Electronic Preprint for Journal of Information Processing Vol.24 No.1

Regular Paper

A Game Theoretic Approach to Power Reduction in Distributed Storage Systems

Koji Hasebe[1,a) Takumi Sawada[2,b) Kazuhiko Kato[1,c)

Received: April 21, 2015, Accepted: August 10, 2015

Abstract: We present a game theoretic approach for power reduction in large-scale distributed storage systems. The key idea is to use a distributed hash table and migrate its virtual nodes dynamically so as to skew the workload towards a subset of physical disks while not overloading them. To realize this idea in an autonomous way, virtual nodes are regarded as selfish agents playing a game in which each node receives a payoff according to the workload of the disk on which it currently resides. We model this setting as a potential game, a kind of strategic game in which the incentive of all players to change their strategy can be represented by a single global function. Thus, any increase in the payoff of a virtual node yields a better state in terms of energy conservation. This game model consists of a pair of global and private payoff functions, derived by the Wonderful Life Utility scheme. The former function evaluates how good the current state of the system is, while the latter determines the current payoff of each node. The performance of our method is measured both by simulations and a prototype implementation. From the experiments, we observed that our method consumed 11.1%–16.4% less energy than the static configuration. In addition, although a small number of responses were heavily delayed because of overloading of some disks at peak time, our method maintained preferred overall average response time in the range 50–190 ms.

Keywords: distributed storage system, power-saving, autonomous control, game theory

1. Introduction

Energy conservation has become a central issue in today's computing systems. In particular, because a high percentage of the total energy utilized by a computing system is consumed by its data storage system, there have been a number of attempts at reducing the power consumption of storage systems. A technique used in most of these studies is to skew the workload towards a subset of disks, thereby enabling the others to be in low-power mode. This technique can be traced back to prominent early studies such as MAID [2], and PDC [19]. However, many early studies restricted their scope to storage systems with a specific kind of central controller to manage the data access or storage systems consisting of a relatively small number of disks (typically, up to several dozens). Thus, when considering increasingly huge computing systems as typified by the cloud, scalability is of major importance.

In this paper, we propose a power-saving method for large-scale distributed storage systems. As our prime interest is to achieve a higher level of scalability, the main objective of this paper is to investigate an autonomic control technique to effectively skew data accesses in a storage array without any kind of central controller. In particular, our study targets storage systems that consist of hundreds to thousands of physical disks. To achieve this goal, our method uses a distributed hash table (DHT) [29] to provide a lookup service for data accesses. More specifically, we use the concept of virtual node of DHT, which was originally used in Refs. [3], [4], [21] for load balancing. In our method, the data are stored in virtual nodes of the underlying DHT, with each node migrating over physical disks according to the current workload of the resident disk. Here, the "principle for migration" can be stated as follows: for any physical disk, the higher the workload, the better it is for the system, but overload must be avoided. This idea was originally introduced in our previous work [7]. However, in that work, the migration destination of each virtual node was strictly predetermined in a specific way, and thus could not migrate freely to another disk that may yield a better state. Moreover, owing to this restriction, it was also difficult to change the system configuration by adding/removing disks (when extending the system or when node failure occurred), as well as to deal with changes in the distribution of popularity (i.e., data access frequency).

To overcome this flexibility issue, in this paper we propose a game theoretic approach based on the following idea. First, the virtual nodes are considered selfish agents that are able to move to any of the neighboring physical disks. Next, we introduce a game in which each of the virtual nodes receives a payoff according to the workload of the physical disk on which it currently resides. Here, the payoff is defined so as to motivate virtual nodes to move in conformity with the "principle for migration" stated above. That is, the higher the workload of the disk on which a virtual node resides, the higher the payoff the node receives. Conversely, if the disk on which the node is currently residing

1 Department of Computer Science, University of Tsukuba, Tsukuba, Ibaraki 305–8573, Japan
2 DWANGO Co., Ltd., Chuo, Tokyo 104–0061, Japan
a) hasebe@cs.tsukuba.ac.jp
b) sawada@osss.cs.tsukuba.ac.jp
c) kato@cs.tsukuba.ac.jp

is overloaded, it is penalized. In this setting, virtual nodes indirectly cooperate to realize the best system state even though the node is only pursuing its own profit. We model this setting as a potential game [16], a class of strategic games in which the incentive of all players to change their strategy can be represented by a single global function. In general, a potential game consists of a pair of global and private payoff functions, in which the former function evaluates how good the current state of the system is, and the other determines the current payoff of each node. To define these functions, we first give a global function, then derive a private payoff function in terms of the scheme of Wonderful Life Utility (WLU) [5], [14], [27] (cf., also Ref. [18] for an overview of game theory). By this process, our private payoff is defined as a WLU, where each agent receives a reward for what it accomplished which would not have been accomplished without his presence. Thus, in our model, increasing the payoff to any virtual node yields a better system state with respect to power consumption.

We evaluated the performance of our method both via simulation and with a prototype implementation in terms of the running time of active physical nodes for estimated power consumption, response time, and migration cost. In the experiments conducted, we considered various changes in the environment that were difficult to deal with in our previous method [7]. The results of the experiments showed that our method saves, on average, 12.7%–18.7% of the running time of the disks in active mode compared with static configurations (i.e., all disks are always in active mode). This means that our method reduces power consumption by 11.1%–16.4%. Further, the average response times were in the range 50–190 ms. These results indicate that our method skews the workload while maintaining overall response time in a preferred range by setting suitable parameters in the utility functions. In addition, the results confirm that our method improves the flexibility issue remaining in Ref. [7]. On the other hand, we also observed that a few responses were heavily delayed. A major cause is that some virtual nodes were not able to keep up with the change in workload, and were boxed in the current disks before they were overloaded at peak time. We shall address this issue in future work towards practical use of our method.

Finally, we note that this paper is a revised and extended version of Ref. [9] and includes new simulation results and explanations that were not presented previously because of space limitations.

This paper is organized as follows. Section 2 discusses related work. Section 3 presents our proposed game-theoretic power-saving method. Sections 4 and 5 outline the evaluations conducted using simulations and a prototype implementation, respectively. Finally, Section 6 concludes this paper and outlines ideas for future work.

2. Related Work

There have been a number of suggestions for power-saving in storage systems (cf., Ref. [1] for a comprehensive survey of this research area). As explained in the previous section, these are based on similar ideas, but are classified into the following categories according to variations in their approaches.

The first category focuses on popularity (i.e., frequency of access) of the data. More specifically, popular data are gathered into a subset of disks by migrating data across disks. PDC [19] sorts files according to data access frequency, and the files are placed across the ordered disks of an array according to their latest access frequencies. Hibernator [30] applies the idea of PDC to RAID and dynamic rotations per minute (DRPM) systems.

The second category uses redundancy (i.e., data replication) for workload consolidation and/or diverting disk access. In DIV [20], original and redundant data are separated onto different disks, thereby allowing read/write requests to be concentrated on those disks that contain the original data. RIMAC [28] provides two layered caches, one for storing storage data and the other for parity conservation. EERAID [12] is based on the idea of diverted accesses in the context of RAID-1 and RAID-5. PARAID [26] is also a power-saving technique for RAID, that allocates the replica in a specific way, meaning that data are collected/spread to adapt to changes in operational workloads. Another well-known technique is eRAID [25], which is based on a quite similar idea as EERAID.

The final category uses caching and buffering to avoid disk access, thereby extending the idle time. Thus, although caching was originally introduced to improve performance, it also facilitates disk power conservation. A typical example is MAID [2], which provides specific disks to act as a cache to store frequently accessed data, thereby reducing access to the other disks. The energy conservation algorithms PA- and PB-LRU [31] operate by replacing data blocks when a cache miss occurs. Pergamum [22] uses non-volatile random-access memory (NVRAM) to buffer write accesses and store data signatures, thereby reducing direct access to the disks.

The studies cited above restricted their scope to storage systems consisting of a relatively small number of disks. However, in recent years the target in this research area has shifted to datacenter-scale systems. GreenHDFS [10] is a recent study in the first category. It targets Hadoop distributed file system (HDFS) and divides disks into hot and cold zones based on the idea underlying PDC. Lightning [11] is also classified into the first category. This technique is designed for distributed cloud storage file systems and closely resembles GreenHDFS, but features four zones instead of two. Our previous work [7] is also included in this category, which is based on the DHT technique and dynamically changes the allocation of its virtual nodes in a specific way. Recent proposals in the second category include sample-replicate-consolidate mapping (SRCMap) [23] and sliding window replica strategy (SWIN) [24].

The motivation for this research is in line with those of the recent studies mentioned previously. In particular, our research can be classified into the first category and considered a direct successor of Ref. [7]. The main difference between their approach and ours is that our method does not require any kind of central controller for data allocation, which enables dealing with various changes in the system environment. For example, in the case of Lightning, it employs various super nodes, such as the data migrator, power manager, and data placement manager. Although Lightning is demonstrated to be applicable to cloud storage of a similar size to that of our target for the reduction of its power consumption, such a relatively complex architecture may hinder system flexibility. For example, changing the configurations in many of the super nodes is requested when disks are added/removed. In addition, super nodes may become a performance bottleneck (i.e., response time) when the system is scaled up. In contrast to such a centralized approach, our method is fully decentralized, so the inherent advantage of the distributed hash table remains applicable to our method.

Finally, we briefly discuss the relationship between our method and virtual machine (VM) consolidation techniques for reducing power consumption in datacenters. In recent years, VM consolidation has been studied intensively (cf., Ref. [15] for a survey of this research area). The idea behind VM consolidation is to gather VM instances into few physical machines, enable other machines to be turned off, and thus save on energy. This idea is quite similar to that of most power-saving techniques for storage systems. A major difference between our approach and that taken in many VM consolidation techniques is that our approach is fully decentralized. This characteristic equates to an advantage in scalability and flexibility. However, various inherent aspects of VM should also be considered, such as CPU utilization and dependency between VMs, when our idea is applied to VM consolidation.

3. System Architecture

Our proposed method is targeted at storage systems comprising hundreds of physical disks. In this section, we first describe the underlying system, and then give an informal explanation of the game theoretic approach underlying the autonomic popular data concentration mechanism. Subsequently, we present a formal decision-making model for the virtual nodes, and finally describe the migration algorithm.

3.1 Underlying System

In our system, a basic lookup service is provided by an underlying DHT mechanism. The DHT enables clients to access data by sending a request to any of the disks, regardless of their location. In addition, our method uses the concept of virtual nodes. A virtual node works as a single physical disk in the system, while each real physical disk may be responsible for multiple virtual nodes. As mentioned in Section 1, virtual nodes are used for load balancing in Refs. [3], [4], [21]. However, in our system, the mechanism is used to skew the workload via dynamic migration of stored data in combination with the lookup service.

3.2 The Central Idea

In our method, virtual nodes are considered selfish agents playing the following game. The virtual nodes are initially allocated in a physical disk array and provide a lookup service in collaboration with the other virtual nodes, as in the usual DHT. After a fixed period of time, each virtual node receives a payoff according to the workload of the physical disk on which it currently resides. Here, we note that the workload of a physical disk is the sum of the workloads of the virtual nodes stored on that disk. As explained in Section 1, this payoff is determined by the private function such that the higher the workload a disk has, the higher the payoff each virtual node on the disk receives. Conversely, they receive a negative payoff (i.e., penalty) if the disk on which they reside is overloaded. After receiving a payoff, each virtual node on the disk checks the workloads of their neighboring disks and independently chooses its strategy. More specifically, it moves to another disk that is expected to yield a higher payoff in the future, or it remains on the current disk. The decision-making and payoff process is repeatedly conducted at regular intervals. More precisely, virtual nodes on the same physical disk receive payoffs and decide on their strategies at the same time, but they do not have to be synchronized with other nodes on different disks.

Our model can be regarded as a potential game. More specifically, our model is developed according to the following steps. First, we introduce a global function that evaluates the state of the system in terms of power-saving and overloading. Next, using the scheme of WLU, we derive a local function, which determines the payoff to virtual nodes, from the global function. We formally describe these functions in the next subsection.

3.3 System Model as a Potential Game

3.3.1 Preliminaries

As preliminaries, we first introduce some notation and functions. In the following, $\mathbb{N}$ is used to denote the set of natural numbers. Let $P = \{p_1, p_2, \ldots, p_n\}$ be the set of physical nodes and $V = \{v_1, v_2, \ldots, v_m\}$ be the set of virtual nodes. Time progress is represented as discrete steps indicated by a natural number $t = 0, 1, 2, \ldots \in T$, where 0 indicates the time of the initial state. Every virtual node is stored on a physical node, with the allocation determined by the function $place^V : V \times T \to P$. Intuitively, $place^V_t(v) = p$ means that virtual node v is on p at time t. We also use this function to denote the set of virtual nodes $V' \subseteq V$ that are on a physical node p, i.e., $place^P_t(p) = V'$. Formally, this function can be defined by using $place^V_t$, i.e. $place^P_t(p) = \{v \in V \mid place^V_t(v) = p\}$. For each physical node, the capacities of workload and volume are respectively determined by the functions $cap_{load} : P \to \mathbb{N}$ and $cap_{vol} : P \to \mathbb{N}$. The current workload and total volume of stored data of a virtual node at any time are respectively determined by the functions $load : V \times T \to \mathbb{N}$ and $size : V \times T \to \mathbb{N}$. These functions are also used for physical nodes according to the following definitions: $load(p, t) = \sum_{v \in place^P_t(p)} load(v, t)$ and $size(p, t) = \sum_{v \in place^P_t(p)} size(v, t)$, respectively. Note that no physical node is allowed to have virtual nodes whose total stored data volume exceeds its volume capacity (i.e., $cap_{vol}(p) \leq \sum_{v \in place^P_t(p)} size(v, t)$ for any t). In addition, p is considered to be overloaded if $load(p, t) > cap_{load}(p)$.

3.3.2 Formal Definition of the Global Function

We first introduce the global function G to evaluate the state of the system. However, to respond to rapid changes in the system workload, it estimates the expected state at some later steps provided that the rate of change in the workload during the past j steps is the same in the future. The formal definition of the global function is as follows:

$$G(place^V_t) = \sum_{i=0}^{s} \sum_{p \in P} (load^2_{pred}(p, i, t) - c \cdot over^2_{pred}(p, i, t)).$$

Here, the value s indicates the number of steps considered to estimate the expected state. The function $load_{pred}(p, i, t)$ indicates the expected workload of p at time $i + t$ (i.e., i steps later), and is defined as follows:

$$load_{pred}(p, i, t) = load(p, t) + \frac{i}{j}(load(p, t) - load(p, t - j)).$$

The function $over_{pred}(p, i, t)$ indicates the level of overload for p, and is defined as follows:

$$over_{pred}(p, i, t) = \beta \cdot cap_{load}(p) - load_{pred}(p, i, t),$$

where β (with $0 < \beta \leq 1$) is a coefficient indicating the proportion of capacity that may be devoted to migration. For example, $\beta = 0.6$ means that the physical node may spare 60% of its capacity to respond to data access requests, while the remainder (i.e., 40% of its capacity) is kept for migration. c is a weighting constant with $c > (s + 1) \sum_{p \in P} cap^2_{load}(p)$.

In the definition of G, $load^2_{pred}(p, i, t)$ gives a positive evaluation for effectively skewing the workload, whereas $-c \cdot over^2_{pred}(p, i, t)$ gives a negative evaluation according to the level of overloading. We note that the argument of G (i.e., $place^V_t$) determines the allocation of virtual nodes at time t. Thus, in game-theoretic terminology, it is a strategy profile of the virtual nodes.

3.3.3 Formal Definition of the Local Function

Next, in terms of the scheme of WLU, the global function defined above can be used to derive a local function L_v for node $v \in V$ as follows.

$$\begin{aligned} L_v(place^V_t) = & \sum_{i=0}^{s}[(load^2_{pred}(place^V_t(v), i, t) \\ & -load'^2_{pred}(place^V_t(v), i, t)) \\ & -(c \cdot over^2_{pred}(place^V_t(v), i, t) \\ & -c \cdot over'^2_{pred}(place^V_t(v), i, t))]. \end{aligned}$$

Here, $load'_{pred}(p, i, t)$ and $over'_{pred}(p, i, t)$ denote the values obtained from $load_{pred}(p, i, t)$ and $over_{pred}(p, i, t)$, respectively, by removing v from p.

3.4 Migration Algorithm

Because the model introduced above is a potential game, any migration that improves a virtual node's payoff increases the value of the global function. However, owing to the limitation of network bandwidth in a system, it is difficult to allow all virtual nodes to migrate as they wish. Thus, we introduce a priority order according to the degree of contribution to improving the value of the global function (i.e., the amount by which the value of the global function increases).

As the degree of contribution, we consider the increase in the payoff of migration divided by the time required to complete the migration. The formal definition is as follows:

$$Mig_{eval}(v, p_{from}, p_{to}) = \frac{L_v(place') - L_v(place)}{Mig_{time}(v, p_{from}, p_{to})},$$

where p_{from} and p_{to} ($\in P$) respectively denote the source and destination of migration, and Mig_{time} represents the time required to migrate v from p_{from} to p_{to}. This function is defined as follows:

$$Mig_{time}(v, p_{from}, p_{to}) = \frac{diff(v, p_{from}, p_{to})}{\min\{bw(p_{from}), bw(p_{to})\}},$$

where $diff$ denotes the total amount of data that must be transferred for the migration of v from p_{from} to p_{to}, and bw represents the bandwidth of p, defined as $bw(p) = (1 - \beta)cap_{load}(p)$ (with $0 < \beta \leq 1$). Note that β is the same predefined constant as in the global function.

In a real system, the evaluation of payoffs and migration are autonomously controlled by the physical nodes on the basis that the virtual nodes behave as selfish agents. Roughly speaking, every physical node (denoted by p) independently collects information to evaluate the payoffs from its neighboring nodes. As we shall see in Section 4, the set of neighbors for each physical node is predetermined. Then, p selects the best migration, and sends a request to the destination. The destination node decides whether to accept or reject the request. If it is accepted, the migration starts immediately; otherwise, it is rejected. Note that, to reduce the migration cost, physical nodes are allowed to send or receive one virtual node at a time.

More precisely, every physical node (denoted by $p \in P$) independently executes the following steps at regular intervals. Here, the current time is indicated by $t \in T$.

(1) p obtains the following information from each of its neighboring nodes (denoted by $p' \in P$).
- Estimated workload: $load_{pred}(p', i, t)$;
- Amount of free space: $cap_{vol}(p') - size(p')$;
- The value of the migration that p is executing: $Mig_{eval}(v, p_{from}, p_{to})$, where v is the virtual node that is currently migrating from/to p'.

(2) Among all possible migrations for the virtual nodes in p and the migrations previously requested to p, find the best migration (whose virtual node is denoted by v) yielding the largest among $Mig_{eval}(v, p, p')$ (for the migration of a virtual node in p) and $Mig_{eval}(v, p', p)$ (for the migration previously requested). If there is no such v (i.e., $Mig_{eval}(v, p, p') \leq 0$ for any v), then p quits and returns to Step 1 in the next time interval. Note that the currently migrating virtual node is also included as a candidate.

(3) Execute one of (3-a)–(3-d) according to the current state:

(3-a) If v is currently migrating from p to p', then the migration continues;

(3-b) If $place^V_t(v) = p$ and p is asking its neighbor to receive v, but this has not yet been approved, then p continues to wait for a response;

(3-c) If $place^V_t(v) = p$ and p has not asked its neighbor to receive v, then p drops its current request that was previously sent to a neighbor, and asks the destination to receive v.

(3-d) If $place^V_t(v) \neq p$, then p drops the request that was previously sent by p to a neighbor and the migration currently being executed, and then approves the migration of v;

(4) Reject all other requests from the neighbors.

In closing this section, we comment on the migration cost. Instead of moving all data stored in a virtual node, there is an alternative that reduces the volume of data to be migrated. In other words, when a virtual node is removed from a disk, the stored data remain in the disk and are reused at the time of the next migration. This enables the migration to proceed by copying only those data that are different from the previous residence at that physical node. Indeed, the advantage of this technique is the trade-off with the disk space. However, if the system has enough space on the disk and can afford to create some redundancy, this technique can be useful for effective migration. We adopt this technique in our simulation and experiments as reported in later sections.

4. Simulation Results

We developed a simulator of a storage system based on our proposed method that mimics the implementation of a storage system comprising 1,000 physical disks. Using this simulator, we measured the change in the number of active physical nodes to estimate power consumption. In addition, we also measured the response time and the number of migrations. We next consider various changes in the environment, none of which our previous method [7] can cope with, and demonstrate that our method overcomes these limitations below.

4.1 Parameters and Settings

In the evaluation, we considered a system consisting of 1,000 physical nodes whose volume and workload (i.e., maximum transfer rate of data access) capacities are 500 GB and 55 MB/s, respectively. As per the suggestion in Ref. [21], the number of virtual nodes is 10 times the number of physical disks. The system stores 200 million files, each of size 500 KB, and these are randomly allocated between the virtual nodes. In the intended usage environment, we assumed that the system workload varies over a day. By setting the maximum transfer rate of data access and the file size, we consider the capacity of the workload for each physical disk to be 110 accesses/s. In other words, if a physical disk is accessed more frequently than this capacity, the response latency increases gradually with the disk queue length.

During the course of a day, as modeled by discrete time intervals t (s) with $0 \le t \le 3{,}600 * 24$, the workload of all virtual nodes is initially at its lowest, then increases until the middle of the day, before decreasing until the end, where the gap is 4-fold. As suggested by many studies (e.g., Refs. [4], [17], [19]), we assumed that the data access frequency conforms to a Zipf distribution [32]. According to this law, the access frequency of the r-th most popular file is determined by the following formula:

$$load_r(t) = load_{sys}(t)\frac{\frac{1}{r^{\alpha}}}{\sum_{i=1}^{F}\frac{1}{i^{\alpha}}},$$

where F is the total number of files, $load_{sys}$ indicates the total number of data accesses, and α is a coefficient. In our simulation, we consider the case where $\alpha = 0.7$. Thus, the workload of the system is determined by the following formula:

$$load_{sys}(t) = \begin{cases} 22{,}000 * (\frac{3t}{12*3{,}600} + 1) & (0 \le t < 12*3{,}600) \\ 22{,}000 * (-\frac{3t}{12*3{,}600} + 7) & (12*3{,}600 \le t < 24*3{,}600). \end{cases}$$

For example, the total number of accesses is 22,000 during the off-peak period, rising to 88,000 at peak time. Note that if we consider a static configuration (i.e., without our method) in which the workload is ideally balanced among disks, making 800 physical nodes active is needed to avoid overloading the system because of the total number of accesses and the load capacity of disks (In fact, as discussed in the next subsection, realizing complete load balancing is difficult. In a real environment, more disks are therefore needed).

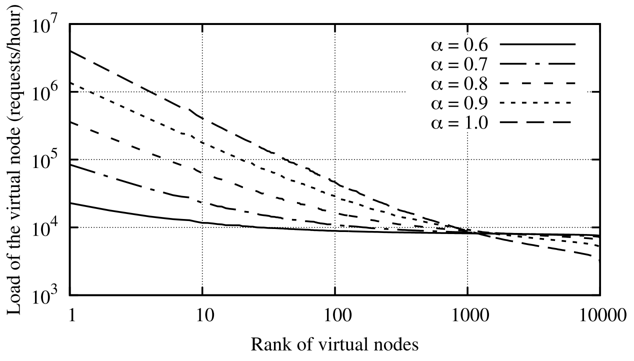


Fig. 1 Popularity distribution of the virtual nodes (in the case without replication).

As mentioned in Section 3, after migrating a virtual node, the stored data remain at the source physical disk unless it overflows. Thus, we reuse the remaining unchanged data when the virtual node returns to its original position. We assumed that 10% of files are rewritten by write accesses during the course of a day.

Finally, the remaining parameters were as follows: In the global function defined in Section 3.3, we set $s = 30 * 60$, $\beta = 0.9$, $j = 15 * 60$. The neighbors for each physical node p_i are $\{p_j \mid i - 50 \le j \le i - 1, i + 1 \le j \le i + 50\}$ (i.e., the number of neighboring physical nodes is 100). Note that the future workload was estimated using only the workloads of 15 minutes ago.

4.2 Popularity Distribution of the Virtual Nodes

As stated in the previous subsection, the frequency of file accesses conforms to Zipf's law in the simulation, but the popularity distribution of the virtual nodes is not clear. Consequently, as a preliminary study we analyzed the change in the popularity distributions of virtual nodes when the coefficient α of the Zipf's equation varies from 0.6 to 1.0.

Figure 1 shows the resulting distribution. It is clear that, in the case where $\alpha = 1.0$, the most frequently accessed virtual node is approximately 1,000 times more popular as the node that is least accessed. At the same time, the access frequencies of the most popular file and the most popular virtual node are respectively 4,025,710 requests/hour and 4,022,130 requests/hour. This indicates that the workloads of popular virtual nodes are sometimes caused by a few files being very heavily accessed. Moreover, it is easy for such a file to yield an overload that cannot be fixed by data migration.

To alleviate excessive workload skew, we introduce load balancing among virtual nodes by means of replication of files. A thorough explanation of the implementation of this mechanism is beyond the scope of this paper. However, in the initial simulation settings, we consider the following preliminary data allocation rearrangement. First, as in the usual DHT mechanism, all files are randomly allocated on the disks, and all virtual nodes are also randomly assigned to physical disks. Next, if a physical disk is overloaded, then a replica of the most frequently accessed file in this disk is created and placed on the disk that is accessed the least

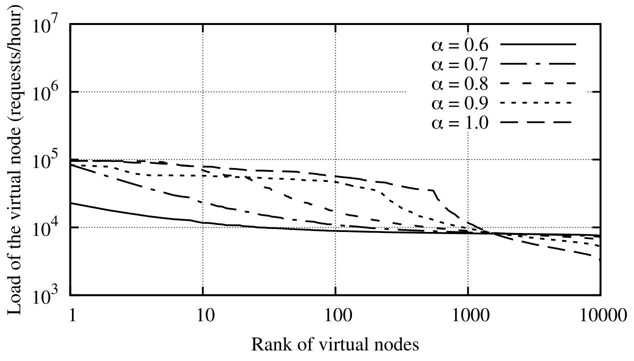


Fig. 2 Popularity distribution of the virtual nodes (in the case with replication).

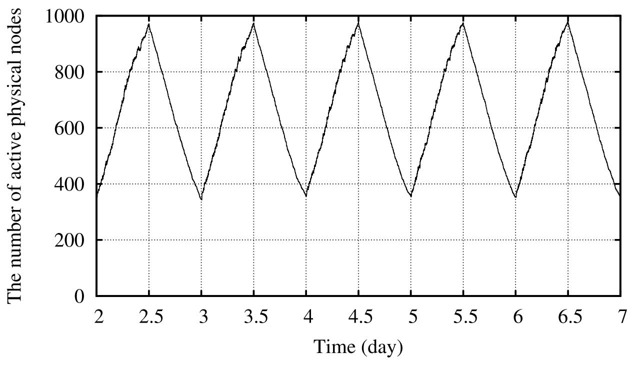


Fig. 3 Number of active physical nodes.

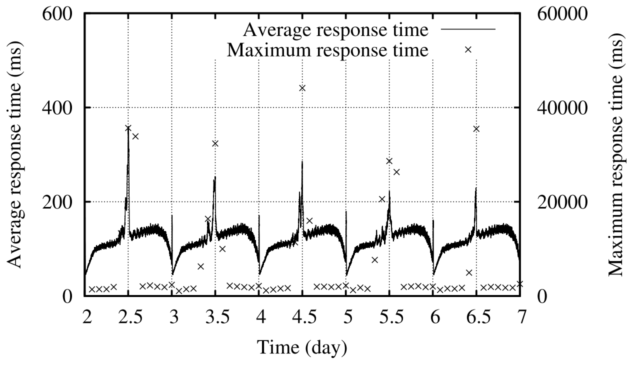


Fig. 4 Response time.

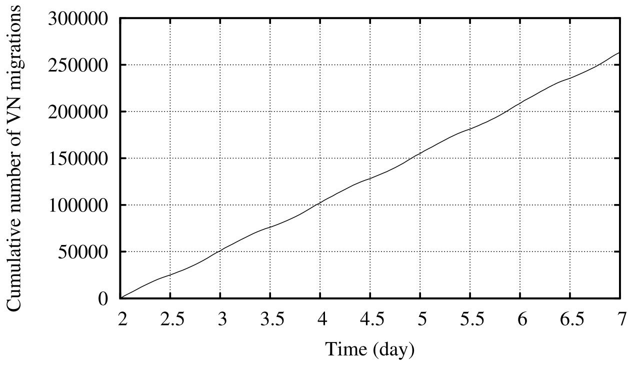


Fig. 5 Cumulative number of migrations.

among all the disks. This process is repeated until no overloaded physical disk remains.

Figure 2 shows the results of popularity distributions following this rearrangement process. In the cases where α = 1.0, 0.9, and 0.8, respectively, 420, 138, and 9 replicas are introduced.

4.3 Power Reduction

Figure 3 shows the change in the number of active physical nodes for the 5 days following the initial configuration. Throughout this section, we omit results from the first 2 days, because from the third day, reusable data are spread across the system as a result of the migrations in the first 2 days. Figure 3 shows that the number of active physical nodes, indicating the total running time, was reduced by an average of 17.3% relative to the power consumption in the system consisting of 800 physical nodes with a static configuration (i.e., all disks are always in active mode). Here, in our system, the state transition of physical disks between active and standby mode occurs less frequently. Thus, we can omit the energy requirement for state transitions and estimate the total power consumption by multiplying the durations of active and standby mode by the energy requirement per unit time. The specifications for a popular disk drive, specifically, Seagate Barracuda ST500DM002, indicate that the disk power in active and standby modes are respectively 6.57 W and 0.79 W. Therefore, in this setting, our method is expected to reduce the power consumption by 15.0%.

4.4 Response Time and Migration Cost

The evaluation of the response time was conducted by the following calculation. First, when a client accesses a physical node (say, p), the simulator estimates the current disk queue length based on the workload of p then calculates the queuing time of the request. Next, the simulator calculates the time for file transfer based on the maximum bandwidth for file transfer and the file size. The response time is expressed as the sum of the times calculated above. We here assume that a physical node can receive multiple data access requests but cannot process them simultaneously. We note that, in our method, the proportion of capacity that may be devoted to migration is predetermined, thus the response time is not affected by migration traffic.

Figures 4 and **5** show the respective changes in the overall average and maximum response times and the cumulative number of migrations over the 5 days.

Figure 4 shows that the overall average response time was 121 ms, although the maximum was 44,134 ms. The results of this experiment show that roughly the same number of virtual nodes always migrated, regardless of the system workload. For each virtual node, there were, on average, 5.27 migrations. In our previous work [7], the number of migrations required of each virtual node in a similar environment was approximately two, which is less than one-half the number required using this method.

Our method has room for improvement in the response time during peak periods. A major reason for these problems is that some virtual nodes are slow at adapting to rapid changes in workload. To rectify this problem, as explained in Section 3, we introduce a mechanism to predict future workload and the virtual nodes decide on strategies according to the predicted value, instead of the current workload. However, our prediction mechanism is still naive. On the other hand, a number of techniques aimed at providing accurate prediction of future workloads from past access patterns have been proposed (Refs. [6] and [13] are

Table 1 Effect of the reduction of neighboring disks.

	Reduction of Nodes (Power) [%]	Ave. Res. [ms]	Max Res. [ms]
50 neighbors	17.0 (15.0)	129	97,469
20 neighbors	16.7 (14.7)	184	182,168

Table 2 Simulation results for Cases 1–4.

	Reduction of Nodes (Power) [%]	Ave. Res. [ms]	Max Res. [ms]
Case 1	14.4 (12.6)	142	79,638
Case 2	12.7 (11.1)	190	159,358
Case 3	18.7 (16.4)	130	157,345
Case 4	16.9 (14.8)	128	56,124

examples addressing this issue in the context of a YouTube video. The authors also investigate in Ref. [8]). We believe that applying these techniques to our method could prove worthwhile in future work.

On the whole however, our method effectively skews the workload while maintaining a preferred overall average response time in large-scale systems.

4.5 Effect of the Reduction of Neighboring Disks

We also analyzed the effect of parameter changes on the performance. In this paper we show the simulation results in the case that the number of neighbors for each physical disk is reduced. Generally, this restriction is useful to reduce both the overhead of each physical node to calculate the best migration and the internal network traffic.

In this study the cases of 50 and 20 were evaluated, and the results for each case are summarized in **Table 1**. Here, "Nodes (Power)", "Ave. Res.", and "Max Res." denote the results of reducing the number of active physical nodes and power consumption (shown in parentheses) relative to the static configuration, overall average response time, and maximun response time, respectively. These results indicate that although the restriction worsened the maximum response time, the effect was quite limited to a small number of responses and there was little degradation in the power reduction rate and overall average response time.

4.6 Flexibility toward Various Environments

To evaluate our method in various environments, we considered the following cases, which cannot be treated using our previous method [7]:

Case 1: Peak times of some virtual nodes are different from others.
Case 2: Distribution of popularity changes over time.
Case 3: Some physical nodes are added to the system.
Case 4: Some physical nodes are removed from the system.

For Case 1, we considered the environment where 80% of the virtual nodes encounter peak time at noon, while for the others this occurs at midnight. For Case 2, we changed the difference in workload between peak and off-peak times to be 8-fold for 10% of the virtual nodes, and to be a factor of 32/9 for the other 90%. For Cases 3 and 4, we added 50 and removed 30 physical nodes, respectively.

The results for each case are summarized in **Table 2**. Overall, though the performance worsened in some cases, our method was able to effectively skew the workload, adapting to various changes in environment.

4.7 Remark on the Impact of Network Traffic on Performance

In our simulation, we actually omitted the cost for a lookup service provided by the underlying DHT and the cost of information exchange between physical disks on the response time. According to Ref. [21], the impact of the former on response time is expected to increase a further few hundred milliseconds and does not degrade performance in actual operation. Also, the impact of the latter is limited, because the amount of data for information exchange is small as compared to the amount of files transferring to clients. In addition, information exchange between disks is localized and not executed very often.

5. Implementation Experiments

We conducted experiments with the current prototype implementation to evaluate the applicability of our method to real systems. In the experiments, we measured the change in the number of active physical nodes and the response time in an environment with a variable system workload.

Our prototype consisted of 28 PC servers, each of which was equipped with Intel Core i7 2.67 GHz × 8 CPUs, 11.8 GB memory, and a single 500 GB ATA disk. In this implementation, we omitted the DHT mechanism.

5.1 Parameters and Settings

The system consisted of 300 virtual nodes, with each storing 5.6 million files of 500 KB size each. Before conducting the experiments, as a preliminary step, we measured the load capacity of the disks in this real environment and observed that it was 60 accesses/s. In this experiment, the distribution of data access frequency was also assumed to conform to Zipf's law. Basing on the load capacity of disks, we set the system workload as 336 accesses/s at off-peak times, and this increased to 1,344 accesses/s at peak times, a value similar to that in the simulations. The remaining parameters were set as the same as in the case of the simulation described in Section 4.1, i.e., $s = 30$, $j = 15$, and $\beta = 0.9$.

For this setting, if we consider a static configuration, making 23 physical nodes active is necessary to deal with the workload because of a calculation similar to the simulations (i.e., $22.4 \approx 1,344/60$).

5.2 Number of Active Physical Nodes

Applying this configuration, we observed the performance for 24 hours. **Figure 6** indicates the change in the number of active physical nodes. The figure shows that our method reduced the running time of active physical nodes by an average of 14.1% relative to the static configuration. This means that the disks in our system consumed about 13.5% less energy. This result shows that our method adjusts the number of physical nodes to changes in workload, and reduces power consumption effectively.

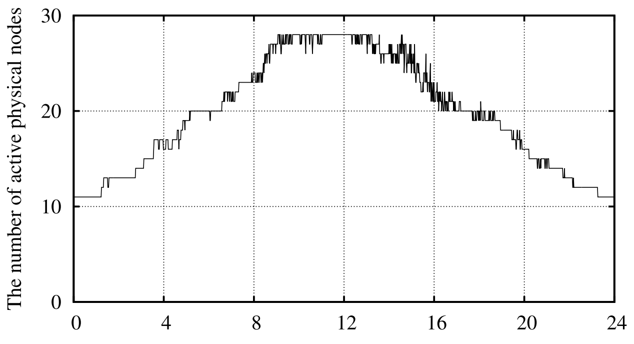


Fig. 6 Number of active physical nodes.

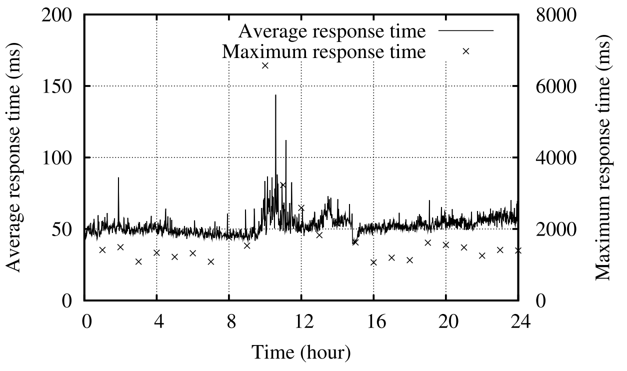


Fig. 7 Response time.

5.3 Response Time

Figure 7 indicates the change in the average and maximum response times. This figure shows that the overall average response time was 50 ms, while the maximum was 6,573 ms. We observed some long delays in responses at peak time but on the whole, our method retains an intended latency in the real system.

6. Conclusions and Future Work

In this paper, we presented a game theoretic approach for reducing power consumption in large-scale storage systems. To enhance the scalability in a commonly-observed technique, our method was based on DHTs to dynamically migrate virtual nodes, thereby skewing the workload and avoiding overloads. Moreover, to improve the flexibility problem in the method of Ref. [7], we investigated a technique that allowed virtual nodes to freely migrate to any neighboring disks, thereby enabling autonomous adaptation to various changes of environment, such as changes in the popularity distribution, extension of a system, and node failures. For this objective, the idea behind our method was to consider virtual nodes as selfish agents, and play a game in which each node receives a payoff according to the workload of the disk on which it currently resides. This idea was modeled as a potential game using the WLU scheme. We also evaluated the performance of our method using simulations and a prototype implementation. The results showed that our method effectively skewed the workload while maintaining a preferred response time in a large-scale distributed environment.

In future work, we will improve the migration algorithm so as to reduce the latency at peak-time. We are also interested in an evolutional mechanism to allow virtual nodes to autonomously change the parameters of local payoff functions according to the current environment, with the aim of achieving a better optimization.

References

[1] Bostoen, T., Mullender, S. and Berbers, Y.: Power-Reduction Techniques for Data-Center Storage Systems, *ACM Computing Surveys*, Vol.45, No.3, Article No.33 (2013).

[2] Colarelli, D. and Grunwald, D.: Massive arrays of idle disks for storage archives, *ACM/IEEE Conference on Supercomputing*, pp.1–11 (2002).

[3] DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P. and Vogels, W.: Dynamo: Amazon's highly available key-value store, *21st ACM SIGOPS Symposium on Operating Systems Principles*, pp.205–220 (2007).

[4] Godfrey, B., Lakshminarayanan, K., Surana, S., Karp, R. and Stoica, I.: Load balancing in dynamic structured p2p systems, *INFOCOM*, Vol.4, pp.2253–2262 (2004).

[5] Gopalakrishnan, R., Marden, J.R. and Wierman, A.: An architectural view of game theoretic control, *SIGMETRICS Perform. Eval. Rev.*, Vol.38, No.3, pp.31–36 (2010).

[6] Gursun, G., Crovella, M. and Matta, I.: Describing and Forecasting Video Access Patterns, *30th IEEE International Conference on Computer Communications* (*INFOCOM 2011*), pp.16–20 (2011).

[7] Hasebe, K., Niwa, T., Sugiki, A. and Kato, K.: 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).

[8] Okoshi, J., Hasebe, K. and Kato, K.: Power-Saving in Storage Systems for Internet Hosting Services with Data Access Prediction, *4th International Green Computing Conference* (*IGCC 2013*), p.10 (2013).

[9] Hasebe, K., Sawada, T. and Kato, K.: Using a potential game for power reduction in distributed storage systems, *IEEE International Workshop on Software Defined Systems* (*SDS 2014*), pp.550–555 (2014).

[10] Kaushik, R.T. and Bhandarkar, M.: GreenHDFS: Towards an energy-conserving, storage-efficient, hybrid Hadoop compute cluster, *2010 international conference on Power aware computing and systems* (*HotPower'10*), pp.1–9 (2010).

[11] Kaushik, R.T., Cherkasova, L., Campbell, R. and Nahrstedt, K.: Lightning: Self-adaptive, energy-conserving, multi-zoned, commodity green cloud storage systems, *19th ACM International Symposium on High Performance Distributed Computing* (*HPDC'10*), pp.332–335 (2010).

[12] Li, D. and Wang, J.: EERAID: Energy efficient redundant and inexpensive disk arrays, *11th ACM SIGOPS European Workshop*, No.29, p.6 (2004).

[13] Li, H., Ma, X., Wang, F., Liu, J. and Xu, K.: On popularity predicion of video shared in online social networks, *22nd ACM international conference on Conference on information and knowledge management*, pp.169–178 (2013).

[14] Marden, J.R. and Wierman, A.: Distributed welfare games, *Operations Research*, Vol.61, No.1, pp.155–168 (2013).

[15] Mastelic, T., Oleksiak, A., Claussen, H., Brandic, I., Pierson, J.-M. and Vasilakos, A.V.: Cloud Computing: Survey on Energy Efficiency, *ACM Computing Surveys*, Vol.47, No.2, pp.33:1–33:36 (2015).

[16] Monderer, D. and Shapley, L.S.: Potential games, *Games and Economic Behavior*, Vol.14, No.1, pp.124–143 (1996).

[17] Okoshi, J., Hasebe, K. and Kato, K.: Power-Saving in Storage Systems for Internet Hosting Services with Data Access Prediction, *4th International Green Computing Conference* (*IGCC 2013*), p.10 (2013).

[18] Osborne, M.J. and Rubinstein, A.: *A course in game theory*, The MIT Press (1994).

[19] Pinheiro, E. and Bianchini, R.: Energy conservation techniques for disk array-based servers, *International Conference on Supercomputing*, pp.68–78 (2004).

[20] Pinheiro, E., Bianchini, R. and Dubnicki, C.: Exploiting redundancy to conserve energy in storage systems, *ACM SIGMETRICS Conference on Measurement and modeling of computer systems*, pp.15–26 (2006).

[21] Stoica, I., Morris, R., Karger, D., Kaashoek, M.F. and Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications, *ACM SIGCOMM*, pp.149–160 (2001).

[22] Storer, M., Greenan, K., Miller, E. and Voruganti, K.: Pergamum: Replacing tape with energy efficient reliable, disk-based archival storage, *USENIX Conference on File and Storage Technologies* (*FAST'08*), pp.1–16 (2008).

[23] Verma, A., Koller, R., Useche, L. and Rangaswami, R.: SRCMap: Energy proportional storage using dynamic consolidation, *8th USENIX*

Conference on File and Storage Technologies (*FAST'10*), pp.154–168 (2010).

[24] Vrbsky, S.V., Lei, M., Smith, K. and Byrd, J.: Data replication and power consumption in data grids, *2010 IEEE Second International Conference on Cloud Computing Technology and Science* (*Cloud-Com'10*), pp.288–295 (2010).

[25] Wang, J., Zhu, H. and Li, D.: eRAID: Conserving energy in conventional disk-based raid system, *IEEE Trans. Computers*, Vol.57, No.3, pp.359–374 (2008).

[26] Weddle, C., Oldham, M., Qian, J., Wang, A., Reiher, P. and Kuenning, G.: PARAID: A gear-shifting power-aware RAID, *USENIX Conference on File and Storage Technologies* (*FAST'07*), pp.245–260 (2007).

[27] Wolpert, D.H. and Tumer, K.: An introduction to collective intelligence, *Handbook of Agent Technologies*, AAAI (1999).

[28] Yao, X. and Wang, J. RIMAC: A novel redundancy-based hierarchical cache architecture for energy efficient, high performance storage systems, *ACM SIGOPS/EuroSys European Conference on Computer Systems*, pp.249–262 (2006).

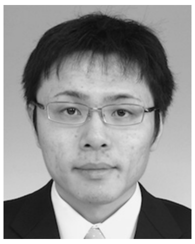
[29] Zhang, H., Wen, Y., Xie, H. and Yu, N.: *Distributed Hash Table: Theory, Platforms and Applications*, Springer (2013).

[30] Zhu, Q., Chen, Z., Tan, L., Zhou, Y., Keeton, K. and Wilkes, J.: Hibernator: Helping disk arrays sleep through the winter, *ACM symposium on Operating systems principles*, pp.177–190 (2005).

[31] Zhu, Q. and Zhou, Y.: Power-awere storage cache management, *IEEE Trans. Computers*, Vol.54, pp.587–602 (2005).

[32] Zipf, G.K.: *Human behavior and the principle of leZast effort*, Addison-Wesley Press (1949).

Koji Hasebe received his Bachelor's, Master's, and Ph.D. degrees in Philosophy from Keio University in Japan in 1998, 2000, and 2006, respectively. He is an assistant professor at the Department of Computer Science of the University of Tsukuba. His research interests include formal verification, distributed systems, game theory, and computer security.



Takumi Sawada received his Bachelor's degree in Engineering from Advanced Course of Gifu National College of Technology in 2012. He received his Master's degree in Engineering from the University of Tsukuba, Japan in 2014. He is currently at DWANGO Co., Ltd. His research interests include distributed systems and storage systems. He contributed to this study in his Master's course.

Kazuhiko Kato received his Bachelor's and Master's degrees in Engineering from the University of Tsukuba, Japan, in 1985 and 1987. He received his Ph.D. from The University of Tokyo, Japan, in 1992. He is currently a professor at the Department of Computer Science of the University of Tsukuba. His research interests include operating systems, distributed systems, and secure computing. He received distinguished paper awards from JSSST and IPSJ in 2004 and 2005, respectively.