

## Energy Conservation for Real-Time Disk Systems with I/O Burstiness

Adam Roth

BigTribe Corporation  
330 Townsend St Ste 209  
San Francisco, CA 94107,  
USA  
aroth@bigtribe.com

Adam Manzanares, Kiranmai  
Bellam, Xiao Qin<sup>†</sup>, Xiaojun Ruan

Department of Computer Science and  
Software Engineering  
Auburn University, Auburn, Alabama  
36830, USA  
{acm0008, kzb0008}@eng.auburn.edu,  
xqin@auburn.edu

Mais Nijim

Computer Science, School of  
Computing  
The University of Southern  
Mississippi  
Hattiesburg, MS 39406  
Mais.nijim@usm.edu

### Abstract

*Energy conservation has become a critical problem for real-time embedded storage systems. Although a variety of approaches to reducing energy consumption has been extensively studied, energy conservation for real-time embedded storage systems is still an open problem. In this paper, we propose an energy management strategy (IBEC) using I/O burstiness for real time embedded storage systems. Our approach aims at blending the IBEC energy management strategy with low level disk scheduling mechanism to conserve energy consumed by storage systems. Extensive experiments are conducted involving a number of different load patterns as well as real-world data-intensive applications. To prove the efficiency of IBEC, we compare the performance of IBEC against three existing strategies, namely, PA-EDF, DP-EDF, and EDF. Results demonstratively show that compared with the alternative strategies, IBEC reduces the power consumption of real-time embed disks system by up to 60%.*

### 1. Introduction

Because of the advances in computational power, disk performance, and high-speed networks, storage systems have been an object of interest. Such a trend can be attributed to the rapidly growing demands of data-intensive applications, which include but not limited to video surveillance [1] and remote-sensing database systems [9]. Main components of real-time embedded include VLSI chips and hard-disk drives [4]. Recent development of magnetic disk techniques has made it possible to provide real-time embedded systems with small form-factor disk drivers that have high capacity. It is often the case that the hard-disk drives are one of the most energy-consuming components [16]. A recent study shows that storage devices account for almost 27% of the total energy consumption. Emerging fast disks with higher power needs make this problem even worse. Hence, reducing power consumption of hard disks is an efficient

approach to achieving long battery life for real-time embedded systems.

In recent years, processor energy conservation for real-time embedded systems has been extensively studied. However, energy conservation for disk storage in real-time systems remains an open problem.

In this study we aim at developing a novel energy management strategy for real-time embedded storage systems. Our approach is to blend the energy management strategy with low-level disk scheduling mechanisms to conserve energy consumed by storage systems. The proposed approach can minimize energy consumption of real-time embedded disks while making the best effort to meet timing constraints of real-time disk requests.

The rest of the paper is organized as follows. In the next section we summarize related work. Section 3 describes a model of energy management in real-time embedded disks. In section 4, we propose the energy management strategy. Section 5 presents the experimental results. Finally, Section 6 concludes the paper.

### 2. Related Work

There exist many excellent studies on storage systems, and previous techniques used to improve the performance of storage systems include caching and buffering [15], parallel file systems [10], load balancing [7], and disk striping [6]. Disk scheduling mechanisms play an equally important role in bridging the performance gap between CPUs and disks [11]. Although the shortest seek time first (SSTF) algorithm is efficient in minimizing seek times, it is starvation-bound and unfair in nature [13]. The SCAN scheduling algorithm is conducive in tackling the unfairness problem while optimizing seek times [13]. These disk scheduling algorithms are inadequate to meet the demands of disk requests with timing constraints.

Many data-intensive applications are real-time in nature in the sense that disk requests must be completed before specified deadlines. While some disk schedulers were implemented for a mixed-media data

<sup>†</sup> Corresponding Author. xqin@auburn.edu <http://www.eng.auburn.edu/~xqin> The work reported in this paper was supported by the US National Science Foundation under Grants No. CCF-0742187 and No. CNS-0713895, Auburn University under a startup grant, and the Intel Corporation under Grant No. 2005-04-070.

set, a mixture of data accessed by multimedia applications and best-effort applications [3], other disk-scheduling algorithms were proposed to provide quality of service guarantees to different classes of applications [7]. Our approach differs from the existing disk scheduling algorithms in that ours can effectively conserve energy by blending an energy management strategy with real-time disk scheduling algorithms. In other words, our strategy can be readily integrated into any existing disk scheduler to minimize energy consumption of real-time embedded storage systems.

A lot of research has been carried out in disk energy management in the context of storage servers [8][12]. Most research in disk energy management has focused on the issue of when disks should be put to sleep to reduce power consumption while achieving high performance [14]. Carrera and Bianchini developed an energy conservation technique to save energy by combining laptop disks and server disks [8]. These energy management schemes are focused on non-real-time disk, and are unable to guarantee deadlines of real-time disk requests. The major difference between our approach and the prior disk energy conservation techniques is that ours can reduce energy consumption of disks while meeting deadlines of disk requests.

Very recently, we proposed an energy-aware message scheduling algorithm (referred to as PARM) for real-time wireless networks [2]. Unfortunately for this scheduling algorithm, PARM limits its applicability to wireless networks. Therefore, PARM can not be directly applied to storage systems.

### 3. Modeling Energy Consumption

Each real-time disk request submitted a disk system specifies its timing constraint in form of a deadline, by which the request must be completed by the disk system. Note that deadlines of disk requests can be derived from real-time applications issuing the disk I/O operations. A disk request  $r$  is modelled by three parameters,  $r = (a, s, t)$ , where  $a$  is the disk address,  $s$  is the data size measured in KB, and  $d$  is the deadline.

To model disk power consumption, we use a finite state machine representation built around real performance data gleaned from an IBM DTTA 350-640 hard-disk drive [4]. Note that we are concerned with relative variance in power consumption from one algorithm to the next. Thus, it does not matter if the normalization process changes the absolute value of the result so long as the relationships that exist in the original power state model between the power consumed in one state versus the power consumed in another versus the power used when transitioning from one state to another are preserved.

Let  $R = \{r_1, r_2, \dots, r_n\}$  be a list of disk requests issued to a real-time embedded disk system. Let  $T_{on}$  and  $T_{off}$  denote the time intervals in which the disk system is active and inactive, respectively. We refer to  $T_{tr}$  as the time required to enter and exit the inactive state. The total energy consumed by the disk system can be computed as

$$\begin{aligned} E_{total} &= E_{on} + E_{off} + E_{tr} \\ &= T_{on}P_{on} + T_{off}P_{off} + T_{tr}P_{tr} \end{aligned} \quad (1)$$

where  $P_{on}$  and  $P_{off}$  are respectively the power of the disk in the active and inactive state,  $P_{tr}$  is the state transition power.

The transition time  $T_{tr}$  and power  $P_{tr}$  can be computed by the following two equations.

$$T_{tr} = N_{on,off}T_{on,off} + N_{off,on}T_{off,on}, \quad (2)$$

$$\begin{aligned} P_{tr} &= f_{on,off}P_{on,off} + f_{off,on}P_{off,on} \\ f_{on,off} &= \frac{T_{on,off}}{T_{on,off} + T_{off,on}}, f_{off,on} = \frac{T_{off,on}}{T_{on,off} + T_{off,on}}, \end{aligned} \quad (3)$$

where  $T_{on,off}$  and  $T_{off,on}$  are the required times to enter and exit the inactive state,  $P_{on,off}$  and  $P_{off,on}$  are the power values when the disk enters and exits the inactive state. In Equation (2)  $N_{on,off}$  and  $N_{off,on}$  are the number of times the disk enters and exits the inactive state. Without loss of generality, we assume that  $N_{on,off}$  and  $N_{off,on}$  are identical. Thus, Equation (2) can be simplified as

$$\begin{aligned} T_{tr} &= N_{tr}(T_{on,off} + T_{off,on}) \\ N_{tr} &= N_{on,off} = T_{off,on}. \end{aligned} \quad (4)$$

In general,  $T_{on}$  in Equation (1) is the sum of total serving and idle times (See Equation 5).

$$T_{on} = T_{serve} + T_{idle}, \quad (5)$$

where the total serving time is the sum of the serving time of each disk request. Thus, we have

$$\begin{aligned} T_{serve} &= \sum_{i=1}^n T_s(i), \\ T_s(i) &= T_{seek}(i) + T_{rot}(i) + T_{trans}(i). \end{aligned} \quad (6)$$

where  $T_{seek}$  is the amount of time spent seeking the desired cylinder,  $T_{rot}$  is the rotational delay and  $T_{trans}$  is the amount of time spent actually reading from or writing to the disk.

We can quantitatively compute the energy saved by the energy management strategy as below

$$\begin{aligned} E_{save} &= (T_{on} + T_{off} + T_{tr})P_{on} - E_{total} \\ &= (T_{on} + T_{off} + T_{tr})P_{on} - (T_{on}P_{on} + T_{off}P_{off} + T_{tr}P_{tr}) \\ &= (P_{on} - P_{off})T_{off} + (P_{on} - P_{tr})T_{tr}. \end{aligned} \quad (7)$$

#### **4. The Energy Management Strategy**

This section presents the energy management strategy using I/O burstiness for energy conservation (hereafter as IBEC). It is assumed that the overhead of IBEC is negligible when compared to the processing times of disk requests. This assumption is reasonable because recent studies show that energy management strategies spend less than 1% of computation.

The IBEC algorithm is based upon the idea that by delaying the execution of some requests it is possible to allow the disk to remain in a "sleep" state for a longer duration of time while processing larger contiguous blocks of requests when the disk is active, thus making better use of the disk when it is being fully powered. The consequences of this are twofold. First, we improve the efficiency of the disk by reducing the amount of time that it sits idle in the active state. Second, as a direct consequence of this policy we also reduce the sum total number of power state transitions that the disk will make when servicing a given request-stream.

This is important because transitioning from the "sleep" state to the "active" state is a process that is expensive both in terms of power consumed and the amount of time that it takes to complete this transition, so by reducing the number of these transitions it is possible to substantially improve the power consumption characteristics of the system. The general concept is that if a request arrives while the disk is in the sleep state, and if we determine that the request's deadline does not require immediately waking the disk, then we do not wake the disk, and instead cache the request for later process.

In fact, under the IBEC algorithm, if the disk is in the "sleep" state, we do not wake it until we determine that we must do so in order to guarantee the deadline of the waiting disk request(s). Note that it is probable that while we are waiting to wake the disk to service our waiting request other requests will continue to arrive, and that these requests are also queued until the disk is transitioned to the "active" state, at which point all waiting requests are serviced at once. This is the mechanism by which IBEC reduces the sum total number of necessary state transitions, and also how it attempts to create contiguous blocks of requests to service out of a potentially arbitrarily sporadic stream of requests. Also note that as more requests end up queued and consequently delayed, it is necessary to not just ensure that we guarantee the deadline of the first request in the waiting queue, but also to ensure that by delaying the first request we are not causing the deadlines of subsequent requests to become unrealizable. Thus it may quite often be the case that we will need to transition to the "active" state some

length of time before it is absolutely imperative to do so just to meet the deadline of the first request so that we can still hope to guarantee the deadlines of subsequent requests. This imposes a couple additional requirements on the algorithm. First, it becomes necessary to re-evaluate the threshold at which we must transition the disk to the "active" state whenever a new request is added to the waiting queue. Second, there must be a mechanism for estimating the amount of time spent in serving a given request given what we currently know about the state of the system. The first criteria is fairly trivial, the second is a bit less so, however, the estimation does not need to be exact, so long as it is always greater than or equal to the worst-case service time of the given request.

It is obviously better to be as close to exact as possible, but as long as our estimation is never less than the actual value we can generally assume that we are not causing deadlines to be missed by delaying a series of disk requests, although it is possible to posit scenarios under which it is not possible to avoid missing a deadline when applying this energy management policy. As we will see, this sort of situation actually occurs only extremely rarely in practice. A simple way to estimate the worst-case service time of a request while remaining reasonably accurate within the bounds of the system is to simply assume that the request we are estimating for will require seeking the read/write head entirely across the disk, from the first cylinder to the last, and that once it gets there we will have to wait for one full revolution of the hard-disk before the data we are interested in passes under it and reading/writing can commence. Assuming that the disk we are working with performs consistently in terms of its read and write speeds, which for the purposes of our simulation it does, we then have an estimate that is not only reasonably cheap to compute but which should also always exceed the actual service time of any real disk request without being so far off from the actual value as to render such estimation pointless.

It should also be noted that by itself, IBEC is a energy management policy, not a scheduling algorithm. It is meant to be implemented on top of or in conjunction with a separate scheduling algorithm, which should be a real-time scheduling algorithm such as EDF or LLF. As requests arrive, they are first scheduled according to whichever algorithm is being used, and then processed by the IBEC energy management policy. A high-level algorithmic description of IBEC is given in Fig. 1.

Step 7 verifies that deadlines can still be guaranteed as new requests are added to the waiting queue. Note that step 7 requires a bit of thought when actually being implemented. Also note that it may also

be desirable to allow the disk to idle for some threshold of time before sleeping it as opposed to immediately sleeping the disk as soon as there are no requests active or pending. This is due to the relatively large amount of power consumed in transitioning from the "sleep" state back to the "active" state, which creates a situation in which it could conceivably be more efficient to allow the disk to sit idle for a brief period of time in hopes that more requests will arrive. The exact threshold at which it is no longer beneficial to allow the drive to sit idle is something that is a function of the disk's power consumption characteristics as well as the relative distribution of requests in the request-stream, and as such this is something which can be computed if we know details about our request-stream in advance or if we have some way of making a best-guess based on observation of the request-stream at runtime. This however is beyond the scope of both this research and the IBEC algorithm in its current incarnation.

```

1. if there are no requests active or pending then
2.     the disk enters the inactive state;
3. else
4.     if the disk is active then
5.         dispatch a request with the earliest deadline;
6.     else /* the disk is inactive */
7.         while deadlines can be guaranteed do
8.             Cache any requests that have arrived;
9.         the disk enters the active state again;
10.        dispatch any waiting requests;
    
```

Fig. 1. The IBEC energy management strategy

## 5. Results

We evaluate in this section the performance of the IBEC algorithm using synthetically generated workloads as well as traces generated from real-world applications. To reveal performance improvements gained by our proposed algorithm, we compare IBEC with three existing scheduling algorithms, namely, EDF (Earliest Deadline First), DP-EDF (Delayed Power EDF), and PA-EDF (Power-Aware EDF). While DP-EDF and PA-EDF are two widely used power-aware scheduling algorithms, EDF is a well-known real-time scheduling scheme with no power-awareness at all. The three baseline algorithms are briefly described below. (1) EDF: The disk request with the earliest deadline is always executed first. (2) DP-EDF: Switch a disk to the "sleep" state whenever it has been idle for a certain amount of time and wake it up immediately whenever there is a request arrives. All arrived requests will be processed in an EDF order. (3) PA-EDF: Switch a disk to the "sleep" state whenever it is idle and wake it up

immediately whenever there is a request arrives. All arrived requests will be processed in an EDF order.

The first goal of the performance evaluation is to examine the overall performance improvements of IBEC over the three competitive algorithms. Second, we will investigate the performance impacts of request distribution patterns in terms of power consumption and guarantee ratio. Third, we study the performance sensitivity of the IBEC algorithm to request arrival rate. Last but not least, we validate the results from the synthetic real-time requests by running three real world real-time applications with IBEC.

Before presenting empirical results in detail, we present the simulation model as follows. Table 1 summarizes the key configuration parameters of the simulated disk scheduling system used in our experiments. The performance metrics by which we evaluate system performance include: *Power Factor (PF)* and *guarantee ratio (GR)*. *PF* indicates a normalized power consumption of an algorithm compared to the maximal power consumption consumed by the least power efficiency algorithm, whose *PF* is set to 1. Consequently, the values of *PF* fall into a range between [0, 1], where a smaller *PF* indicates a higher level of power efficiency. *GR* is measured as a fraction of total submitted requests that are found to be schedulable.

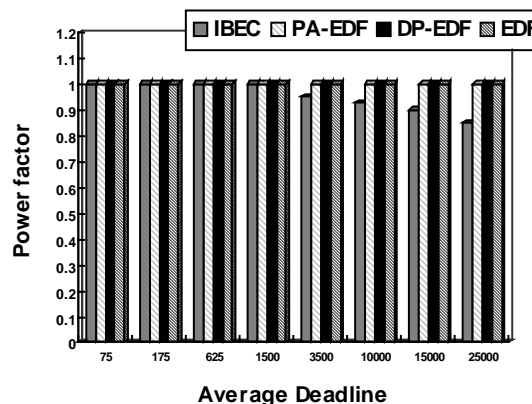


Fig. 2. Power consumption vs. deadlines.

Fig. 2 shows the power consumption of these four algorithms when average request deadline varies from 75 ms to 25000 ms. We observe from Fig. 3 that each of the four algorithms consumes the same amount of power at the maximal level when the average request deadline is less than 3500 ms. This is because the hard-disk has to be kept active all the time to service the arrival disk requests which have very tight deadlines. In other words, there is no opportunity for IBEC to

conserve some power. Therefore, IBEC gracefully degraded to existing power-aware scheduling algorithms like DP-EDF and PA-EDF. When the average request deadline is equal to or larger than 3500 ms, however, IBEC starts to conserve some energy while the three baseline algorithms remain the same performance in power consumption. We attribute the *PF* improvement of IBEC over the three baseline algorithms to the fact that IBEC judiciously employ the loose deadlines to conserve some energy. More interestingly, the improvement of IBEC over the three existing schemes in terms of *PF* is more pronounced when the deadline becomes looser for IBEC can further improve its power consumption performance when more slack time is available. On average, IBEC can save 10.8% power compared with the baseline algorithms.

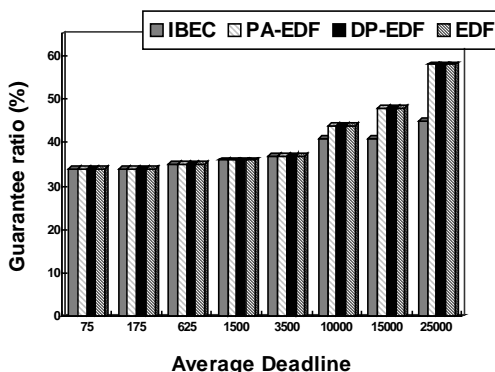


Fig. 3. Guarantee ratio vs. deadlines.

Fig. 3 plots *GR* of the four algorithms when the deadline is increased from 75 to 25000 ms. It reveals that IBEC performs exactly the same, with respect to *GR*, as all the rest approaches when the deadline is less than 1000 ms. The reason is that the relatively high workload along with the tight deadlines make IBEC only concentrate on guaranteeing arrival requests' timing constraints, which have a higher priority than power conservation requirement. However, the *GR* performance of IBEC suddenly drops off when the deadline is 10,000 ms. In fact, this is an artifact of our specific implementation of the IBEC algorithm. In order to keep simulation times manageable and to closely approximate a real system where no infinite amount of time is available to re-evaluate the schedulability of a queue, we limited the maximum number of requests that IBEC would ensure their deadline constraints. In particular, when the length of the waiting queue of requests is larger than 1,000, our implementation of IBEC will no longer guarantee the schedulability of requests after the 1,000th.

Now we focus on performance impact of the request distribution pattern. Specifically, we evaluate

the performance of the four algorithms in the cases where disk request arrival pattern follows Normal Distribution, Sparse Distribution, and Clustered Distributed, respectively. The letters on the X-axis denote the names of distribution pattern being used. For example, "N" means normal, "S" means sparse, and "C" means clustered. The numbers following the letters denote the parameters that were specified to the corresponding distribution patterns. For instance, "N, 0.5" denotes normal distribution with an arrival rate of 0.5, "S, 500" indicates sparse distribution with a sparse idle threshold of 500 ms, and "C, 25-50" indicates a clustered distribution mode with between 25 and 50 requests occurring per cluster.

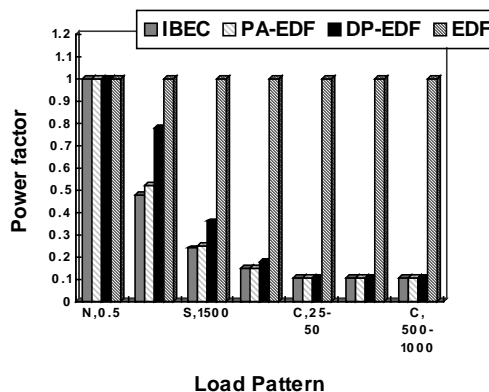


Fig. 4. Energy vs. workload distribution.

From Fig 4, we can make three important observations. First, all algorithms perform identically in power consumption under the Normal Distribution. Second, the three power-aware algorithms noticeably outperform the EDF scheme, which has no power-awareness at all, when the Sparse Distributions were applied. This is because the nature of the Sparse Distribution decides a relatively large time interval between two continuous disk requests, which in turn gives the three power-aware algorithms chances to switch the hard-disk to "sleep" mode to save energy. Furthermore, IBEC slightly outperforms DP-EDF and PA-EDF, two naive power-aware algorithms. The rationale behind this phenomenon is that IBEC can make most use of the slack time of each arrival request. Put it in another way, IBEC only wakes up the disk at the last second from which all arrived requests' deadlines can still be met, while DP-EDF only lets the disk sleep for a fixed period of time no matter whether a request is waiting for service or not. Third, for clustered workloads, IBEC and the two naive power-aware techniques perform comparably, and they both significantly perform better than EDF. This is due to

the fact that IBEC, DP-EDF, and PA-EDF can put the disk to “sleep” status completely between clusters of requests. Thus, the three power-aware algorithms can substantially save power compared with EDF. The reason why IBEC ties with DP-EDF and PA-EDF is that the performance improvement of IBEC in terms of power consumption essentially depends on slack times of arrival requests rather than the arrival patterns.

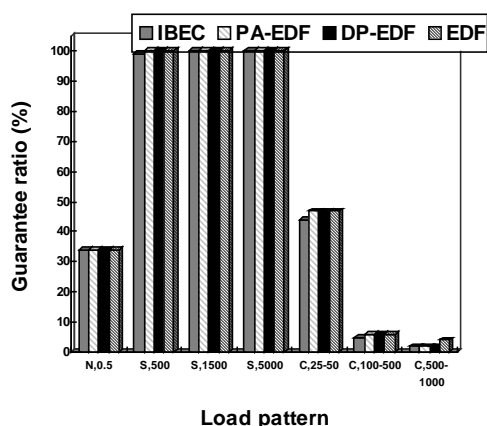


Fig. 5. Guarantee ratio vs. workload distributions.

four algorithms deliver a 100% guarantee ratio under the Sparse Distribution. The reason for this is that the average request deadline is generally much shorter than the sparse idle threshold, which means that even though IBEC aggressively slows down the processing pace of disk requests, their deadlines can still be satisfied. When we applied a Cluster Distribution pattern, the performance of the four algorithms goes down when the parameters of the Cluster Distribution increase. This is because there were a large number of requests arrived during a burst of incoming requests. Consequently, all the algorithms can only guarantee the deadlines of a small part of them.

## 7. Conclusions and Future Work

In this paper, we developed an energy management strategy (IBEC) using I/O burstiness for real time embedded storage systems. To compare the performance of the IBEC strategy against three existing approaches (PA-EDF, DP-EDF, and EDF), we conducted extensive simulation experiments using synthetically workload conditions as well as real-world data-intensive applications. The results show that compared with the three alternative strategies, IBEC substantially reduces energy consumption by up to 60% while maintaining a comparable performance in the guarantee ratio.

## References

- [1] D. Avitzour, “Novel scene calibration procedure for video surveillance systems,” *IEEE Trans. Aerospace and Elec. Sys.*, vol. 40, no. 3, pp. 1105-1110, July 2004.
- [2] M. Alghamdi, T. Xie, and X. Qin, “PARM: A Power-Aware Message Scheduling Algorithm for Real-Time Wireless Networks,” *Proc. Workshop Wireless Multimedia Networking and Perf. Modeling*, Oct. 2005.
- [3] E. Balafoutis, G. Nerjes, P. Muth, M. Paterakis, G. Weikum, P. Triantafyllou, “Clustered Scheduling Algorithms for Mixed-Media Disk Workloads in a Multimedia Server,” *J. Cluster Comp.*, no. 1, 2003.
- [4] L. Benini, A. Bogliolo, G. D. Micheli, “A Survey of Design Techniques for System-Level Dynamic Power Management,” *IEEE Trans. VLSI Sys.*, vol. 8, no. 3, pp. 299-316, June 2000.
- [5] T. Bisson and S. Brandt, “Adaptive Disk Spin-Down Algorithms in Practice,” *Proc. 3rd USENIX Conf. File and Storage Technologies*, 2004.
- [6] R. Bordawekar, R. Thakur, and A. Choudhary, “Efficient Compilation of Out-of-core Data Parallel Programs,” *Tech Report, SCCS-662*, NPAC, 1994.
- [7] J. Bruno, E. Gabber, B. Ozden, and A. Silberschatz, “Disk Scheduling Algorithms with Quality of Service Guarantees,” *Proc. IEEE Conf. Multimedia Comp. Sys.*, June 1999.
- [8] E.V. Carrera, E. Pinheiro, and R. Bianchini, “Conserving Disk Energy in Network Servers,” *Proc. Int'l Conf. Supercomputing*, June 2003.
- [9] C. Chang, B. Moon, A. Acharya, C. Shock, A. Sussman, and J. Saltz. “Titan: a High-Performance Remote-Sensing Database,” *Proc. Int'l Conf. Data Eng.*, 1997.
- [10] Y. Cho, M. Winslett, M. Subramaniam, Y. Chen, S. Kuo, and K. E. Seamons, “Exploiting Local Data in Parallel Array I/O on a Practical Network of Workstations,” *Proc. Workshop I/O Parallel and Distr. Sys.*, pp.1-13, Nov. 1997.
- [11] Jr. Coffman and M. Hofri, “Queueing Models of Secondary Storage Devices,” *Stochastic Analysis of Computer and Communication Systems*, Ed. Hideaki Takagi, North-Holland, 1990.
- [12] D. Colarelli and D. Grunwald, “Massive Arrays of Idle Disks for Storage Achieve,” *Proc. Supercomp.*, 2002.
- [13] P. J. Denning, “Effects of Scheduling on File Memory Operations,” *Proc. AFIPS Conf.*, 1967.
- [14] F. Douglis, P. Krishnan, and B. Marsh, “Thwarting the Power-Hungry Disk,” *Proc. Winter USENIX Conf.*, pp.292-306, 1994.
- [15] B. Forney, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, “Storage-Aware Caching: Revisiting Caching for Heterogeneous Storage Systems,” *Proc. Int'l Symp. File and Storage Tech.*, 2001.
- [16] P. M. Greenawalt, “Modeling Power Management for Hard Disks,” *Proc. Int'l Workshop Modeling, Analysis, and Simulation Comp. Telecom. Sys.*, pp.62-66, 1994.
- [17] S. Gurusurthi, A. Sivasubramaniam, M. Kandemir, and H. Fanke, “DRPM: Dynamic Speed Control for Power Management in Server Class Disks,” *Proc. Int'l Symp. Computer Arch.*, 2003.