively long time compared with the transition between the ACTIVE and IDLE states. For example, a commercially available HDD [16] typically takes 15 seconds to spin-up, because the HDD waits until its rotation becomes stable before read/write operations are acceptable. Moreover, it consumes large amounts of electricity to stop or to resume the rotation of a HDD, even compared with the ACTIVE state. For these reasons, trying to stop and start disk rotation aggressively may lead to bad effects on both the file-access latency and the energy consumption by disks because the number of disk spin-ups would increase. To conclude, there are two important issues in reducing energy consumption by a storage system. One is to stop the rotations of most of the HDDs, and keep them in the STANDBY state as long as possible. The other is to suppress the number of unnecessary spin-ups caused by an overaggressive spin-down policy.

3.2 Storage System Configuration

The target storage system comprises multiple HDDs. We do not assume special functions for the HDDs, except that the HDD is able to spin-down (stop rotation of its spindle motor) if there is no access during a specified time period. A HDD in the STANDBY state will spin-up when it receives an access request, and enter the ACTIVE state.

We divide HDDs in the target system into two groups, placing frequently accessed files in one group and the remaining infrequently accessed files in the other group. In this paper, we call a HDD in the former group a high-frequency disk, and a disk in the latter group a low-frequency disk. Whereas high-frequency disks are not spun down, a low-frequency disk will be spun down when it has had no accesses for the predetermined period of time.

The file system manages placement information and the metadata for each file to locate the files in the HDDs. The file metadata contains information about the access frequency for the file. For the access-frequency information, we use the number of accesses during a specified time span in this paper. We assume that these data are managed by some index structure such as a B-tree [5].

3.3 Motivating File-access Patterns

If there are a number of files related to a task and they are scattered across different low-frequency HDDs that have already been spun down as a result of no access during the specified time period, all these HDDs holding related files will have to be spun up when the user accesses these files as a part of the task later. This will lead to rather large access latency and will consume a large amount of energy.

On the other hand, if these related files are located in one disk, the number of spin-ups for the task will be reduced. As a result, the access latency and the energy consumption will be reduced. This will be effective for cases in which the access frequency of files changes periodically.

Files possibly used for the same task tend to be accessed nearly simultaneously. Moreover, files accessed nearly simultaneously several times can be assumed to be used for the same task. Therefore, these files will be accessed nearly simultaneously in the future. The proposed method utilizes this characteristic of file-access patterns.

4. PLECO

Here, we describe the proposed PLECO (Placement of files for Latency and Energy Consumption Optimization) method in detail. It extracts relationships between files from the file-access log for a file server. It utilizes these relationships to determine the appropriate HDD placement for infrequently accessed files.

PLECO relocates infrequently accessed files which have a strong relationship to the same low-frequency disk to reduce the number of spin-ups caused by scattered accesses among low-frequency disks.

4.1 File-placement Strategy

To place files accessed nearly simultaneously into the same HDD, PLECO migrates files between HDDs. It migrates a file to a low-frequency disk if it has not been accessed for a time longer than the predetermined threshold time. It also periodically migrates files without accesses for a certain period of time by observing the list of files without accesses.

The storage system does not execute relocation immediately. To suppress the number of spin-ups triggered by relocation processes, it keeps files to be relocated in a relocation buffer associated with the destination disk. It tries to execute relocation processes when the disks required for relocation are active for as long as possible.

4.2 File Correlation Discovery

The FI and SUGOI methods [20] use file-access logs to find association rules between files using a data-mining algorithm. These methods were originally proposed to support

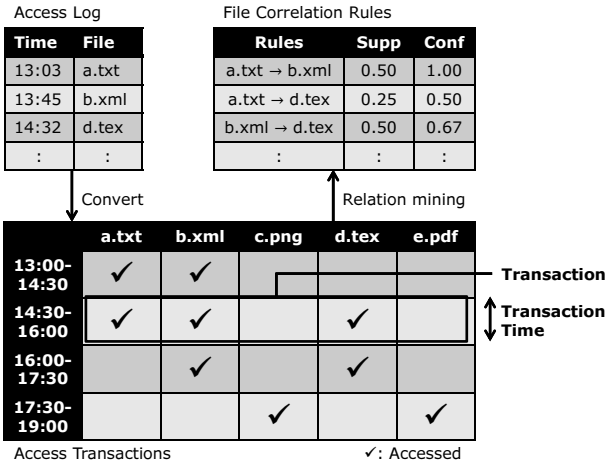


Figure 1: Mining file-correlation rules from access logs

file searches. The basic idea is that files used within the same time period are related to each other. Using extracted rules, these methods classify the files and represent them as virtual directories to the user.

Our idea is to apply the FI method to find relationships between files for file relocation. We briefly describe how to calculate relationships between files in the FI method.

Figure 1 illustrates the overall flows for discovering file-correlation rules. Firstly, the FI method collects the file-access logs for the storage system. The file-access logs should contain at least the timestamp and the ID of the requested file for each access issued by a client. It then splits the file-access logs into "transactions" based on the timestamps. The transaction duration is the range of the timestamps for all the requests in each transaction. The transaction duration for each transaction is the same.

To derive the association rules from these transactions, the FI method adopts a data-mining algorithm. In this paper, we describe our use of the Apriori algorithm [2] as the data-mining algorithm for PLECO. Other data-mining algorithms for discovering association rules could also be used. Here, an association rule for file access, i.e., a file-correlation rule, is represented by the form $\{lhs\} \rightarrow \{rhs\}$, which indicates that transactions containing items in $\{lhs\}$ also tend to have items in $\{rhs\}$. In our proposal, only rules whose length is two (i.e., $\{lhs\}$ and $\{rhs\}$ each have only one file) are used. Each rule indicates that both of two files in the rule have been accessed in a certain number of transactions.

The Apriori algorithm uses two measurements about occurrence possibilities to discover the rules. Support is the probability that all items in both left-hand and right-hand sets appear in a transaction. Confidence is the probability that all items in the right-hand set appear in a transaction when all items in the left-hand set appear. The Apriori algorithm filters file-correlation rules by setting thresholds for these two measurements. The thresholds for support and confidence are called the minimum support and the minimum confidence, respectively. The algorithm does not extract a rule whose support or confidence is less than these threshold values. Therefore, we can control the number of discovered file-correlation rules by setting the appropriate values for minimum support and minimum confidence.

4.3 Rule Application to Migration

Using the file-correlation rules derived by the FI method, PLECO chooses the best HDD among all the low-frequency disks to locate each file.

PLECO uses these rules when it migrates files from a high-frequency disk to a low-frequency disk. Because considering file locations between multiple disks that are always active does not help these disks reduce the number of spin-downs and spin-ups, it does not utilize rules for migration in the reverse direction. Instead, it determines locations for the files using the hash code of each file.

When PLECO relocates a file from a high-frequency disk to a low-frequency disk, it determines the destination disk for each unpopular file on using the relation score calculated for each low-frequency disk. All relation scores are initialized to 0, and increased by the iterative process described below.

Firstly, PLECO collects all rules that have a file to be relocated in their right-hand set. For each rule, it checks if the file in its left-hand set is in the relocation buffer of a low-frequency disk. If the file is in one of the relocation buffers, PLECO increases the relocation score of the disk which corresponds to the relocation buffer by the confidence value of the rule between it and the file. Then, to migrate the file, PLECO places it in the relocation buffer that corresponds to the HDD with the maximum relation score.

Figure 2 gives an example of the association-rule application during a file relocation. The file named "e.pdf" (shown as "e" in the figure) is now to be relocated. There are two candidate destination disks for the file. Each disk has two files to be relocated in its relocation buffer, with the left-hand disk in the figure having "a.txt" and "b.xml" and the right-hand disk having "c.png" and "d.tex". There are four rules that represent the relationships between these files and "e.pdf". In this case, the relocation scores for the left-hand and right-hand disks are 0.8 and 0.9, respectively, and PLECO therefore decides to place "e.pdf" in the right-hand disk.

PLECO applies rules which files in their left-hand set are in relocation buffers. It aims to prevent performance degradation by scanning all the metainformation about files to find locations for all files in related rules. This omission of rules can be justified because files with similar access patterns tend to be relocated nearly simultaneously.

5. EVALUATION

We evaluated the proposed method by a simulation experiment, using a software disk simulator that we developed. This simulates multiple accesses to different HDDs, and outputs the processing time and energy consumption for each access request.

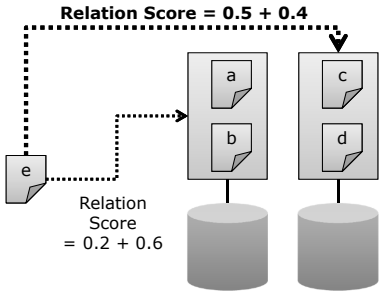
5.1 Setting

The parameters used in the simulation for the disk model, system configuration, and workload are as follows.

5.1.1 Disk Model

The target system in the evaluation had 16 HDDs for storing files. We adopted a disk model by reference to the specification of the commercially available HDD shown in Table 1.

5.1.2 System Configuration



Rules	Supp	Conf
a.txt → e.pdf	0.2	0.2
b.xml → e.pdf	0.3	0.6
c.png → e.pdf	0.1	0.5
d.tex → e.pdf	0.3	0.4

Figure 2: Applying association rules for a relocation

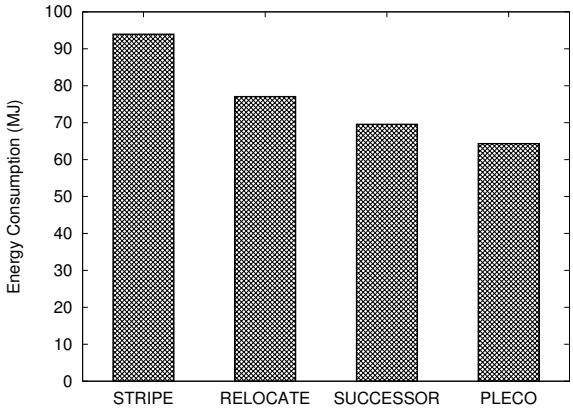


Figure 3: Energy-consumption comparison for the various placement strategies

Table 1: The specification of the HDD model

Referenced model	HGST Deskstar 7K2000 [16]
Capacity	2 TB
Cache Size	32 MB
Cache Replacement	LRU
Power on ACTIVE	11.1 W
Power on IDLE	7.5 W
Power on STANDBY	0.8 W
Energy for SPIN-DOWN	35.0 J
Energy for SPIN-UP	450.0 J
Time for SPIN-DOWN	0.7 sec.
Time for SPIN-UP	15.0 sec.

We assumed that the system could handle up to 32 requests simultaneously. If there are more than 32 simultaneous requests, all except the 32 first requests are blocked. A request that requires access to a disk being issued another request is also blocked until the other request finishes its access.

Initially, the files in the system are randomly scattered. To compare the effectiveness of the various relocation methods, the file server simply swaps files, keeping the number of files in each disk the same.

5.1.3 Workload

We used an open trace of an actual NFS file server in the EECS Department of Harvard University [8] as the workload for the evaluation. In this experiment, we extracted records of READ and WRITE requests from the trace between September 1st, 2001, and September 10th, 2001.

5.2 Competitors

To compare energy consumption and access latency, we implemented four methods, including PLECO, that place files in multiple HDDs.

A baseline method named STRIPE simply distributes files based on the hash code of each file. It does not attempt any file relocation. To compare its energy consumption to methods that utilize two classes of disks (high-frequency and low-frequency disks), our STRIPE implementation maintained two HDDs as always active, and spun down any of the 14 HDDs without accesses for 30 seconds.

RELOCATE is a method for relocating files based on their access frequency. It maintains a list of files that have not been not accessed and periodically relocates files in the list to disks with infrequently accessed files. In this experiment, we configured a RELOCATE system with two high-frequency disks and 14 low-frequency disks. It observes the states of all disks and tries to execute relocation when both the source disk and the destination disk are spinning. This helps reduce the number of spin-ups invoked by the file-relocation processes. RELOCATE determines the destination disk for relocated files using the hash code of each file.

The last-successor method is known as a basic file prefetching method, which logs the file accessed just after it is accessed [11]. We modified the last-successor method for file correlation extraction to examine the characteristics of file correlations extracted by our method. SUCCESSOR is a method based on the last-successor. It assumes that each file has a relationship to its successor file. In other words, SUCCESSOR treats an edge of the access sequence $A \rightarrow B$ as an association rule $\{A\} \rightarrow \{B\}$ whose confidence score would be 1.0 in the PLECO method.

PLECO behaves similarly to RELOCATE but uses information about the access relationships between files in the file-relocation process. It recalculates file correlations using the Apriori algorithm every 24 hours. We chose 90 minutes as the length of each transaction. We assumed that the data-mining process can be executed as a background process, with no adverse effects on the services of the file server. Although the process has to read the file-access log once, there are only a few of these processes compared with the number of file accesses to the server, and we assumed that the effect of this process on energy consumption would be negligible. Therefore, we did not include the execution and energy overheads of the data-mining process.

5.3 Energy Consumption

Energy consumption by the HDDs for the various placement strategies is compared in Figure 3.

As shown in the figure, PLECO consumes less energy than the other three methods. STRIPE consumed over 90 MJ, which is the highest energy consumption among the methods. As described in Section 5.2, the STRIPE method does not attempt any file migration between disks. Many popular

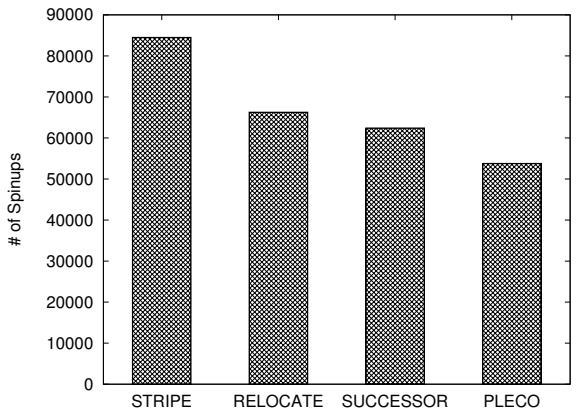


Figure 4: Comparison of the number of spin-ups

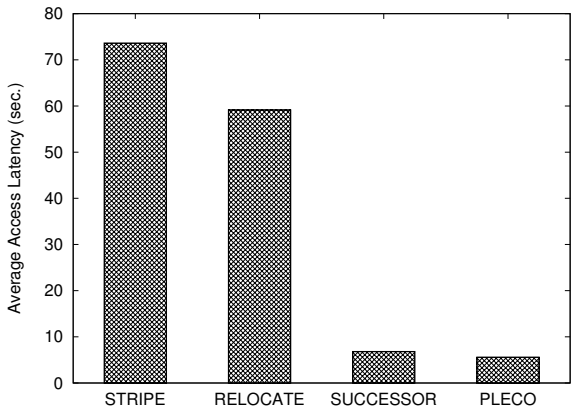


Figure 5: Comparison of the average access latency

files were put on one of the low-frequency disks, where they remained. As a result, accesses to these popular files caused a number of spin-ups.

By relocating files based on their access frequencies, the RELOCATE method reduced the energy consumption of disks by 18% compared with the naive STRIPE method. This is because the number of spin-ups caused by accesses to popular files was decreased by moving those files to always-active disks. Furthermore, the SUCCESSOR method required less energy than the RELOCATE method by utilizing file correlations.

Although RELOCATE and SUCCESSOR reduced the energy consumption required by disks to some extent, PLECO consumed much less energy even than them, with an energy reduction of 32% compared with the naive STRIPE method. PLECO even achieved 8% less energy consumption than the SUCCESSOR method.

To investigate the relationship between energy consumption and disk spin-ups, we counted the number of spin-ups during the simulation. Figure 4 shows the results, which indicate the same tendencies as the energy consumption result in Figure 3. This confirms that reducing the number of HDD spin-ups reduces the energy consumption.

5.4 Access Latency

We measured the average file-access latency to evaluate the performance of each data-placement strategy. The average access latency is defined as the amount of time required to respond to an file-access request when it is issued by a client. Figure 5 compares the access latency for the four methods.

These results show the same tendency as the energy consumption comparison described above. Because there are a large number of accesses in very short periods of time in the workload, simply scattering files across the disks can easily increase the average access latency. In fact, STRIPE required over 70 seconds on average.

This serious access latency is caused by a number of requests being issued nearly simultaneously. In such a situation, most requests are blocked until the server finishes processing the preceding access requests, which may require some disk spin-ups.

By migrating files depending on their access frequency and consolidating disk accesses in high-frequency disks, RELOCATE can reduce the average access latency by 20%. However, it remains high because RELOCATE cannot completely eliminate cases where accesses are distributed to multiple low-frequency disks. If there are many accesses to files that have not been accessed for a while, the system may have to spin-up many low-frequency disks. Therefore, RELOCATE can have the same problems as the STRIPE method.

The experimental results indicate that utilizing file correlations can be one of the effective ways of solving this problem. SUCCESSOR can respond to file accesses in only 12% of the time required by RELOCATE. SUCCESSOR is able to uncover relationships between files to some extent, and these file correlations help the file system consolidate the files that tend to be accessed nearly simultaneously. This is the reason for SUCCESSOR having a reduced average access time compared with STRIPE and RELOCATE.

PLECO has the best results in this evaluation. PLECO only requires 7.6% of the time required by STRIPE and is faster than SUCCESSOR by 18%.

5.5 Effects of Parameters

Two parameters control PLECO's behavior, namely minimum support and minimum confidence. To examine the effects of these two parameters, we evaluated PLECO with various combinations of them. We used minimum support values of 0.10, 0.15, and 0.20, and minimum confidence values of 0.2, 0.4, 0.6, and 0.8.

The evaluation results are shown in Figures 6–8. These three results show similar characteristics, confirming that the number of HDD spin-ups is strongly related to their energy consumption and performance.

As these results show, lower confidence tends to result in better energy consumption and access latency. A higher confidence means that the association rules for files extracted from the file-access log become more reliable while the number of rules decreases. For minimum support values, the results show similar characteristics. According to these results, applying as many rules as possible, including possibly unreliable rules, is considered effective in most cases for reducing energy consumption and access latency.

However, the idea to utilize as many rules as possible does not always give the best result. For example, when the minimum support is 0.15, the best confidence value among the four choices is 0.4 rather than 0.2. As described in 4.3, PLECO calculates the relocation scores for each disk by ac-

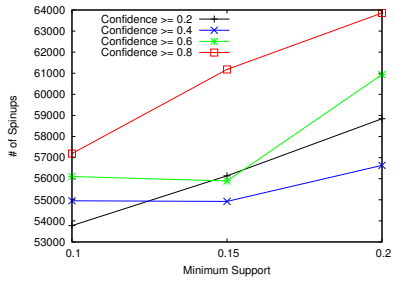


Figure 6: Number of spin-ups for PLECO

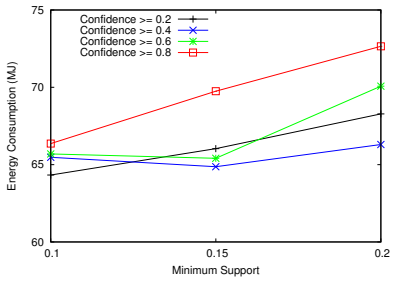


Figure 7: Energy consumption for PLECO

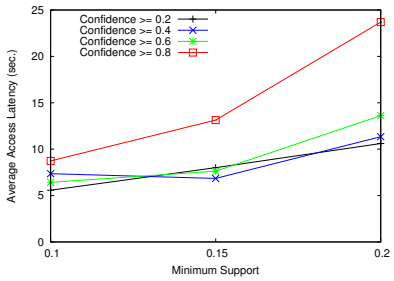


Figure 8: Average access latency for PLECO

cumulating the confidence values of all related rules. If the number of unreliable rules is relatively large, these wrong rules may overwhelm the few reliable rules.

To conclude, whereas lower thresholds of support and confidence values result in better energy consumption and performance in most cases, overly low values of minimum support and minimum confidence may lead to the inappropriate application of unreliable rules in some cases. Finding optimal threshold values based on the behavior of accesses is included in our plans for future work.

5.6 Discussion

In this evaluation, all results confirm that reducing the number of spin-ups helps the file system reduce the energy consumption and the access latency.

In the experiments, we found that migrating files based on file-access frequency alone can reduce energy consumption and improve performance. However, these results also show that it is not always sufficiently effective, particularly when there are many accesses to file groups distributed across multiple low-frequency disks. Utilizing file correlation can handle this problem and showed better performance with less energy consumption.

For both energy consumption and access latency, the proposed PLECO method achieved the best results of all competing methods. This is because PLECO can detect relationships between files and utilize this information in migration processes better than a file-prefetching method alone. A prefetch method using access sequences such as SUCCESSOR can find inaccurate correlations because the sequential access to two files does not always mean that there is a strong relation between them. PLECO uses an association-mining method, which can find access patterns that frequently occur, giving file-correlation rules that are more reliable than those of methods that use only access sequences.

6. CONCLUSION AND FUTURE WORK

In this paper, we proposed a novel data placement method called PLECO for reducing the power consumption and improving the performance of a storage system comprising multiple HDDs. It derives association rules by mining file-access logs to discover correlations between files and relocates the correlated files to the same HDD, thereby eliminating irrelevant HDD spin-ups.

The evaluation results, from simulations that used an open trace of an actual NFS file server, indicated that PLECO provided the best results when compared with a baseline method and successor-based methods with respect to energy consumption by the storage system and the average access latency. Compared with the baseline system, PLECO reduced the energy consumption of the system and the average access latency by 32% and 92%, respectively.

In our evaluation, we assumed that there is no cost in calculating or managing the association rules. Although the Apriori algorithm is efficient, it does consume a certain amount of time. This cost may not be negligible, particularly when there are many files in the file system. In future work, we plan to manage correlation rules between files in a distributed data structure.

We evaluated our method using a set of file-access logs. The optimal value of the parameters of the proposed method may depend on the behavior of the file accesses in its workload and may vary with workload to some extent. Evaluating the proposed method with different workloads will be required to discover any changes in optimal parameter values caused by changes in workload and to discover a way to optimize these values.

Acknowledgments

This work is partly supported by Grants-in-Aid for Scientific Research from the Japan Science and Technology Agency (A) (#22240005).

7. REFERENCES

[1] Report to congress on server and data center energy efficiency public law 109-431. Technical report.

[2] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, SIGMOD '93, pages 207–216, New York, NY, USA, 1993. ACM.

[3] A. Amer and D. Long. Noah: low-cost file access prediction through pairs. In *Performance, Computing, and Communications, 2001. IEEE International Conference on.*, pages 27 –33, apr 2001.

[4] H. Amur, J. Cipar, V. Gupta, G. R. Ganger, M. A. Kozuch, and K. Schwan. Robust and flexible power-proportional storage. In *Proceedings of the 1st ACM symposium on Cloud computing*, SoCC '10, pages 217–228, New York, NY, USA, 2010. ACM.

[5] R. Bayer and E. McCreight. Organization and maintenance of large ordered indexes. *Acta informatica*, 1(3):173–189, 1972.

[6] T. C. Bell, J. G. Cleary, and I. H. Witten. *Text compression*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1990.

[7] D. Colarelli and D. Grunwald. Massive arrays of idle disks for storage archives. In *Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, Supercomputing '02, pages 1–11, Los Alamitos, CA, USA, 2002. IEEE Computer Society Press.

[8] D. Ellard, J. Ledlie, P. Malkani, and M. Seltzer. Passive nfs tracing of email and research workloads. In *Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, pages 203–216, Berkeley, CA, USA, 2003. USENIX Association.

[9] J. Griffioen and R. Appleton. Performance measurements of automatic prefetching. In *Proceedings of the ISCA International Conference on Parallel and Distributed Computing Systems*, page 16, 1995.

[10] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Franke. Drpm: dynamic speed control for power management in server class disks. In *Proceedings of the 30th annual international symposium on Computer architecture*, ISCA '03, pages 169–181, New York, NY, USA, 2003. ACM.

[11] T. M. Kroeger and D. D. E. Long. The case for efficient file access pattern modeling. In *Proceedings of the The Seventh Workshop on Hot Topics in Operating Systems*, HOTOS '99, pages 14–, Washington, DC, USA, 1999. IEEE Computer Society.

[12] D. Li and J. Wang. Eeraid: energy efficient redundant and inexpensive disk array. In *Proceedings of the 11th workshop on ACM SIGOPS European workshop*, EW 11, New York, NY, USA, 2004. ACM.

[13] B. Mao, D. Feng, H. Jiang, S. Wu, J. Chen, and L. Zeng. Graid: A green raid storage architecture with improved energy efficiency and reliability. In *Modeling, Analysis and Simulation of Computers and Telecommunication Systems, 2008. MASCOTS 2008. IEEE International Symposium on*, pages 1 –8, sept. 2008.

[14] D. A. Patterson, G. Gibson, and R. H. Katz. A case for redundant arrays of inexpensive disks (raid). In *Proceedings of the 1988 ACM SIGMOD international conference on Management of data*, SIGMOD '88, pages 109–116, New York, NY, USA, 1988. ACM.

[15] H. Satoshi, H. H. Le, and Y. Haruo. A power saving storage method that considers individual disk rotation. In *Database Systems for Advanced Applications*, 2012.

[16] H. G. S. Technologies. *Deskstar 7K2000 & Ultrastar A7K2000 Specification v1.4*.

[17] E. Thereska, A. Donnelly, and D. Narayanan. Sierra: a power-proportional, distributed storage system. Technical report, Technical Report MSR-TR-2009-153, Microsoft, 2009.

[18] D. Vengerov. A reinforcement learning framework for online data migration in hierarchical storage systems. *J. Supercomput.*, 43:1–19, January 2008.

[19] C. Weddle, M. Oldham, J. Qian, A.-I. A. Wang, P. Reiher, and G. Kuenning. Paraid: A gear-shifting power-aware raid. *Trans. Storage*, 3, October 2007.

[20] Y. Wu, K. Otagiri, Y. Watanabe, and H. Yokota. A file search method based on intertask relationships derived from access frequency and rmc operations on files. In *Database and Expert Systems Applications*, pages 364–378, 2011.

[21] P. Xia, D. Feng, H. Jiang, L. Tian, and F. Wang. Farmer: a novel approach to file access correlation mining and evaluation reference model for optimizing peta-scale file system performance. In *Proceedings of the 17th international symposium on High performance distributed computing*, HPDC '08, pages 185–196, New York, NY, USA, 2008. ACM.

[22] J. Xuehua, W. Yousuke, and Y. Haruo. Data allocation based on xml query patterns to reduce power consumption. In *InternationalConference on Cloud and Green Computing*, 2011.

[23] Q. Zhu, Z. Chen, L. Tan, Y. Zhou, K. Keeton, and J. Wilkes. Hibernator: helping disk arrays sleep through the winter. In *Proceedings of the twentieth ACM symposium on Operating systems principles*, SOSP '05, pages 177–190, New York, NY, USA, 2005. ACM.