A Flexible Data Migration Strategy for Power Savings in Distributed Storage Systems

Koji Hasebe, Sho Takai and Kazuhiko Kato
Department of Computer Science, University of Tsukuba, 1-1-1, Tennodai, Tsukuba 305-8573, Japan

Keywords: Power Savings, Distributed Storage Systems, Data Migration.

Abstract: We present a power-saving technique for datacenter-scale distributed storage systems. In particular, we focus on storage in environments where a large number of data are continuously uploaded, as typified by the platforms of social networking services. In achieving this objective, the main idea is to use virtual nodes and migrate them dynamically so as to skew the workload toward a small number of disks without overloading them. We improve this previously introduced idea by making the data migration strategy flexible in the present study. As a result, our proposed technique can maintain a high-load aggregation rate during the continuous addition of disks, which was difficult to handle in our previous study. The performance of our technique is evaluated in simulations. The results show that our technique improves the technique in the previous study and effectively skews the workload during a constant massive influx of data.

1 INTRODUCTION

Data sharing services, as typified by social networking services, are rapidly developing. In the datacenters of such services, a large number of data are continuously uploaded from users around the world; e.g., every minute, users share 300,000 tweets on Twitter (Telegraph, 2013), 680,000 pieces of content on Facebook (statistics, 2013a), and 100 hours of video on YouTube (statistics, 2013b). (Cf. also (Tatar et al., 2014).) High-performance, highly scalable, and low-cost (i.e., energy-saving) storage technologies are required to realize such data-intensive services.

There have been a number of attempts at reducing the power consumption of storage systems. A commonly observed feature of many of the developed techniques is the skewing of the workload toward a small number of disks, thereby allowing other disks to be in low-power mode. To realize this idea, a technique was proposed to migrate the stored data dynamically in such a way that all data are gathered onto as few disks as possible without overloading them (Hasebe et al., 2010). More specifically, for dynamic migration, data stored at virtual nodes are managed using a distributed hash table (DHT). In that setting, data migration was managed by rules for the gathering or spreading of virtual nodes according to the daily variation of the workload so that the number of active physical nodes was reduced to a minimum.

However, most previous studies either explicitly or implicitly assumed that the set of stored data is fixed, whereas this assumption is not valid for many real datacenter-scale platforms of data sharing services. In the study (Hasebe et al., 2010), for example, the rules for migration uniquely defined how each virtual node moves to a physical node on a fixed number of disks.

To tackle the above issue, we propose in this paper a power-saving technique for distributed storage systems based on the technique introduced by (Hasebe et al., 2010). The basic idea is to improve the data migration strategy so that the destination of the virtual node can be chosen from multiple options according to the current state of the system. As a result, when a new disk is added, data may be moved to the disk as necessary.

The performance of our proposed technique is measured by simulation in terms of the average load and the number of active physical nodes by comparing with the performance of the previous technique. In the simulations, we observed that our improved technique makes possible to aggregate the load at the intended value. Also, our technique effectively skews the workload even in environments where a vast amount data are continuously uploaded.

The remainder of the paper is organized as follows. Section 2 presents related work. Section 3 describes the underlying system. Section 4 presents our

Hasebe, K., Takai, S. and Kato, K.
A Flexible Data Migration Strategy for Power Savings in Distributed Storage Systems.
DOI: 10.5220/0006809803520357
In *Proceedings of the 7th International Conference on Smart Cities and Green ICT Systems (SMARTGREENS 2018)*, pages 352-357
ISBN: 978-989-758-292-9

data migration strategy for power reduction. Section 5 presents the simulation results. Finally, Section 6 concludes the paper and presents future work.

2 RELATED WORK

There have been a number of suggestions for reducing the power used by storage systems. These suggestions have a similar basis as mentioned in the previous section, but can be classified into two categories according to their approach.

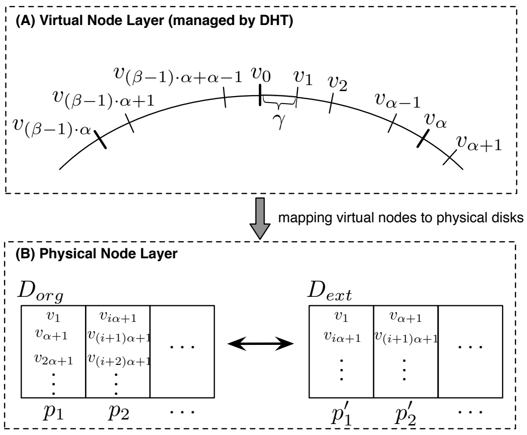
The first category uses data replication (i.e., redundancy). DIV (Pinheiro et al., 2006), for example, separates original and redundant data onto different disks, thereby allowing read/write requests to be concentrated on the disks with the original data. Hibernator (Zhu et al., 2005) and PARAID (Weddle et al., 2007) collect or spread data to adapt to changes in operational loads. Harnik et al. (Harnik et al., 2009) applied the idea of DIV to a large distributed storage system. Verma et al. (Verma et al., 2010) developed sample-replicate-consolidate mapping (SRCMap), which gathers accesses to the replicas on active disks, while Vrbsky et al. (Vrbsky et al., 2010) proposed a replication approach called the sliding window replica strategy (SWIN).

The second category dynamically migrates stored data. Massive Array of Idle Disks (Colarelli and Grunwald, 2002) provides specific disks that are used as a cache to store frequently accessed data, thereby reducing the number of accesses of other disks. PDC (Pinheiro and Bianchini, 2004) periodically reallocates data in the storage array according to the latest access frequencies. Kaushik et al. (Kaushik and Bhandarkar., 2010) proposed the idea of dividing disks in Hadoop distributed file systems into hot and cold zones.

The techniques used in our previous studies (Hasebe et al., 2010; Hasebe et al., 2015; Hasebe et al., 2016) are classified into the second category. The techniques introduced in (Hasebe et al., 2010; Hasebe et al., 2016) are based on the DHT and skew the workload by migrating virtual nodes. The present work improves the technique proposed in (Hasebe et al., 2010). In particular, our main motivation is to explore power savings in an environment where a vast number of data are continuously uploaded.

3 UNDERLYING SYSTEM

Our proposed technique is targeted at datacenter-scale storage systems. In particular, it is aimed at systems


(A) Virtual Node Layer (managed by DHT)
$v_{(\beta-1)\cdot\alpha+\alpha-1}$
v_0
v_1
v_2
$v_{(\beta-1)\cdot\alpha+1}$
$v_{\alpha-1}$
$v_{(\beta-1)\cdot\alpha}$
γ
v_α
$v_{\alpha+1}$
mapping virtual nodes to physical disks
(B) Physical Node Layer
D_{org}
D_{ext}
v_1 $v_{\alpha+1}$ $v_{2\alpha+1}$
$v_{i\alpha+1}$ $v_{(i+1)\alpha+1}$ $v_{(i+2)\alpha+1}$
v_1 $v_{i\alpha+1}$
$v_{\alpha+1}$ $v_{(i+1)\alpha+1}$
p_1 p_2
p'_1 p'_2
Figure 1: Underlying system.

that store data with relatively high access frequency. We assume that the system workload fluctuates over the cycle of a day, with the difference in workload between peak time and off-peak times is a factor of approximately 4 to 6. The basic idea of our technique is to use virtual nodes and to realize operations that store/retrieve data using the lookup mechanism of a DHT. (Here we will explain the use of Chord (Stoica et al., 2001) as an example of a DHT.) In addition, according to the load of the system, virtual nodes are migrated dynamically among physical disks.

As preliminaries, we first introduce notations and functions to describe the configuration of our target system. Let $P = \{p_1, \ldots, p_n\}$ and $V = \{v_1, \ldots, v_m\}$ be the sets of physical nodes (i.e., disks) and virtual nodes, respectively. As will be explained later, physical nodes are divided into two groups called the *original space* (denoted P_{org}) and *extended space* (denoted P_{ext}). The placement of each virtual node is described by the function $place_V : V \to P$. Intuitively, $place_V(v) = p$ means that virtual node v is located at physical node p. We also use the function $place_P : P \to 2^V$ to denote the set of virtual nodes $V' \subseteq V$ that are on a physical node p; i.e., $place_P(p) = V'$. Formally, the function can be defined using $place_V$; i.e., $place_P(p) = \{v \in V \mid place_V(v) = p\}$. For readability, we use the notation $v \in p$ to denote $v \in place_P(p)$.

In the system, the DHT has a key space consisting of $\alpha \cdot \beta \cdot \gamma$ keys as shown in Fig. 1 (A). The virtual nodes are arranged in this key space at equal intervals with width γ. Here, α, β, and γ shall be large enough for operation.

In the initial state, β virtual nodes $v_1, v_{\alpha+1}, \ldots, v_{(\beta-1)\cdot\alpha+1}$ are placed at specified positions of the key space. Furthermore, they are stored in order from the leftmost physical node in the disk array of P_{org} as shown in Fig. 1 (B). Data are written to one of the existing virtual nodes when uploaded by the client; i.e., the load of the write request from the client is distributed by β virtual nodes. It is here assumed that the virtual node has an upper limit of the total volume of stored data. When virtual node v_i (for $i = 0, \dots, \alpha \cdot \beta - 1$) becomes full, v_{i+1} is newly inserted at a predetermined position in the key space and stored at the empty physical node on the leftmost of P_{org}. Additionally, when the leftmost disk p_i in P_{org} becomes full, a new empty physical disk p_{i+1} is added.

When the system workload increases, each physical node independently checks its own workload, and if the workload exceeds the capacity, one of the virtual nodes is moved to a lowly loaded active physical node in P_{ext}. If there is no such physical node, one of the nodes in P_{ext} in low-power mode is activated. In contrast, when the system workload decreases, virtual nodes in P_{ext} are gradually moved back to the original positions. Finally, if a physical node has no active virtual node, it enters a low-power mode and thus reduces its power consumption. In our system, travel paths of the virtual nodes are recorded for 1 day. We denote the set of physical nodes in which a virtual node is placed during a day by the function $mig : V \to 2^P$. Intuitively, if a virtual node v originally placed at $p \in P_{org}$ is also placed in $p_2, p_3 \in P_{ext}$ by migrations over the course of the day, $mig(v) = \{p_2, p_3\}$.

We finally remark on the migration cost of our technique. To reduce the migration cost, instead of moving all data stored at a virtual node in each migration, remaining old data in P_{ext} are reused when the system workload increases again. This allows migration by copying the difference from the previous day. In the next section, we use the function $place_{Old} : V \to P$ to indicate the allocation of a virtual node on the previous day. More precisely, $place_{Old}(v) = p$ if virtual node v was placed on p by migration on the previous day.

4 DATA MIGRATION STRATEGY

The power reduction technique of (Hasebe et al., 2010) relies on two types of migration strategies for optimizing power consumption. These cope with the daily variation of the system workload, but the one is used when the workload is increasing and the other when the workload is decreasing.

In the original strategies, the destination of the virtual node from P_{org} to P_{ext} was rigorously fixed. It was therefore difficult to apply the technique to an environment where data are added sequentially. In this paper, we improve these strategies so that it can accommodate such environments.

4.1 Migration for Extension

When the system workload is increasing during the nominal period of a day, each active physical node (say, p_i) both in P_{org} and P_{ext} checks its own workload at regular intervals. If the workload exceeds the capacity (i.e., the maximum workload that can maintain a preferable response performance), then the virtual node $v \in place_P(p_i)$ to be moved is determined in the following way.

Case 1. There is a physical node $p \in P_{ext}$ with $place_{Old}(v) = p$.

Case 1-1: p is active, and v is moved to p.

Case 1-2: There is no such active p, and v is moved to p with activation.

Case 2. There is a physical node $p \in P_{ext}$ satisfying all the following conditions.

- C1: No two virtual nodes stored at a certain physical node move to the same physical node (i.e., $\forall v, v' \in p(mig(v) \cap mig(v') = \emptyset)$.
- C2: p does not exceed its volume and workload capacity even if v is moved to p.

Case 2-1: p is active, and v is moved to p.

Case 2-2: There is no such active p, and v is moved to p with activation.

Case 3: There is no physical node in P_{ext} satisfying both of the above two conditions, and a new disk (say, p') is added to P_{ext} and v is moved to p'.

Here we describe the improvement. The major difference from the previous research is that the proposed method flexibly determines the destination of the virtual node according to the situation in the proposed method. As a result, even if a disk is newly added, an appropriate destination can be found. Meanwhile, in the previous research, because the destination of the virtual node is uniquely determined in advance, it is not easy to add the disk.

In addition, our proposed technique is devised so as to maintain the advantage of the previous technique of efficiently aggregating the workload. For example, the reason for prioritizing the physical node in the active state as the migration destination of the virtual node is to avoid increasing the number of physical nodes in an active state. Condition C1 in Case 2 is introduced for a similar reason. That is to say, according to this condition, when the physical node selects its own destination of the virtual node, it is possible to generate more candidates for the destination.

We here present a simple example to clarify the process. (See also Fig. 2 for a graphical presentation.)

In the example, p_1, p_2, and p_3 are in P_{org} and the virtual nodes placed at these nodes are represented by

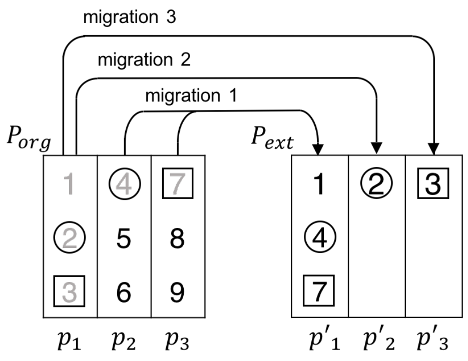


Figure 2: Example of migrations.

integers. We assume that only v_1 has been moved and, in P_{ext}, p'_1 is active while p'_2 and p'_3 are in low-power mode. In this setting, we consider the situation that the workloads of nodes p_1, p_2, and p_3 exceed the corresponding capacities. (In Fig. 2, the migrations of circled virtual nodes can reuse the old data remaining in P_{ext} while the migrations of the virtual nodes enclosed in squares involve newly written data in P_{ext}.)

Migration 1 is the migration of v_4 and v_7 from p_2, p_3 to p'_2, p'_3, respectively. v_4 is selected for p_2 because p'_1 is active and has reusable data, while v_7 is selected for p_3 and newly written in p'_1 because there are no reusable data in P_{ext}. Migration 2 is the case that p_1 moves another virtual node. p'_2 is selected at this time because it is in low-power mode yet stores reusable data (i.e., the data of v_2). Migration 3 is conducted because the workload of p_1 exceeds the capacity again. In this case, because p_1 has no other reusable data in P_{ext}, it is necessary to migrate all data of a virtual node. Moreover, because p'_1 and p'_2 have received v_1 and v_2, respectively, p'_3 is selected as the destination.

4.2 Migration for Reduction

When the system workload is decreasing following the day's peak workload, reducing power consumption requires gathering the widely dispersed virtual nodes into their original positions. To realize this mechanism, physical disks have been divided into groups and an optimization algorithm introduced to find migrations of the virtual node that result in the fewest active disks for each group (Hasebe et al., 2010). This optimization algorithm can be directly applied as it is to our technique explained so far. In our case, P_{ext} is divided into groups of approximately 10 physical nodes and the algorithm is executed for each group at regular intervals. In our setting, however, because the destinations of the virtual nodes are widely expanded irregularly, it is necessary to execute the algorithm more frequently than in the previous study.

5 EVALUATION BY SIMULATIONS

We developed a simulator that mimics a storage system targeted in this study. By using this simulator, in order to show that the proposed technique improves the technique of the study (Hasebe et al., 2010), we first compare the average load (i.e., the ratio of workload to capacity) of the changes in the active physical nodes and in the number of active physical nodes in an environment where the workload varies. Next, we evaluate the change of the avarage load under the circumstance where data are continuously uploaded.

5.1 Parameters and Settings

In the evaluation of this section, we considered the following environment which was similar to the setting considered in (Hasebe et al., 2010). The data were stored in 10,000 virtual nodes. During the course of a day (that is modeled by discrete time progress in 10 minutes), the workload of all virtual nodes was initially at its lowest, and increased until the middle of the day then decresed until the end, where the gap was sixfold. In addition, due to the popularity of stored data, we considered two groups of virtual nodes with different workloads: group G_1 of 2,000 nodes are the busier ones, while group G_2 of 8,000 were the normal nodes. In each group, the workloads of all virtual nodes were the same. The ratio of workloads of a node G_1 to a node in G_2 was denoted by α, and we considered the cases that $\alpha = 1.2$, 1.5, and 2. In each case, we set the initial workload of the system to be 60% of its capacity.

In the first simulation (presented in Section 5.2), as the environment to evaluate the proposed technique, we assumed that each of P_{org} and P_{ext} consisted of 100 physical nodes, and initially all virtual nodes were equally stored only in P_{org} (thus P_{ext} was empty). On the other hand, as the environment to evaluate the previous study, we assumed that the system consisted of six blocks each of which consisted of 100 physical nodes, and initiall all the virtual nodes were stored only in one block.

5.2 Comparison with Previous Study

Fig. 3 indicates the comparison of the change in the number of active nodes, while Fig. 4 indicates the comparison of the change in the average load of active physical nodes. These figures show that our proposed technique improves the efficiency of usage of the physical nodes. In the case of using proposed technique and the technique in the previous study (described by

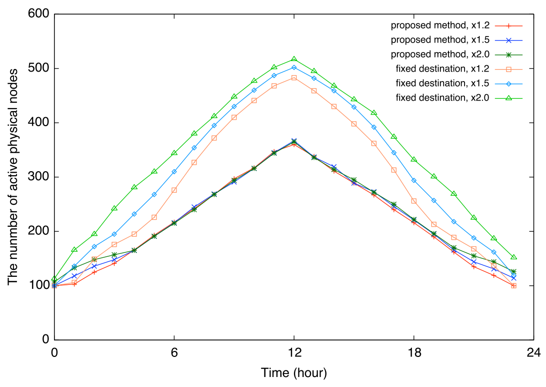


Figure 3: Number of active physical nodes.

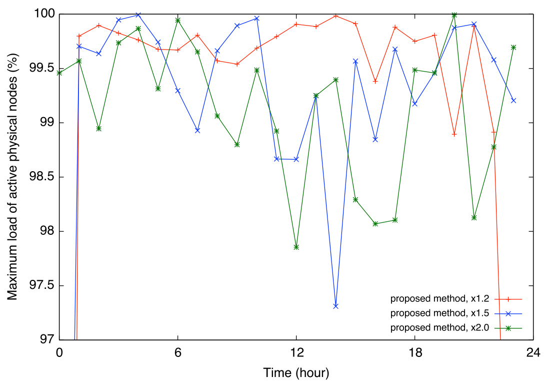


Figure 5: Maximum load of active physical nodes.

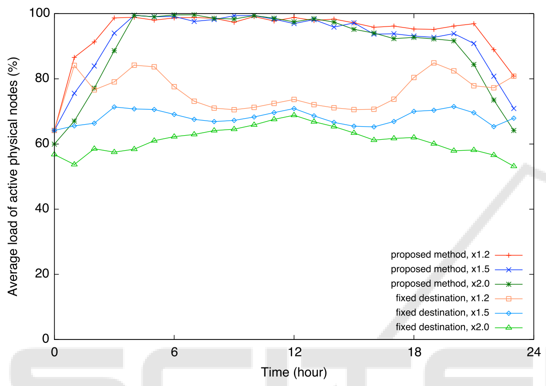


Figure 4: Average load of active physical nodes.

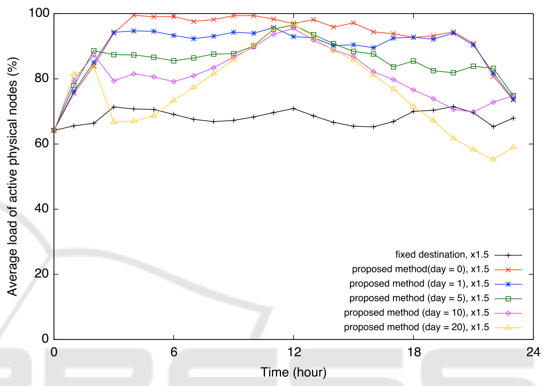


Figure 6: Average load of active physical nodes.

"fixed destination"), the average values of the daily load was approximately 100% and 76%, respectively when α is set as 1.2.

Also, the change in the maximum load on the active physical nodes in the case using proposed technique is shown in Fig.5. For each case of the values of α, the maximum load is about 100%.

The result of these simulations show that our proposed technique makes possible to aggregate the load at the intended value at any time during the course of a day.

5.3 Flexibility toward Sequential Addition of Data

In this simulation, we considered an environment similar to the simulation in Section 5.2. In addition we assumed that five new physical nodes each of which stored 100 virtual nodes were added to P_{org} a day and measured until P_{org} reached 200 physical nodes.

Figs. 6 and 7 indicate the daily change in the average load of the active physical nodes. The results show that the aggregation rate of load decreases with each day. However, the average load over 21 days is still about 82% and our proposed technique effectively skews the workload even in continuous addition of data.

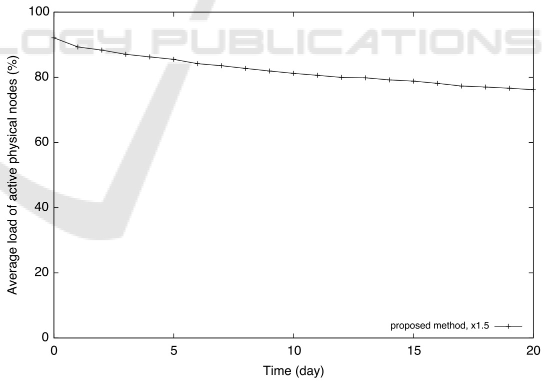


Figure 7: Daily change in the average load.

6 CONCLUSIONS AND FUTURE WORK

We presented a power-saving technique for datacenter-scale distributed storage systems. Our main motivation was to explore power savings in an environment where a vast number of data are continuously uploaded. The basis of our proposed technique was to improve a technique introduced by (Hasebe et al., 2010), which uses virtual nodes and migrates them dynamically. Our improvement was a modification of the data migration strategy so that the destination of the virtual nodes can be chosen from multiple options according to the current state of the system. Finally, the performance of our systems was evaluated in simulations. The results showed that our technique improves the technique in the previous study and effectively skews the workload during a constant massive influx of data.

In future work, we will develop a prototype implementation and evaluate its performance on real systems.

REFERENCES

Colarelli, D. and Grunwald, D. (2002). Massive arrays of idle disks for storage archives. In *ACM/IEEE Conference on Supercomputing*, pages 1–11.

Harnik, D., Naor, D., and Segall., I. (2009). Low power mode in cloud storage systems. In *IEEE International Symposium on Parallel and Distributed Processing*, pages 23–29.

Hasebe, K., Niwa, T., Sugiki, A., and Kato., K. (2010). Power-saving in large-scale storage systems with data migration. In *IEEE International Conference on Cloud Computing Technology and Science (CloudCom'10)*, pages 266–273.

Hasebe, K., Okoshi, J., and Kato., K. (2015). Power-saving in storage systems for cloud data sharing services with data access prediction. *IEICE Transactions*, E98-D(10):1744–1754.

Hasebe, K., Sawada, T., and Kato., K. (2016). A game theoretic approach to power reduction in distributed storage systems. *Journal of Information Processing*, 24(1):173–181.

Kaushik, R. T. and Bhandarkar., M. (2010). Greenhdfs: towards an energy-conserving, storage-efficient, hybrid hadoop compute cluster. In *2010 international conference on Power aware computing and systems (HotPower'10)*, pages 1–9.

Pinheiro, E. and Bianchini, R. (2004). Energy conservation techniques for disk array-based servers. In *International Conference on Supercomputing*, pages 68–78.

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

statistics, F. (2013a). https://newsroom.fb.com/News.

statistics, Y. (2013b). http://www.youtube.com/yt/press/statistics.

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

Tatar, A., de Amorim, M. D., Fdida, S., and Antoniadis, P. (2014). A survey on predicting the popularity of web content,. *Journal of Internet Services and Applications*, 5(8).

Telegraph (2013). http://www.telegraph.co.uk/technology/twitter/9945505/Twitter-in-numbers.html.

Verma, A., Koller, R., Useche, L., and Rangaswami, R. (2010). Srcmap: energy proportional storage using dynamic consolidation. In *8th USENIX Conference on File and Storage Technologies (FAST'10)*, pages 154–168.

Vrbsky, S. V., Lei, M., Smith, K., and Byrd, J. (2010). Data replication and power consumption in data grids. In *2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom'10)*, pages 288–295.

Weddle, C., Oldham, M., Qian, J., Wang, A., Reiher, P., and Kuenning, G. (2007). Paraid: a gear-shifting power-aware raid. In *USENIX Conference on File and Storage Technologies (FAST'07)*, pages 245–260.

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