A Distributed Spin-Down Algorithm for an Object-Based Storage Device with Write Redirection

Timothy Bisson Joel Wu Scott A. Brandt
Computer Science Department
University of California, Santa Cruz
{tbisson,jwu,scott}@cs.ucsc.edu

Abstract

We present a distributed spin-down algorithm for conserving energy in an object-based storage device (OSD) cluster. It differs from traditional spin-down algorithms in that it is cooperative—the decision to spin-down an OSD considers not only its own internal state, but also the state of the other OSDs. Each OSD executes an instance of the spin-down algorithm and adapts to the overall workload. To maximize energy saving, a write request for a spun-down OSD may be redirected to another OSD, and a read request for a redirected object will be redirected to the OSD storing that object, thus increasing the spin-down duration of a spun-down OSD.

1. Introduction

Energy is an expensive resource that is often overlooked in the design of storage systems. Powering a large storage system over a long period of time can be very expensive, possibly millions [21], and may exceed the cost of the system itself. Cooling such a system also requires significant energy, and the more energy consumed, the more cooling required. Unfortunately, energy consumption is often not a priority in the design of storage systems because its cost is amortized over the lifetime of the system and not apparent at the beginning.

Within a typical system there are several subsystems that are candidates for power conservation: the CPU [20], networking [19], and hard-disk [10] subsystems all have power reduction potential. In desktop systems, the hard-disk alone can account for up to 30% of the energy consumption of the entire system [6]. The hard disk is therefore a prime candidate for energy conservation because of its large overall power consumption and because of its well-defined power states that can be exploited for energy savings.

Disk drives are electro-mechanical devices with rotating magnetic media platters and on-board electronics. The mechanical components of the drive—the rotating disk platter and the moving read-write head—consume a significant amount of energy. When not in use, energy can be saved by ceasing the rotation (*spinning down*) of the disk platter. This comes at a cost, however, as it may take several seconds and significantly more power to spin the disk back up to operational speed after it has been spun down. Thus it is only useful to spin down the disk if it will be idle long enough to more than compensate for the extra spin-up energy, and if the delay caused by spinning it back up is acceptable. Because disk request streams are generally not known in advance, heuristics are used to decide when to spin down the drives.

Spin-down algorithms for disk drives have been studied extensively. Existing spin-down algorithms can be classified into three categories: fixed timeout [5, 9], adaptive [14, 7], and predictive [13]. All share the common goal of saving energy by spinning down the platter when the device is not in use. They differ in how they decide when to spin down the platter.

Fixed timeout spin-down algorithms maintain a constant pre-set timeout value independent of the disk parameters, spinning down the platter whenever the device has received no requests during the time-out interval. Adaptive spin-down algorithms vary the timeout value based on disk request patterns and disk state power statistics. Predictive spin-down algorithms also use disk request and disk state power statistics, but use them to determine when it is beneficial to spin-down and may do so following a disk request without waiting for a timeout to expire.

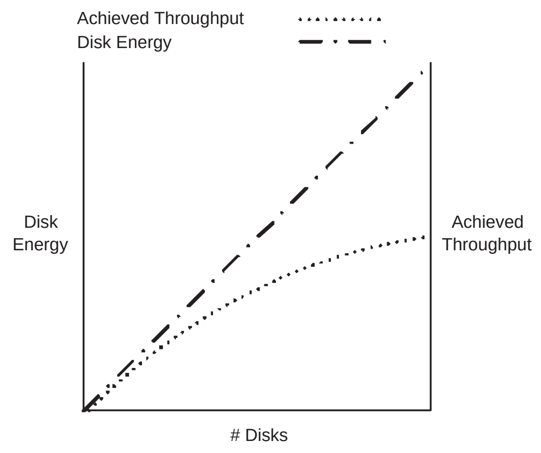

Figure 1: Achieved throughput and energy consumption vs. number of disks

We are developing *Ceph*, a petabyte scale object-based storage system with a target capacity of 20 PB and throughput of up to 1 TB/s for use in high-performance computing environments with tens of thousands of clients [23]. Ceph is built using a large number of object-based storage devices (OSDs), each of which contains one or more disk drives managed by a local CPU. In Ceph and other large-scale storage systems the energy consumption of the entire system is dominated by the disk drives. Spinning down individual drives in a traditional storage system may be exceedingly complex but, because each OSD in Ceph has a CPU, relatively complex OSD-specific algorithms can be applied to each of the disks in our system, with the potential for a correspondingly large reduction in power consumption.

With traditional spin-down algorithms energy consumption may be reduced, but without realizing the full potential for energy saving. In the worst case both energy consumption and overall storage system performance may actually be worse than without any spin-down algorithm. This is because traditional spin-down algorithms, as described above, are unsuitable in the context of Ceph. Ceph is a distributed storage system that provides load-balancing across all OSDs with the primary goal of avoiding hot-spots. Ceph's data placement algorithm [12] attempts to disperse the load as widely as possible, parallelizing data accesses as much as possible by involving as many OSDs as possible in each data access. An effect of such a load-balancing data-distribution algorithm is that each OSD may experience frequent storage accesses even when Ceph is lightly loaded. This will result in each OSD having frequent and short idle periods, defeating traditional spin-down algorithms and potentially increasing, rather than decreasing, power consumption (and overall latency), if the idle periods are just slightly longer than the timeouts employed. Too-frequent power cycling can also decrease hard drive lifetimes.

A spin-down algorithm, regardless of how it is designed, has a negative impact on performance - spinning a disk down and up is not instantaneous. Furthermore, the relationship between aggregate throughput of multiple storage devices and power consumption are not linear. Figure 1 shows the relative relationship between achievable throughput and energy consumed as a function of the number of OSDs. Energy consumption relative to the number of disks is linear because each additional disk requires a fixed amount of power. However, aggregate throughput typically reaches a limit after some point, after which adding more disks will not translate into a proportional increase in storage bandwidth. Other subsystems can become the bottleneck, such as the CPU or the network bandwidth, preventing the throughput from increasing.

It is therefore likely that the maximum achievable aggregate throughput of a storage system such as Ceph is less than the sum of the individual disks, in which case spinning down an individual storage device may have relatively minimal impact on the performance of the entire system, yet provide a fixed energy reduction. Because a spun-down disk device does not contribute to the aggregate throughput of the storage system, much like a removed device, we can think of spun-down and removed devices as the same in the context of their contribution to performance and energy consumption. As more disks are spun-down, each additional spun-down disk has a larger impact on the performance degradation of the system. The relative benefit of spinning down an individual disk will depend heavily upon the system load, and balancing energy consumption with performance degradation will be the key to successful energy reduction in a large-scale storage system.

We propose a spin-down algorithm for object-based storage devices that is dynamic relative to its workload. When the system is lightly-loaded, more disks will be spun-down, and when the system is heavily loaded, less disks will be spun-down. The distributed spin-down algorithm decision of ẅhen to spin-down?ïs based on recent read and write throughput, not a timeout value. The less read-throughput for a device, the more likely it will be spun-down. By spinning down OSD devices with the lowest read-throughput, we can reduce the number of spin-ups due to reads for an object on a spun-down OSD device. To prevent OSDs from spinning up due to writes, a spin-down redirection mechanism is used to distribute writes across the other OSD devices which are still spun-up. The spin-down algorithm also considers the number of redirected objects it is currently holding for other spun-down OSD devices.

The rest of the paper is organized as follows: Section 2 presents an overview of the Ceph system. Section 3 describes our spin-down algorithm in the OSD, Section 4 discusses redirecting object-writes for spun-down OSD storage devices, and Section 5 shows the architecture for redirected data objects. We conclude our paper in Section 7

2. Ceph Overview

Object-based storage [15] is an emerging paradigm in distributed storage architecture with the potential to achieve high capacity, throughput, reliability, availability, and scalability. The main difference between the the object-based storage model and traditional distributed storage models is that object-based storage offloads the handling of low level storage details to the storage devices themselves. Object-based storage devices export and abstract data object interface with all block-level functions such as disk space management and request scheduling handled internally. In addition, metadata management is typically decoupled from data management and, after authentication by the metadata management subsystem, clients are able to access the storage devices directly at the object level.

Figure 2 shows the three major components in Ceph: the metadata server (MDS) cluster, the clients, and the object-based storage devices (OSDs). The metadata server cluster is responsible for metadata operations such as file and directory management and authentication [24]. A client interface provides the file system API to applications on the client nodes. The OSDs encapsulate block-level storage allocation, scheduling, and management. To access data, a client first contacts the MDS cluster to authenticate itself and obtain the location of the data. The client then transfers data directly to and from the appropriate OSDs, resulting in a very high degree of data parallelism.

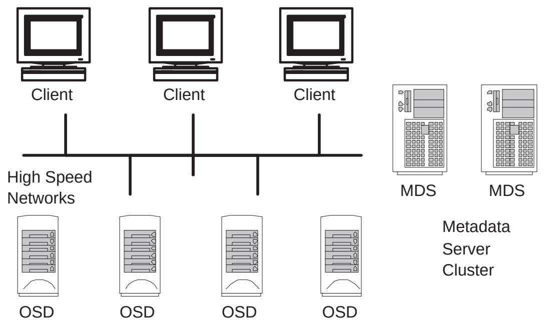

Figure 2: Ceph's Object-based Storage Architecture

The very large number of disks involved in building Ceph brings the issue of energy consumption to the forefront. Power consumption is a pertinent issue for modern day computing installations. In fact, at many sites the computing power is limited by the amount of electrical power that can be provided to the location and the difficulty of dissipating the resulting heat. Economic, environmental, and geographic issues exacerbate the problem.

The OSDs in Ceph are intelligent and autonomous storage devices that not only manage block-allocation locally, they have peer-to-peer capability that allows them to communicate with each other autonomously and perform tasks independently without the direction of a centralized control unit. This peer-to-peer capability enables the unique ability to redirect writes for energy conservation purposes. Each OSD consists of a CPU, network interface, local cache, and storage device (disk or small RAID configuration). If the disks in an OSD are powered off, the on-board intelligence can be put into a semi-hibernation state with very minimal power consumption. A wake-on-LAN capability is maintained that allows the embedded circuitry to be powered on and perform computations associated with the redirection task without having to power up the disk platters.

3. OSD Spin-down Algorithm

Traditional spin-down algorithms are not suitable for a Ceph storage system because Ceph distributes data across all OSDs. By distributing data across OSDs, hot-spots are removed from the Ceph, averaging out the idle-period for each OSD storage device. As a result, a traditional spin-down algorithm on each OSD storage device would generate sub-optimal performance and faster device wear-leveling. By leveraging the properties of a distributed storage system into the spin-down algorithm itself, a more efficient algorithm can be achieved.

In Ceph, a spin-down algorithm is run for each OSD. An OSD may have several storage devices. For a particular storage device, the decision of when to spin down is a function of it's read-throughput (of *read-ops* per *time-period*), write-throughput, and the number of redirected objects the storage device is storing. A redirected object is an object that is destined for a particular spun-down storage device on an OSD but redirected to another OSD because the original was in a low-power mode and couldn't service that object write. We discuss the redirection and it's implications later in Section 4.

The spin-down algorithm for a storage device is based on the evaluation of the following parameters:

- The read throughput index, RTI.
- The write throughput index, WTI.
- The redirect index of the OSD, RDI.

RTI, *WTI*, and *RDI* are parameters which are normalized and then averaged to compute a spin-down index, *SDI*. The parameters are normalized because reads and writes to a particular storage device are not equal—reads incur a bigger penalty since writes can be redirected to another OSD storage device[1]. If the spin-down index (*SDI*) is less than *THRESHOLD*, the storage device will be spun down. *THRESHOLD* is used to control the frequency of spin-downs. It can be fixed or dynamic, similar to the timeout variable in a traditional spin-down algorithm. An adaptive algorithm can be used to dynamically adjust *THRESHOLD* so that a defined metric such as desired % of energy reduction relative to system load can be achieved. An example algorithm is the machine learning Multiple Experts algorithm developed by Herbster and Warmuth [11]. The pseudo-code for the spin-down algorithm is shown below:

```
while(spin-down algorithm on) {
    SDI = (RTI+WTI+RDI)/3;
    if( (!spun-down) and
            (SDI < THRESHOLD) ) {
        spin-down OSD storage device;
    }
    else {
        start next sampling period;
        sleep till period's end;
    }
}
```

1 Replication, not yet fully implemented in Ceph, can offset this difference, at the cost of a slightly more complicated algorithm, by allowing reads to go to the replicas on spun-up disks when load is light

RTI and *WTI* are computed once at the end of each sampling period. The default sampling period is five minutes, but can be modified to suit a desired performance. The sampling period determines the overhead of the algorithm and how fast an OSDs storage device spin-down algorithm responds to Ceph workload changes. Decreasing the sampling period increases a storage device's response to workload changes and vice versa. The algorithm to compute the *WTI* and *WTI* are:

$RTI_n = (rtp + w \times RTI_{n-1})/(1 + w)$
and
$WTI_n = (wtp + w \times WTI_{n-1})/(1 + w)$,
where *w* controls how fast previous computations read and write throughput indices are forgotten. *rtp* and *wtp* are the number of reads and writes in the previous sampling period, respectively. The algorithm to compute the next *RTI* and *WTI* is a weighted average of the previous period's number of read or writes with the previous *RTI* or *WTI* index.

RDI is the redirect index for this OSD storage device. It is computed by:

$RDI = norm(ro) + norm(fo) + norm(uo)$,
and is a function of the number of redirected objects (*ro*) this OSD storage device is currently storing, the number of available objects (*fo*) in this OSD storage device, and the number of stored objects(*uo*) in this OSD storage device. Each parameter is also normalized for the computation of RDI. Storing redirected objects means this OSD storage device may be required to give the redirected objects back to the owner, and if this OSD storage device spins-down it will either have to either migrate those objects to another OSD storage device before spinning down or spin backup in order to give a requested redirect object back. Therefore, holding less redirected objects yields a smaller *ro*. Less free-space on an OSD storage device contributes to a smaller *fo* because fewer new objects can be created on this OSD storage device , possible resulting in less write operations. The higher the number stored objects on this OSD storage device means potentially more reads for this OSD storage device causing it to spin-up. Therefore more stored objects means a higher *uo* value.

4. Write/Read Redirection for Spun-down OSD Disks

When an OSD storage device is spun-down, the OSD is still capable of receiving read and write requests. Write redirection occurs when an OSD receives a write request and the corresponding device is spun-down. When such a request is encountered, the OSD redirects that write object to a separate OSD storage device, either on the same or different OSD. Read redirection occurs when the original object write, destined for a spun-down OSD, is redirected to another OSD. The read request must go to the the OSD storing the redirected object.

When an OSD spins down, the owning OSD creates an empty in-memory *redirect list* for that storage device. Each entry in the list refers to a redirected object. Each entry contains an ID of the redirected object and the OSD now holding that object. Each active (spun-up) OSD storage device also has a *redirect list* maintained by it's OSD, that contains the object id and the owning OSD of each redirected object.

There are three redirection functions available:

- **redirect_query(**$node$**)** - queries OSD $node$ for it's redirect index, RDI. The return value determines it's willingness to store a redirected object. A higher return value mean the OSD is more willing to redirect and a lower value indicates less willingness to redirect. Magic numbers allow a node to specify it isn't available, indicating that all the storage devices on a particular OSD being queried are spun-down.
- **redirect_store(**$node$**,**$object$**,**id**)** - is a request to OSD $node$, to store the object $object$ with id id. The OSD redirecting the store operation to OSD $node$ adds an entry to it's storage device's redirect list containing the OSD $node$ and object id, id. The OSD storing the object also creates an entry to it's storage device's redirect list containing the OSD that sent $object$ and id. redirect_store() should only be called to a particular OSD after redirect_query().
- **redirect_retrieve(**$node$**,** id**)** - is a request to OSD $node$, to retrieve the object with id id. This function returns the object with id id from OSD $node$ to the caller. After completion, both $node$ and the calling OSD remove the entry with id from their respective storage device's redirect lists.

The OSD that issues redirect_query() tries to distribute the redirect requests across all of the nodes. If an OSD has multiple storage devices, the redirection algorithm will favor sending redirection objects to local storage devices. To minimize overhead, a subset of the total OSD list is queried for each redirect operation. The OSD node query list is changed relative the OSD' responses, as well as OSDs' frequency and recency in being queried.

4.1. OSD Spin-down with Redirected Objects

It is likely that an OSD storing redirected objects will also be instructed to spin-down. There are two policies which can be enforced with regard to spinning down an OSD with stored redirected objects. They are:

- Redirected objects may exist on either spun-up or spun-down OSDs
- Redirected objects may only exist on spun-up OSDs

Each policy yields different properties. Forcing redirected objects to live on only spun-up disks means that when a disk storing redirected objects is instructed to spin-down it must migrate those objects to other OSDs through the same process calling request_query(), however it must do so for all objects it stores, which could mean a significant amount of overhead. Additionally, the disk now spinning down must notify the owner of each object it has stored of the new owner or forward any such requests that it receives. However, the benefit of such an algorithm is that any request for redirected objects will not cause the spin-up of an OSD. The overhead of moving objects to other storage devices can be mitigated if storage devices which receive the redirected objects are on the same OSD as the storage device, which has been instructed to spin-down. In any case, redirects will be common in inverse proportion to the load of the system, so the additional overhead will only be incurred when it can be accommodated.

On the other-hand, the policy that redirected objects may exist on spun-down disks means that a request for an object may result in the spin-up of an OSD storage device if the storage device storing the redirected object is spun-down. However, such a policy will result in less transfer overhead as objects do not need to be migrated when an OSD is spun down. Depending on how fast the system is writing to storage devices and spinning-down disks, the overhead may be significant, even for local OSD object transfers.

5. Redirected MetaData-Object Structure

In addition to the in-memory redirect list mentioned previously, it is also necessary for an OSD

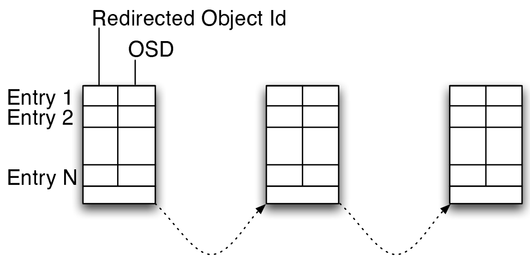

Figure 3: Redirect Descriptor Objects for Redirected Objects

to store the list of redirected blocks for each storage device on disk in the event of OSD failure. Figure 3 shows the disk structure used to represent the list of redirected blocks that a node is storing. Each entry in the meta-block contains the same information that the in-memory redirect list contains. The $RedirectedObjectID$ is the object id of the object being redirected and OSD is the OSD node that owns the object. The last entry in a meta-object redirect descriptor is a pointer to the next descriptor and points to null for the last descriptor.

There are three scenarios that can occur with regard to OSD failure: the OSD with spun-down storage devices fails, an OSD with storage devices holding redirected objects fails, or both. After an OSD fails, and it is rebooted, there several failure recovery steps it must take. First, it must query its storage devices to determine the state of the storage devices at the time of failure. For devices which were storing redirected objects, the OSD must regenerate an in-memory redirect list for that storage device. The OSD then rebuilds the list for any storage devices which were spun-down. Next, the OSD must the query the OSD which contain the storage device's data to verify that the OSD with the redirected data still contains it and didn't move the redirected objects while the failed OSD was down.

6. Related work

Persistent interest exists on energy conservation techniques for mobile computers. It is only relatively recently that energy conservation has become an issue of interest for server type systems. Recent work has proposed energy conservation techniques for internet servers used in data centers. One approach is to concentrate workloads on servers instead of distributing them evenly across all the servers in the cluster. Incoming requests are dynamically switched and directed to the minimum number of servers necessary to handle the load [2, 17, 18]. Redirecting switches are also the building block for the resource management architecture [3]. These approaches work on server clusters because all the servers in the cluster are considered homogeneous, and any of them can handle any request. Functionality specialization is the concept of assigning different functionality to different cluster nodes, so functionality that is used more often can be separated from functionality that is used less often [1]. Five policies for server power management, included IVS (independent voltage scaling), CVS (coordinated voltage scaling), VOVO (vary-on vary-off), combined VOVO-IVS, and combined VOVO-CVS were evaluated, and the result shows that the combination policy of VOVO-CVS yields the largest saving [8]. Our work is similar in that all the OSDs are autonomous units, but differs significantly in that the OSDs are dedicated storage units as opposed to computing servers.

Phoenix [16] is a real-time network storage device that implements a cycle-based disk scheduling algorithm. Data prefetching is used and when a cycle can be skipped, the disk array is put in low power mode to conserve energy. This technique is used within an individual storage device independently. Our technique consists of collaborations between all the storage nodes (OSDs) of the system. Work has also been done on load balancing for disk arrays [22, 25]. This research seeks to distribute and balance loads across disks in order to "cool down" hot disks. In our approach, we allow the tradeoff between energy conservation and load balancing to be made.

In anticipation to the introduction of disks that can spin at different speeds (and hence have different energy consumption and performance characteristics), Hibernator [26] is an energy management system that takes advantage of multi-speed disks in order to balance between energy consumption and performance goals.

The MAID (massive arrays of idle disks) [4] utilizes power management to achieve high performance while using a fraction of the power. These are traditional "dumb" disks that require outside control. The OSDs differ in that they are autonomous and intelligent devices that collaborate with each other through peer-to-peer capability in a fully decentralized manner.

7. Current Status

The Ceph storage system currently exists in prototype form that allows the implementation of the spin-down algorithms. We are planning to implement our distributed spin-down algorithm with write redirection in the OSDs to verify its benefits as well as the correctness and data consistency. We expect significant power savings to result.

A key challenge will be managing spin-down while maintaining overall system performance. Tradeoffs between spin-down, data distribution, replication, reliability, and other factors may result. Coordinating these competing requirements will be challenging, but may also provide additional power-saving opportunities. Replication, in particular, may provide very fertile ground for improving our power-saving algorithms because individual objects may reside on several OSDs, not all of which need to be spun up when load is relatively light.

References

[1] R. Bianchini. Research directions in power and energy conservation for clusters. Technical Report DCS-TR-466, Department of Computer Science, Rutgers University, November 2001.

[2] J. Chase and R. Doyle. Balance of power: Energy management for server clusters, 2001.

[3] J. Chase, R. Doyle, C. Anderson, P. Thakar, and A. Vahdat. Managing energy and server resources in hosting centres. In *Symposium on Operating Systems Principles*, pages 103–116, 2001.

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

[5] Microsoft Corp. Power management in windows xp. www.microsoft.com.

[6] Fred Douglis, P. Krishnan, and Brian Marsh. Thwarting the power-hungry disk. In *USENIX Winter*, pages 292–306, 1994.

[7] Fred Douglis, Padmanabhan Krishnan, and Brian Bershad. Adaptive disk spin-down policies for mobile computers. In *Proceedings of the second USNIX Symposium on Mobile and Location Independent Computing*, 1995.

[8] E. Elnozahy and R. Rajamony. Energy-efficient server clusters, 1994.

[9] Ricardo Galli. Cpudyn. http://mnm.uib.es/gallir/cpudyn/.

[10] David P. Helmbold, Darrell D. E. Long, and Bruce Sherrod. A dynamic disk spin-down technique for mobile computing. In *Proceedings of the 2nd annual international conference on Mobile computing and networking*, pages 130–142, 1996.

[11] Mark Herbster and Manfred K. Warmuth. Tracking the best expert. *Machine Learning*, 32(2):151–178, 1998.

[12] R. J. Honicky and Ethan L. Miller. Replication under scalable hashing: A family of algorithms for scalable decentralized data distribution. In *IPDPS04*, Santa Fe, NM, April 2004. IEEE.

[13] Chi-Hong Hwang and Allen C.-H. Wu. A predictive system shutdown method for energy saving of event-driven computation. *ACM Trans. Des. Autom. Electron. Syst.*, 5(2):226–241, 2000.

[14] P. Krishnan, P. M. Long, and J. S. Vitter. Adaptive disk spindown via optimal rent-to-buy in probabilistic environments. *Algorithmica*, 23(1):31–56, 1999.

[15] Michael Mesnier, Gregory R. Ganger, and Erik Riedel. Object-based storage. *IEEE Communications Magazine*, 41(8):84–900, August 2003.

[16] A. Neogi, A. Raniwala, and T. Chiueh. Phoenix: a low-power fault-tolerant real-time network-attached storage device. In *ACM Multimedia (1)*, pages 447–456, 1999.

[17] E. Pinheiro, R. Bianchini, E. Carrera, and T. Heath. Load balancing and unbalancing for power and performance in cluster-based systems. Technical Report DCS-TR-440, Department of Computer Science, Rutgers University, May 2001.

[18] Eduardo Pinheiro and Ricardo Bianchini. Energy conservation techniques for disk array-based servers. In *ICS '04: Proceedings of the 18th annual international conference on Supercomputing*, pages 68–78. ACM Press, 2004.

[19] Christian Poellabauer and Karsten Schwan. Energy-aware traffic shaping for wireless real-time applications. In *Proceedings of the 10th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 48–55, May 2004.

[20] Johan Pouwelse, Koen Langendoen, and Henk Sips. Dynamic voltage scaling on a low-power microprocessor. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 251–259. ACM Press, 2001.

[21] Yasushi Saito, Svend Fr&#248;lund, Alistair Veitch, Arif Merchant, and Susan Spence. Fab: building distributed enterprise disk arrays from commodity components. *SIGOPS Oper. Syst. Rev.*, 38(5):48–58, 2004.

[22] P. Scheuermann, G. Weikum, and P. Zabback. Adaptive load balancing in disk arrays. In *FODO*, pages 345–360, 1993.

[23] Feng Wang, Qin Xin, Bo Hong, Scott A. Brandt, Ethan L. Miller, Darrell D. E. Long, and Tyce T. McLarty. File system workload analysis for large scale scientific computing applications. pages 139–152, College Park, MD, April 2004.

[24] Sage A. Weil, Kristal T. Pollack, Scott A. Brandt, and Ethan L. Miller. Dynamic metadata management for petabyte-scale file systems. Pittsburgh, PA, November 2004. ACM.

[25] J. Wolf, P. Yu, and H. Shachnai. DASD dancing: A disk load balancing optimization scheme for video-on-demand computer systems. In *Proceedings of the 1995 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 157–166, 1995.

[26] Qingbo Zhu, Zhifeng Chen, Lin Tan, Yuanyuan Zhou, Kimberly Keeton, and John Wilkes. Hibernator: helping disk arrays sleep through the winter. *SIGOPS Oper. Syst. Rev.*, 39(5):177–190, 2005.