

Power-Aware Autonomous Distributed Storage Systems for Internet Hosting Service Platforms

Jumpei Okoshi, Koji Hasebe, and Kazuhiko Kato

Department of Computer Science, University of Tsukuba, Japan
{oks@oss.s., hasebe@, kato@}cs.tsukuba.ac.jp

Abstract. We present a power-saving method for large-scale distributed storage systems of Internet hosting services, whose prime example is a video/photo sharing service. The idea behind our method is to periodically exchange stored data among disks in an autonomous way so as to skew the workload towards a small number of disks while not overloading them. The objective of this paper is to explore a power-saving method that is adaptable to both constant massive influx of data and changes in data popularity. The performance is measured both in simulation and prototype implementation using a real access pattern of 20,000 public photos on Flickr. In the experiments, we observed that our method saved 14.5–39.7% of energy, while the overall average response time was 133 ms, where 6.8–19.1% of total accesses were of disks in low-power mode.

Key words: power-saving, distributed storage systems, autonomous control.

1 Introduction

Energy efficiency has become a central issue in today's cloud computing. In particular, as a high percentage of the total computing system's energy is used by the data storage systems, various attempts at reducing power use in cloud storage systems have been proposed; e.g., [2, 3, 4, 9, 10]. These techniques are essentially based on an idea commonly considered in studies on power-saving in storage systems such as MAID [1] and PDC [5]. That is, they skew the workload towards a small number of disks and thereby enable other disks to be in low-power mode. However, when applying this idea to recent Internet hosting service platforms whose prime examples are YouTube¹ and Flickr², several important issues that have not been thoroughly investigated may arise owing to the following dynamic aspects of the stored data.

First, most previous studies explicitly or implicitly assumed that the number of stored data is fixed, but in a real situation, enormous data quantities are continuously uploaded. In addition, their popularity (i.e., frequency of access) may vary at any moment. Second, previous studies often assumed that there is a specific type of central controller for data allocation to effectively skew the

¹ <http://www.youtube.com>

² <http://www.flickr.com/>

workload, but this technique cannot be directly applied to large-scale storage systems owing to a scalability issue. The objective of this paper is to address these issues and explore a power-saving method that is adaptable to a typical environment of Internet hosting services, i.e., constant massive influx of data and changes in data popularity.

Our method is based on the idea behind MAID and PDC systems, which is the migration of frequently accessed data to a subset of the disks. However, to enhance scalability, our method periodically exchanges data among disks in an autonomous way such that frequently accessed disks tend to gather frequently accessed data from neighboring disks up to their capacity, and the opposite occurs for rarely accessed disks so as to extend their time in low-power mode. In this paper, we also consider several types of restrictions on the exchange of data and evaluate their effect on performance in terms of power consumption, response time, and data migration cost.

To evaluate the effectiveness of our method in a more realistic situation, we measured the performance both in simulation and prototype implementation using a real access pattern of 20,000 public photos uploaded to Flickr, which are observable outside the website. In the experiments, we observed that our method resulted in energy savings of 14.5–39.7%, while the overall average response time was 133 ms. According to our experimental results, our method effectively skewed the workload even if the data migration was conducted autonomously. On the other hand, accesses of data stored on disks in low-power mode totaled 6.8–19.1% of all accesses, and these accesses required extra time for the spinning-up of disks. As a major factor of this problem, we observed that the number of accesses rapidly decreased after one week from the upload for most of the files, and such infrequent accesses were evenly distributed. Thus, in our method, it was difficult to gather such unpopular data on some specific disks completely. This results in a trade-off between the idleness threshold (i.e., the period for never-accessed disks to spin down) and response time.

This paper is organized as follows. Section 2 presents related work. Section 3 describes the design of the proposed storage system. Section 4 gives the results of preliminary investigations of access patterns of public uploaded photos on Flickr, which are used in both simulations and in experiments with a prototype. Sections 5 and 6 present the simulation results and the evaluation of the prototype implementation. Finally, Section 7 concludes the paper and presents future work.

2 Related work

There have been a number of studies on reducing storage power consumption. A common feature of many of the techniques proposed in the literature is that they skew the workload, and they can be classified into the following categories according to variations in their approach.

The first category, which includes MAID [1] and PDC [5], focuses on the popularity and concentrates popular data on specific disks. The second category,

as typified by Pergamum [8], uses NVRAM to extend the low-power mode period by caching data to a write store. The final category considers redundancy (i.e., data replication). In DIV [6], original and redundant data are separated onto different disks, thereby allowing read/write requests to be concentrated on the disks with the original data. In Hibernator [13] and PAROID [11], data are collected or spread to adapt to changes in operational loads.

Although the above studies restricted their scope to storage systems consisting of a relatively small number of disks (typically, up to several dozen), recent works such as those of Harnik et al. [2], Kaushik et al. [4], Verma et al. [9], Vrbsky et al. [10], and our previous work [3] address power-saving in large-scale distributed storage on the basis of the existing skewing technique. Our research can be thought of as a direct successor to these studies based on the approach taken in the first category, but the main motivation is to explore power-saving in an environment where a huge number of data are continuously uploaded and the data access frequency varies at any moment, whose prime example is Internet hosting services.

3 System Design

Our proposed storage system is composed of several thousand (possibly heterogeneous) disks, each of which is classified into one of three groups: Group A, Group B, and the Empty disk pool. (See also Fig. 1 for the graphical presentation.) Each disk autonomously travels among these groups (depicted by the thick arrows in the figure) depending on its capacity in the following way.

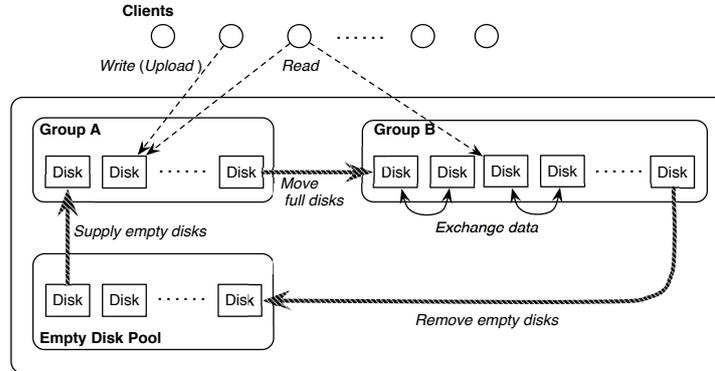


Fig. 1. Architecture of the System

The system is assumed to have many empty disks (with unique IDs), which are stored in the Empty disk pool for future increases in stored data. Initially, some of the empty disks in the pool are moved to Group A, and the data uploaded

by the clients are always written to these disks. If a disk in Group A becomes full, then it moves to Group B and a new empty disk is supplied to Group A from the disk pool.

In Group B, to skew the workload, each disk periodically (e.g., once every hour) exchanges part of its stored data one-for-one following four steps.

Step 1: The disk (say, D_1) with the smallest ID randomly chooses a disk (say, D_2).

Step 2: D_1 and D_2 exchange information of their current workloads.

Step 3: If D_1 is more popular than D_2 , the least frequently accessed file (say, f_L) on D_1 is exchanged for the most frequently accessed file (say, f_H) on D_2 , and this process is repeated while the frequency of access of f_L is less than that of f_H .

Step 4: D_1 issues an instruction to the next disk.

When implementing this process, the access frequency of popular disks gradually increases, until the capacity of the disks is exceeded, by gathering popular data from the neighboring disks, and the opposite process is carried out for unpopular disks so as to extend their time spent in low-power mode. During this exchange process, if a disk eventually becomes empty, it is removed from Group B and queued in the Empty disk pool for the supply of disks to Group A.

Here, we may consider restrictions on the exchange to reduce migration cost, such as limits on the number of exchangeable files during any one exchange, the number of migrations for each file, and the scope of exchangeable disks. In this study, we evaluated the effects of restrictions on the energy performance of disks, which are described in later sections.

The underlying lookup service used by clients to access data is managed by a distributed hash table, such as Chord [7]. However, for adaption to our system in which data are frequently moved among disks, we use pointers in a list of key-value pairs to indicate the current disk on which data are stored. That is, if a file f_1 stored on disk D_1 migrates to D_2 , the key of f_1 is associated with the destination of migration (i.e., D_2) in D_1 's list of key-value pairs. In addition, to avoid multiple hops from pointer to pointer, each file is assumed to have information of the ID of the disk originally storing the file as metadata, and if f_1 originally stored on D_1 and moved to D_2 is moved further to D_3 , the pointer of D_1 is rewritten so as to directly indicate the current position.

4 Data Access Tracing in Flickr

To evaluate the effectiveness of our method in a realistic situation, as a preliminary study, we traced access patterns of photo data uploaded to Flickr, which is one of the largest photo sharing services in the world.

In this preliminary study, we randomly selected 20,000 photos and traced the cumulative number of accesses for each file every hour over two weeks with APIs provided by the website. Owing to limitations of accessible observable data, all

the selected photos are public, although the website has supposedly around four times as many private photos as public photos (according to a Flickr report and our observations).

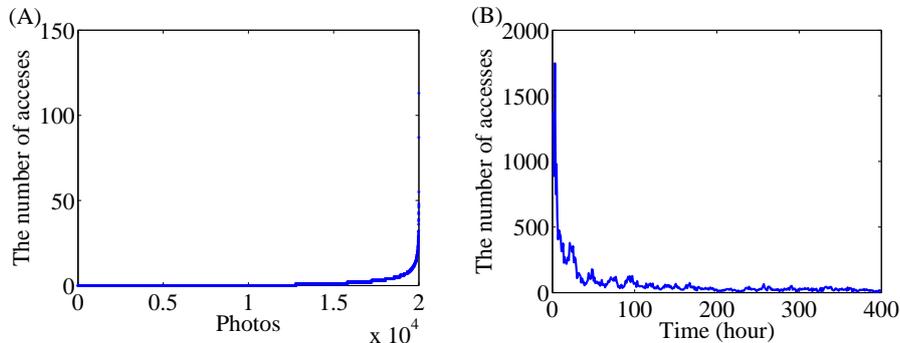


Fig. 2. (A) Distribution of popularity and (B) cumulative number of total accesses of 20,000 files on Flickr

Figure 2-(A) shows the distribution of popularity for all photos at a lapse of 200 hours, while (B) shows the change in the cumulative number of total accesses of all the files over 400 hours. Figure 2-(A) shows that the file access frequencies were highly skewed, with 69.3% of files never being accessed after upload, while the number of accesses of the most popular file is 133. Figure 2-(B) shows strong negative correlation between the frequency of access and elapsed time. More precisely, the result shows that the frequency of access became highest (1748 accesses per hour) at a lapse of 3 hours and then rapidly decreased, eventually reaching 219 accesses per hour at a lapse of 17 hours. However, the frequency of access did not change after 200 hours, with most files being accessed once per hour or not at all. We here note that it is difficult to forecast which files will be accessed in a short time from the past access pattern. This makes it difficult to completely gather the accesses to some specific disks in an effort to avoid access of disks in low-power mode.

5 Simulation Results

To understand the effectiveness of our method for storage systems consisting of several thousands of disks, we first evaluated the running time of disks and the frequency of access of disks in low-power mode.

Parameters and settings. In the evaluation presented in this section, we considered the following system. Group A consisted of a single disk and Group B consisted of up to 1500 disks whose number increases depending on the upload. Each disk in Group B required 5 seconds for spin-up where it had been in low-power mode. We set the idleness threshold as 30, 60, 90, and 120 seconds.

The workload in the simulations was based on the access traces obtained by the observation of Flickr described in Section 3. In our simulation, to determine the trace of each file, we chose at random from the set of 20,000 real traces when uploaded. However, as our traces were observed for 400 hours, we expanded the real traces to 2400 hours by repeating the access pattern from 200 hours to 400 hours for the period after 400 hours.

In the simulations, we first measured the running time to evaluate the impact of both the configurations of idleness threshold and restrictions of data exchange on the power consumption then measured the number of accesses of disks in low-power mode in some configurations.

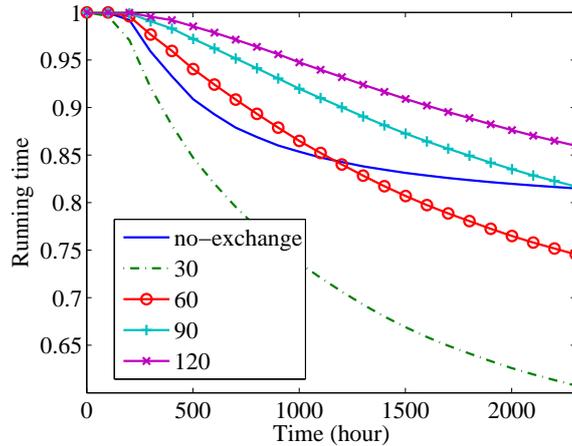


Fig. 3. Running time in some configurations of idleness threshold

Impact of exchange and idleness threshold on power consumption.

Figure 3 shows the change in running time for different configurations of idleness threshold. The figure shows the relative time when the running time in the case that all the disks are always active is equal to one. In this figure, the “no-exchange” configuration means that the disks spin down after an idle time (30 seconds) without data exchange and, in other configurations, disks spin down after an idle time (30, 60, 90, and 120 seconds) with data exchange. We observed that power consumption after 2400 hours was reduced by 14.5–39.7% in each configuration. From this result, we observed that the data exchange was effective in reducing power consumption as demonstrated by comparing the “no-exchange” configuration and all the configurations with data exchange, among which the “30” configuration was the most effective.

Impact of the restrictions of data exchange on power consumption.

Figure 4 shows the change in running time for four configurations, which differed in terms of the application of two restrictions. To denote these configurations,

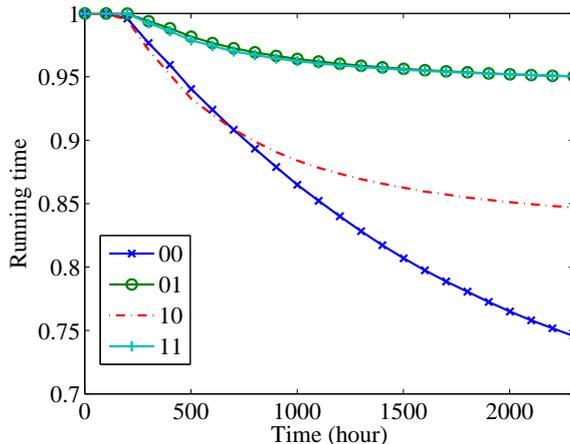


Fig. 4. Running time for some restrictions of data exchange

we use two digits b_1 and b_2 , where b_1 indicates application of a limit on the exchange range (in the case that $b_1 = 1$, one disk can change files with only 20 specific disks), while b_2 indicates application of a limit on data migration (in the case that $b_2 = 1$, the migration capacity is limited to 10% of each disk space). The figure shows the relative time when the running time in the case that all the disks are active is equal to one. Here, the idleness threshold is fixed as 60 seconds in each configuration. In this simulation, we observed that our method still reduced power even if one of the restrictions was introduced, which may reduce the migration cost. In addition, under these parameter settings, the limit on the exchange range (i.e., b_1) was stronger than the limit on data migration (i.e., b_2), which would change depending on a given trace.

Accesses of disks in low-power mode. Figure 5 shows the change in ratio of the access of disks in low-power mode among all accesses in the same configuration as in the case of Figure 3. We observed that 6.8–28.7% of accesses were of disks in low-power mode. This result means that 6.8–28.7% of accesses need extra time for spinning-up. However, it should be emphasized that the data exchange realized a 9.5% reduction in accesses of disks in low-power mode, as seen by comparing the “no-exchange” configuration and “30” configuration, which had the same idle time but differed in terms of there being data exchange.

6 Experiments on an Implementation

We conducted an experiment on the current prototype implementation of our proposed system to evaluate the applicability of our method to a real system. We measured the response time and the cost of data migration in an environment where the system workload was the same as that in the simulation.

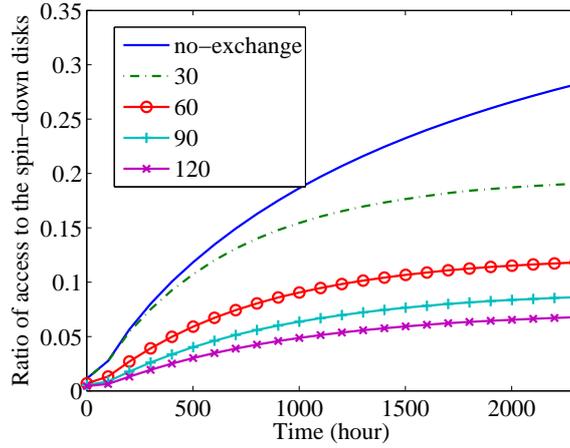


Fig. 5. Ratio of access of disks in low-power mode

Our prototype consisted of 50 PC servers (where one server was used for the client and the other servers were used for Group B), each of which was equipped with a Dual Xeon 3.60 GHz CPU, 2 GB memory, and a single 36 GB SCSI disk. For our prototype, owing to the limitation of our experimental environment (i.e., the bandwidth of different servers), we evaluated the response time of data access by measuring the time from sending a request until the data were loaded into the memory of the server. In addition, no underlying lookup service to access data was implemented in our prototype. Thus, in the experiments, the data were accessed by their storing server.

Parameters and settings. In our experimental environment, although the real capacity of a disk was 36 GB, we assumed that the capacity was 500 GB, which was emulated by only accessing one file. In addition, because it was difficult to spin up or spin down disks in the current system configuration, we realized these actions by letting the server wait before accessing the disk. The two parameters were spin-up time of 5 seconds and idle time before spin-down of 60 seconds. (According to simulation analysis, this configuration was the best in terms of response time and power consumption.)

Response time. Figure 6 shows the change in response time. The figure shows that the overall average response time is 133 ms. However, we observed that some responses were delayed and required more than 5 seconds after 15 hours, among which the worst took 18,307 ms. This result means that some accesses were of the spin-down disk and had to wait for the disk to spin up or for the completion of another file access.

Migration cost. Table 1 shows the migration cost evaluated by measuring the number of files that migrated per data exchange and the elapsed time. Note that this result was for 48 hours and the migration cost would decrease gradually after

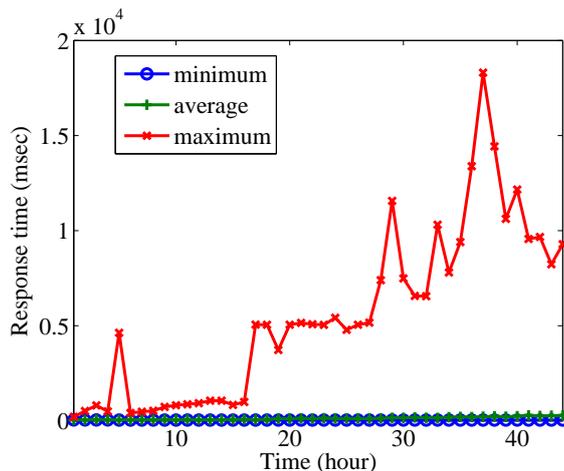


Fig. 6. Response time

Table 1. Migration cost

| | minimum | median | average | maximum |
|-------------------------------|---------|--------|---------|---------|
| The number of exchanged files | 2 | 1691 | 13583 | 163079 |
| Time (s) | 0.2 | 170 | 1426 | 17123 |

48 hours. In addition, owing to the limitation of our experimental environment, times in Table 1 are predicted values. This result shows that the required time for most of the migration was short relative to the frequency of exchange (i.e., once every hour) and would have no adverse impact on the performance.

7 Conclusions and Future Work

We presented a power-saving method for large-scale distributed storage systems, especially those on Internet hosting service platforms. The idea to skew the workload is similar to that of MAID and PDC, but to adapt to the constant massive influx of data and changes in data popularity, our approach periodically exchanges data among disks autonomously, instead of introducing a central controller to manage the relocations. We also evaluated the performance both in simulations and prototype implementation for a real access pattern of 20,000 public photos on Flickr. We observed that our method saved 14.5–39.7% energy, even if we introduced several restrictions on data migration. The ratio of accesses of disks in low-power mode was 6.8–19.1%, and the overall average response time was 133 ms.

The results of this research suggest that our autonomous approach can effectively skew the workload, but some accesses require extra time to wait for the spinning-up of disks in low-power mode. This results in a trade-off between the

idleness threshold of disks, which may increase the power consumption, and the performance of the response time. To more effectively skew the workload, we should investigate a radical solution that possibly considers effective allocations of replicas. In addition, we should thoroughly estimate the cost of underlying lookup services. These topics shall be investigated in future work.

References

1. D. Colarelli and D. Grunwald. Massive arrays of idle disks for storage archives. In *ACM/IEEE Conference on Supercomputing*, pp.1-11, 2002.
2. D. Harnik, D. Naor, and I. Segall. Low power mode in cloud storage systems. *Parallel and Distributed Processing Symposium, International*, pp.1-8, 2009.
3. K. Hasebe, T. Niwa, A. Sugiki, and K. Kato. Power-Saving in Large-Scale Storage Systems with Data Migration. *IEEE International Conference on Cloud Computing Technology and Science (CloudCom'10)*, pp.266-273, 2010.
4. R. T. Kaushik and M. Bhandarkar. GreenHDFS: towards an energy-conserving, storage-efficient, hybrid Hadoop compute cluster. In *2010 international conference on Power aware computing and systems (HotPower'10)*, pp.1-9, 2010.
5. E. Pinheiro and R. Bianchini. Energy conservation techniques for disk array-based servers. In *International Conference on Supercomputing*, pp.68-78, 2004.
6. E. Pinheiro, R. Bianchini, and C. Dubnicki. Exploiting redundancy to conserve energy in storage systems. In *ACM SIGMETRICS Conference on Measurement and modeling of computer systems*, pp.15-26, 2006.
7. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *ACM SIGCOMM*, pp.149-160, 2001.
8. M. Storer, K. Greenan, E. Miller, and K. Voruganti. Pergamum: Replacing tape with energy efficient reliable, disk-based archival storage. In *USENIX Conference on File and Storage Technologies (FAST'08)*, pp.1-16, 2008.
9. A. Verma, R. Koller, L. Useche, and R. Rangaswami. SRCMap: energy proportional storage using dynamic consolidation. In *8th USENIX Conference on File and Storage Technologies (FAST'10)*, pp.154-168, 2010.
10. S. V. Vrbisky, M. Lei, K. Smith, and J. Byrd. Data Replication and Power Consumption in Data Grids. In *2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom'10)*, pp.288-295, 2010.
11. C. Weddle, M. Oldham, J. Qian, A. Wang, P. Reiher, and G. Kuenning. PARAD: A gear-shifting power-aware RAID. In *USENIX Conference on File and Storage Technologies (FAST'07)*, pp.245-260, 2007.
12. X. Yao and J. Wang. RIMAC: a novel redundancy-based hierarchical cache architecture for energy efficient, high performance storage systems. In *ACM SIGOPS/EuroSys European Conference on Computer Systems*, pp.249-262, 2006.
13. Q. Zhu, Z. Chen, L. Tan, Y. Zhou, K. Keeton, and J. Wilkes. Hibernator: helping disk arrays sleep through the winter. In *ACM symposium on Operating systems principles*, pp.177-190, 2005.