Multi-Tier Power-Saving Method in Cloud Storage Systems for Content Sharing Services

Horleang Choeng, Koji Hasebe, Hirotake Abe, and Kazuhiko Kato

Department of Computer Science, University of Tsukuba
1-1-1, Tennodai, Tsukuba 305-8573, Japan
horleang@osss.cs.tsukuba.ac.jp, {hasebe,habe,kato}@cs.tsukuba.ac.jp

Abstract. Fast-growing cloud computing has a mass impact on power consumption in datacenters. In our previous study, we presented a power-saving method for cloud storage systems, where the stored data were periodically rearranged in a disk array in the order of access frequency. The disk containing unpopular files can often be switched to power-saving mode, enabling power conservation. However, if such unpopular files became popular at some point, the disk containing those files spin up that leads to increase power consumption. To remedy this drawback, in this paper, we present a multi-tier power-saving method for cloud storage systems. The idea behind our method is to divide the disk array into multiple tiers. The first tier containing popular files is always active for fast response, while lower tiers pack unpopular files for power conservation. To maintain such a hierarchical structure, files are periodically migrated to the neighboring tiers according to the access frequency. To evaluate the effectiveness of our proposed method, we measured the performance in simulations and a prototype implementation using real access traces of approximately 60,000 time-series images with a duration of 3,000 hours. In the experiments, we observed that our method consumed approximately 22% less energy than the system without any file migration among disks. At the same time, our method maintained a preferred response time with an overall average of 86 ms based on our prototype implementation.

Keywords: power-saving · storage system · cloud computing · content sharing service.

1 Introduction

Cloud computing has grown rapidly over the last few decades, resulting in a dramatic increase in power consumption of datacenters. According to a report [15], in 2014, datacenters in the US consumed approximately 1.8% of the total electricity consumption. As a high percentage of the total computing energy is consumed by storage systems, a number of attempts to reduce storage power consumption. Many of these stuides were essentially based on the commonly employed technique of skewing the workload toward a small subset of disks, thereby enabling the other disks to remain in standby (i.e., low-power) mode.

Massive Arrays of Idle Disks (MAID) [5] is one of the first efforts at this approach. MAID distributed the workload to a subset of disks used as a cache to preserve data. Popular Data Concentration (PDC) [12] periodically rearranged the data based on their new access frequencies in a disk array.

In our previous study [11], we proposed a method based on the idea of PDC. Our method efficiently gathered unpopular files in to a part of the disk array in an environment where large amounts of files were continuously uploaded. However, most of the uploaded files were rarely accessed as time elapsed, but accesses reoccurred occasionally and seemed to happen randomly. The algorithm proposed in our previous study failed to move the files that became popular again from the unpopular disks to the active disks.

To remedy this drawback, in this paper, we propose a multi-tier power-saving method for cloud storage systems for cloud content sharing services. Our method divides the disk array into multiple tiers. The top tier, consisting of always active working disks, stores popular files for fast response, while the lower tiers, consisting of archiving disks, pack unpopular files for power conservation. To maintain this hierarchical structure, files are periodically migrated to the neighboring tiers according to the access frequency.

To evaluate the effectiveness of our proposed method, we measured the performance in both simulations and a prototype implementation using real access traces of approximately 60,000 time-series public photographs on 500px [1] for 3,000 hours. In the experiments, we measured the performance of power-saving and response time in a system consisting of two tiers. The results showed that our method consumed approximately 22% less energy than a system without any file migration among disks. Moreover, we observed that our method maintained a preferred response time with an overall average of 86ms based on our prototype implementation.

The rest of this paper is organized as follows. Section 2 presents related work. Section 3 gives the details of the system design. Section 4 introduces the power consumption model for our evaluation. Sections 5 and 6 present the results of our simulations and evaluation using an implementation. Finally, Section 7 concludes the paper and presents future work.

2 Related Work

There have been a number of studies on saving power consumption in storage systems (cf. also [4] for a comprehensive survey of this research area). These techniques can be classified into the following three categories according to variations in their approach.

The first category focuses on popularity and concentrates popular data on specific disks. Massive Array of Idle Disks (MAID) [5] provides specific disks that are used as a cache to store frequently accessed data, thereby reducing accesses to other disks. PDC [12] periodically reallocates data in the storage array according to the latest access frequencies.

The second category uses NVRAM to extend the standby mode period by caching data to a write store. A typical example is Pergamum [10], which uses NVRAM to buffer write accesses and store data signatures and thus reduce the number of direct accesses to the disks.

The final category considers redundancy (i.e., data replication). In DIverted Accesses (DIV) [6], original and redundant data are separated onto different disks, thereby allowing I/O requests to be concentrated onto the disks that contain the original data. Hibernator [13] applies the idea of PDC to RAID and Dynamic Rotations Per Minute (DRPM) [14]. RIMAC [18] provides two-layered caches, one for storing data and the other for parity conservation. Power-Aware RAID (PARAID) [17] is another power-saving technique for RAID. It allocates the replicas in a specific way so that data are collected or spread to adapt to changes in operational workloads.

Many of these methods mentioned above were targeted at small-scale storage systems. Recently, applications of these techniques to datacenter-scale storage systems have been actively investigated. As a typical example, GreenHDFS [7] divides Hadoop Distributed File Systems (HDFS) into hot and cold zones. Another example is a study that used data replication extended from GreenCloud [9] for energy-efficient [3]. This method can reduce power consumption, bandwidth usage, and communication delays substantially.

Our previous study [11] also aims to save power consumption in large-scale storage systems. That study specifically focuses on the issue of how to efficiently aggregate workloads in an environment where large amounts of files are continually uploaded, as typified by the storage systems for cloud content sharing services. However, as mentioned in the previous section, in that study, files that became less popular over time became popular again at some point, and it was impossible to remove it from an unpopular disk. This paper extends the method of [11] while solving this problem.

3 System Design

Our target systems consist of hundreds to thousands of disks, application servers, I/O servers, and an index manager. Fig. 1 illustrates the overall design, where the I/O servers are omitted for readability. Each disk is physically connected to an I/O server and is logically classified into one of the groups (tiers): Tier 1 to Tier n, and the empty disk pool. The disks in the top tier (Tier 1) are called working disks, while the rest are called archiving disks. We assume that our system continuously uploads the files, such as those in real content sharing service platforms. The application servers manage all I/O requests from the clients. The index manager provides a lookup service for file accesses to any working and archiving disks. An application server always writes the files uploaded by the clients to one of the top tier. After uploading, the index manager assigns each file a unique ID and records it with the disk index in which the file is stored.

The number of working disks increases one-by-one as clients sequentially upload files up to an initial number. If the number of working disks reaches the

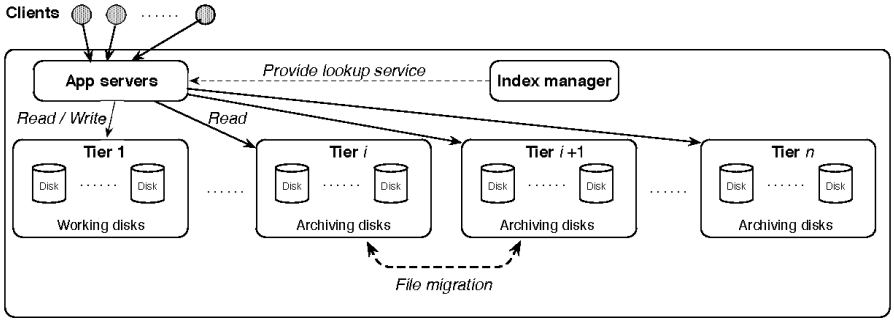

Fig. 1. System design.

maximum, any access to a full working disk leads to file migration. At this point, an archiving disk is added. Then, some of the most unpopular files are migrated to an archiving disk supplied by the empty disks. The index manager records the file name and disk index in the archiving disks to which the file is migrated. If the files are not migrated, the disk index is the corresponding disk itself. File migration is conducted between neighboring tiers as follows (for $1 \leq n \leq n-1$).

Migration from the i-th tier to $i+1$-th tier: This migration happens in two cases: (1) whenever any disk in the first tier becomes full after clients uploaded the files, and (2) whenever any disk in the i-th tier becomes full after the files are migrated from the neighboring disks.

Migration from $i+1$-th tier to i-th tier: This migration happens if the frequency of accesses of a file in the i-th tier reaches the predetermined threshold. The file is moved to a disk in the $i+1$-th tier and then sent to the client.

Here we note that the number of accesses to be migrated is critical because it affects the whole system. The reason is that if the file is migrated with every access, we can reduce many accesses to the archiving disks, but it fills the working disk faster, leading to another migration later. We observed the percentage of power-saving based on the number of accesses to be migrated, which is described in Section 5. The application server ensures that the total number of files in the working disks is no more than 30% of the total files of the whole system. If the volume exceeds its threshold, a new working disk is added to the first tier, enabling the scalability of the system. At the start of the upload, all files are stored in the working disks, and the archiving disk does not exist. When the migration occurs, a disk is pulled from the empty disks and named as the archiving disk to supply the need to store the migrated files. Another disk is added whenever a destination archiving disk is full.

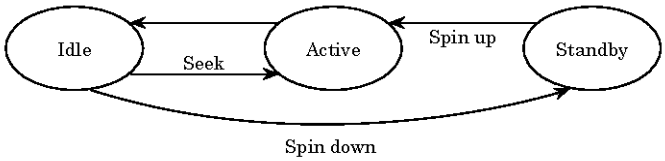

Fig. 2. State transitions for three-state disk drive.

Table 1. HDD parameter setting.

Symbol	Description	Value
D_{si}	Disk capacity	1,000GB
D_{ra}	Average data transfer rate	125MB/s
P_{id}	Power consumption in idle mode	3.36w
P_{sb}	Power consumption in standby mode	0.63w
P_{ac}	Power consumption in active mode	5.9w
P_{sk}	Power consumption to seek	5.9w
P_{up}	Power consumption to spin up	24w
T_{sk}^{rd}	Average seek time for read	8.5ms
T_{sk}^{wr}	Average seek time for write	9.5ms
T_{tr}	Rotational latency	4.16ms
T_{up}	Spin up time	10s
T_q	Time for current processing I/O requests	-
T_{tf}	Data transfer time	-
T_{th}	Idleness threshold	85.6s

4 Power Consumption Model

A modern disk drive enables dynamic power management, meaning that there are transitions among the three states called active, idle, and standby modes. Fig. 2 illustrates the state transitions. Data transfer occurs in the active mode. When a disk is awaiting I/O requests, it is transitioned to the idle mode, where the disk continues to rotate. A disk in idle mode is transitioned to the standby mode after a fixed threshold time (the idleness threshold) has elapsed since the last access. In the standby mode, the spindle is at rest, and the heads are parked, resulting in power savings.

Table 1 summarizes the model parameters and the values that we used in our evaluations. These settings are based on the specifications of the Seagate Desktop HDD ST1000DM004 [2]. In the table, the values for T_q and T_{tf} are unspecified because they depend on the finishing time of the previous I/O request and the size of the data transferred, respectively. In our model, the power consumption of a disk drive executing an I/O request is estimated as the sum of the power required for transferring data and the power required for state transition. For example, the total power consumed by a disk in standby mode spinning up and performing I/O requests is $T_{up} \cdot P_{up} + T_{acs} \cdot P_{ac}$ [mJ], where T_{acs} denotes the

access time explained below. The response time of access to a disk depends on the current mode of the disk. When we set T_{acs} as the access time (I/O request processing time), T_{sk} as the seek time, T_{rt} as the rotational latency, and T_{tf} as the transfer time, we have an equation as

$$T_{acs} = T_{sk} + T_{rt} + T_{tf}.$$

The response time of accesses in each mode is given by

$$T_{rp} = \begin{cases} T_{acs} & \text{(if it is in idle mode)}, \\ T_{acs} + T_{up} & \text{(if it is in standby mode)}, \\ T_{acs} + T_q & \text{(if it is in active mode)}. \end{cases}$$

It is critical to set the idleness threshold to a suitable value if we wish to reduce the power consumption using dynamic power management. A too-small threshold could result in a frequent spin-up, requiring considerable power (denoted by P_{up}). Conversely, a too-long threshold could prohibit state transitions from idle to standby mode, which could further reduce power consumption. To set a suitable idleness threshold, we use the well-known break-even time technique, i.e., we determine the amount of time a disk must be in standby mode to conserve the same energy consumed by transitioning the disk down and back to the active mode. More precisely, the break-even time (denoted by T_{be}) can be calculated as follows. First, T_{be} can be decomposed into

$$T_{be} = T_{up} + T_{sb}^{min}.$$

Here, T_{sb}^{min} is the minimum length of the standby mode after completing the previous I/O request satisfying the following equation:

$$P_{sb} \cdot T_{sb}^{min} + P_{up} \cdot T_{up} = P_{id}(T_{sb}^{min} + T_{up}).$$

This equation means that the total power required to be in standby mode and spin-up (described as the left-hand side) is equal to the total power required to remain in the idle mode (described as the right-hand side). From these results, we obtain

$$T_{be} = \frac{P_{up} \cdot T_{up} - P_{sb} \cdot T_{up}}{P_{id} - P_{sb}}.$$

In our model, $T_{be} = 85.6$ sec, which we used as the idleness threshold in our evaluations.

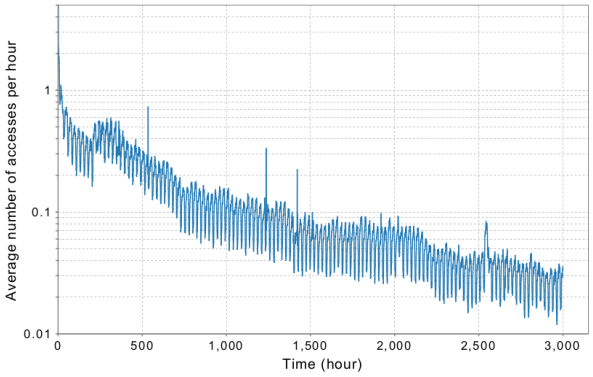

Fig. 3. Hourly average number of access of images.

5 Simulation

5.1 Preparation

Photographs for simulation. For simulations, we used 63,204 time-series images from 500px [1], which is an online photo sharing service. All photographs include the cumulative number of accesses, "likes" (the number showing the popularity of the photographs voted by users), comments, and tag information of every hour for 3,000 hours. We analyzed the bias of the popularity of each photograph to evaluate the relationship with the time since they were uploaded.

Analysis of photographs for simulation. Fig. 3 shows the hourly average number of accesses of images over 3,000 hours. Here, the horizontal and vertical axes represent the elapsed time and the average number of access per hour, respectively. Consequently, we can assume that many accesses tend to concentrate on some specific images. Fig. 4 shows the distribution of total accesses of each image after 3,000 hours elapsed. The maximum access is 182,669 times, while the minimum access is only two times. The rate of total accesses in the top 1% of total images holds almost half of the total accesses, while the rate of accesses in the top 30% of the total images holds almost 90% of the total accesses. Consequently, we can assume that most of the accesses to the images belong to the top 30% of the total images. Therefore, we consider that by storing 30% of the images in the working disks, and the rest (around 70%) in the archiving disks, we can reduce the power consumption of the whole system. Most (90%) of the accesses will likely be concentrated on the working disks, and the workload in archiving disks is at rest, resulting in power consumption reduction.

Parameters and setting. Based on our system design in Section 3 and the power consumption model in Section 4, we developed a simulator. The number of

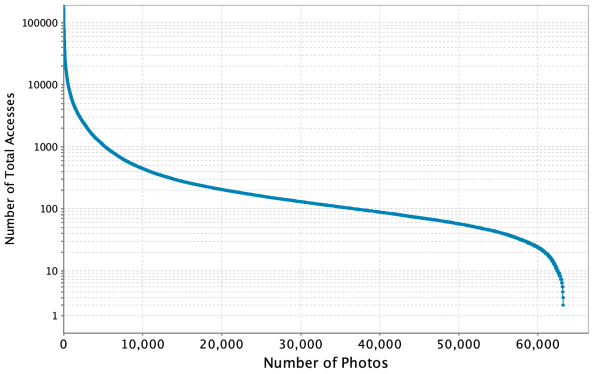

Fig. 4. Distribution of total accesses after 3,000 hours.

Table 2. Workload setting.

Type	Value
File size	5MB
Write	26 images per minute
Read data	500px's image access pattern
Simulation duration	3,000 hours

accesses mimics the access traces of the images mentioned in Section 5.1. Table 2 describes the workload setting. In our simulations, the working disks initially consist of eight disks, while the archiving disks consist of zero disks supplied from empty disks (21 disks were used at the end of the simulations). The number of archiving disks increased one-by-one, depending on the file migration from the working disks.

5.2 Simulation results

We observed the power consumption of our system and compared it to a simple system in which the same set of files are uploaded to an array of disks (increasing one-by-one) without any file migration, named no-migrating system. The number and type of disks in this no-migrating system are the same as those in the proposed system. Because our proposed method focuses on file migration between disks in different tiers (working disk and archiving disk), it is critical to define the two main factors for migration. The first factor is unpopular files. In our experiments, we assume that unpopular files are those with less access frequency or with the oldest access time. The second factor is to determine the number of accesses of files needed to migrate back to any working disk because the performance of the system would worsen if many migrations occur. We experimented on these two main factors to determine the best migration method for our proposed system. Table 3 illustrates the result of our experiment on the percentage of power-saving, where WD and AD represent working and archiving

Table 3. Experiment on percentage of power-saving.

AD→WD / WD→AD	1 access	3 accesses	5 accesses	10 accesses	20 accesses
Number of accesses	17.8%	19.4%	21.3%	22.1%	22.1%
Old accessed time	18.6%	20.0%	22.2%	23%	21.9%

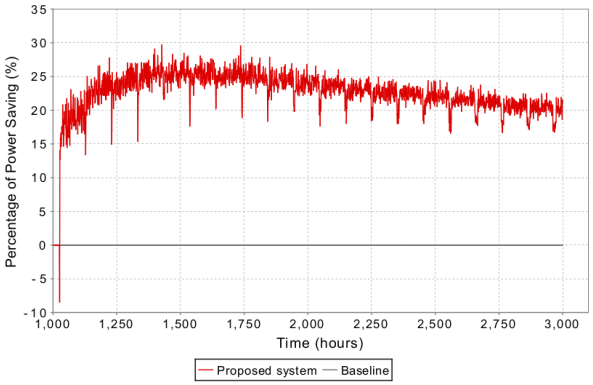

Fig. 5. The change of percentage of power-saving.

disks, respectively. The results show that the best result for power consumption is when we migrate the oldest accessed files from working disks to archiving disk and when the number of access to old files is more than 10 times.

Fig. 5 shows the change in the percentage of power consumption of the best results we described above. Here, the horizontal and vertical axes represent the elapsed time and the percentage of power-saving. We started measuring the power consumption between the proposed method and a no-migrating system after the migration process started. Our proposed system conducted file migration, while the files stored in the no-migrating system are static. According to our results, our method performed better with a maximum of 29% less power consumption. Also, over the 3,000 hours, this method consumed approximately 22% less energy than the no-migrating system.

Since simulation is not enough to evaluate our proposed system, we conducted a prototype implementation, described in Section 6.

6 Experiments on Implementation

6.1 Parameters and setting

We observed the response time in an environment where the system workload was the same as that in the simulation described in Section 5. Our prototype consisted of a client server and an application server connected to an array of 12 disks, each of which was equipped with a single 3 TB hard disk drive. We

assumed that the capacity of each disk is 1 TB to match our simulation. Because of the limitation of our experimental environment, we mimicked the simulation of our target system in the following way. All the disks were created as RAID 0 and were connected to the app server. The app server consisted of an index manager mapping the location of each disk and file. We evaluated the response time by measuring the time from sending requests until the data were delivered to the clients. To avoid the effect of the cache, we disabled the cache in each disk before experimenting and erased the memory cache every 30 sec during the experiment. Since the controller of our RAID 0 disks can only set the spin-down and spin-up time (for a power-saving mode) to a minimum of 30 min, we mimicked the spin-down and spin-up time as the same as those in the simulation. In other words, we mimicked the spin-up time by letting the server wait before accessing the disk. Also, as described in Table 1, we set the idleness threshold and the spin-up time to 85.6 sec and 10,000 sec, respectively. Our server network bandwidth is a 1,000 BASE-T network. We selected three disks as working disks and the rest of the nine disks as archiving disks. The files we placed on these disks were the same as those after 3,000 hours have passed in the simulation. We measured the response time of every access for one hour from this point. At the beginning of this experiment, the number of read and write requests sent to the application servers was approximately 240,000 requests.

6.2 Implementation result

Fig. 6 and Fig. 7 show the change in the maximum and average, respectively. The horizontal axis represents the duration of implementation, while the vertical axis shows the response time. We can see in Fig. 6 that some files had a response time of more than 10,000ms, indicating that the files were accessed when the disks were in power-saving mode. Furthermore, we observed that the files with a response time over 10,000ms were all stored in the first two archiving disks, meaning that our proposed method can enable the disk to switch to power-saving mode. We experimented with the response time of the no-migrating system to compare it with that in the proposed system. We observed that the overall average response time of the no-migrating system is 52ms. Meanwhile, the overall average of the response time of the proposed system is only 86ms (Fig. 7). Although the proposed system performed slower than the no-migrating system, we still prefer the proposed system because we can reduce power consumption by approximately 22%.

7 Conclusion and Future Work

In this study, we proposed a multi-tier power-saving method in storage systems, targeting cloud content sharing services. The main objective of this study was to reduce the number of accesses in disks that are in power-saving mode. We presented a method that performed data migration between two subsets of an array. One subset, working disks, was always in the active mode, and the other

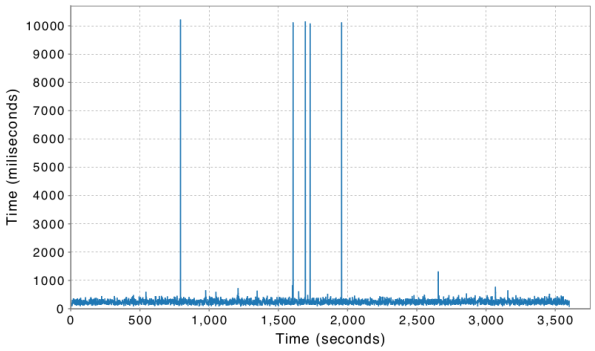

Fig. 6. Maximum response time.

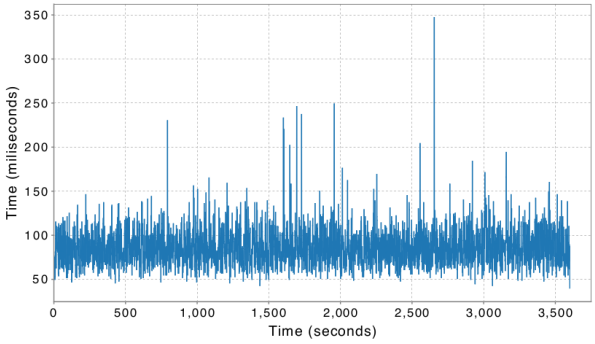

Fig. 7. Average response time.

subset, archiving disks, switched to power-saving mode whenever there was no access after a threshold of time. By migrating the files between archiving disks and working disks, we could reduce the number of accesses, enabling power conservation. To evaluate our proposed system, we conducted a simulation by mimicking the actual social service using approximately 60,000 time-series images. The simulation showed that our system consumed a maximum of 29% and an average of 22% less energy compared to the system with no file migration among the disks. Furthermore, our prototype showed that although there is file migration among the disks, our system still could offer a preferred response time of 86ms in the overall average.

In future work, it is critical to investigate migration factors because we only focused on the oldest accessed time and number of accesses. Recently, some studies apply machine learning to predict the number of views of a file before it is uploaded [16], [8]. These studies could precisely predict the popularity of files by using the file itself, its content, and its social context. It would be interesting if we could apply this technique to our system by predicting the popularity of files to find the unpopular files to migrate to the archiving disks.

References

1. https://web.500px.com,
2. https://www.seagate.com/www-content/product-content/barracuda-fam/desktop-hdd/barracuda-7200-14/en-us/docs/100686584y.pdf,
3. Boru, D., Kliazovich, D., Granelli, F., Bouvry, P., Zomaya, A.Y.: Energy-efficient data replication in cloud computing datacenters. Cluster computing **18**(1), 385–402 (2015)
4. Bostoen, T., Mullender, S., Berbers, Y.: Power-reduction techniques for data-center storage systems. ACM Computing Surveys (CSUR) **45**(3), 33 (2013)
5. Colarelli, D., Grunwald, D.: Massive arrays of idle disks for storage archives. In: SC'02: Proceedings of the 2002 ACM/IEEE Conference on Supercomputing. pp. 47–47. IEEE (2002)
6. E. Pinheiro, R.B., Dubnicki, C.: Exploiting redundancy to conserve energy in storage systems. In: Proc. ACM SIGMETRICS Conference on Measurement and modeling of computer systems. pp. 15–26 (2006)
7. Kaushik, R.T., Bhandarkar, M.: Greenhdfs: towards an energy-conserving, storage-efficient, hybrid hadoop compute cluster. In: Proceedings of the USENIX annual technical conference. vol. 109, p. 34 (2010)
8. Khosla, A., Das Sarma, A., Hamid, R.: What makes an image popular? In: Proceedings of the 23rd international conference on World wide web. pp. 867–876. ACM (2014)
9. Kliazovich, D., Bouvry, P., Khan, S.U.: Greencloud: a packet-level simulator of energy-aware cloud computing data centers. The Journal of Supercomputing **62**(3), 1263–1283 (2012)
10. M. Storer, K. Greenan, E.M., Voruganti, K.: Pergamum: Replacing tape with energy efficient reliable, disk-based archival storage. In: Proc. USENIX Conference on File and Storage Technologies
11. Okoshi, J., Hasebe, K., Kato, K.: Power-saving in storage systems for internet hosting services with data access prediction. In: 2013 International Green Computing Conference Proceedings. pp. 1–10. IEEE (2013)
12. Pinheiro, E., Bianchini, R.: Energy conservation techniques for disk array-based servers. In: Proceedings of the 18th annual international conference on Supercomputing. pp. 68–78. ACM (2004)
13. Q. Zhu, Z. Chen, L.T.Y.Z.K.K., Wilkes, J.: Hibernator: helping disk arrays sleep through the winter. In: Proc. ACM symposium on Operating systems principles. pp. 177–190 (2005)
14. S. Gurumurthi, A. Sivasubramaniam, M.K., Franke, H.: Drpm: dynamic speed control for power management in server class disks. In: ACM SIGARCH Computer Architecture News
15. Shehabi, A., Smith, S., Sartor, D., Brown, R., Herrlin, M., Koomey, J., Masanet, E., Horner, N., Azevedo, I., Lintner, W.: United states data center energy usage report (2016)
16. Trzciński, T., Rokita, P.: Predicting popularity of online videos using support vector regression. IEEE Transactions on Multimedia **19**(11), 2561–2570 (2017)
17. Weddle, C., Oldham, M., Qian, J., Wang, A.I.A., Reiher, P., Kuenning, G.: Paraid: A gear-shifting power-aware raid. ACM Transactions on Storage (TOS) **3**(3), 13 (2007)
18. Yao, X., Wang, J.: Rimac: a novel redundancy-based hierarchical cache architecture for energy efficient, high performance storage systems. In: Proc. ACM SIGOPS/EuroSys European Conference on Computer Systems. pp. 249–262 (2006)