Expecting the Unexpected: Adaptation for Predictive Energy Conservation

Jeffrey P. Rybczynski, Darrell D. E. Long[†]
Storage Systems Research Center
University of California, Santa Cruz

Ahmed Amer[‡]
Department of Computer Science
University of Pittsburgh

ABSTRACT

The use of access predictors to improve storage device performance has been investigated for both improving access times, as well as a means of reducing energy consumed by the disk. Such predictors also offer us an opportunity to demonstrate the benefits of an adaptive approach to handling unexpected workloads, whether they are the result of natural variation or deliberate attempts to generate a problematic workload. Such workloads can pose a threat to system availability if they result in the excessive consumption of potentially limited resources such as energy. We propose that actively reshaping a disk access workload, using a dynamically self-adjusting access predictor, allows for consistently good performance in the face of varying workloads. Specifically, we describe how our *Best Shifting* prefetching policy, by adapting to the needs of the currently observed workload, can use 15% to 35% less energy than traditional disk spin-down strategies and 5% to 10% less energy than the use of a fixed prefetching policy.

Categories and Subject Descriptors

D.4.3 [**Operating Systems**]: File Systems Management; D.4.5 [**Operating Systems**]: Reliability; D.4.6 [**Operating Systems**]: Security and Protection; H.3.m [**Information Storage and Retrieval**]: Miscellaneous—*Caching, Prefetching, Power Management, Storage*

General Terms

Algorithms, Design, Security, Performance

Keywords

Mobile Computing, Power Management, Adaptive Policies, Prediction, Prefetching, Disk Spin-down

1. INTRODUCTION

In addition to protecting storage and data from intrusion and misuse, ensuring the security of a storage device includes guaranteeing the system's continued availability in the face of differing workloads. Unforeseen workloads can be the result of unexpected changes in load or application behavior, or even the result of malicious and deliberate user activity. In the latter case, it may be possible to limit requests from problematic client systems or applications, but even with a nominally light load a workload can be harmful to the system. For example, two clients making the exact same number of requests to a disk subsystem can result in radically different energy consumption by the system. To see this, consider a set of read requests that are presented to a disk in one burst. After these requests are satisfied, and a timeout period of inactivity has passed, the disk may enter a low-power state to conserve energy and remain in that state until the next set of requests arrives. On the other hand, if the exact same number of requests arrive with an inter-arrival time that is slightly greater than the inactivity timeout of the disk, it will continuously enter a low power state, only to be returned to an active state almost immediately afterwards. This behavior leads to the consumption of excess energy at each such awakening, the disk spin-up cost expended to bring the disk back to active state, resulting in a great deal of excess energy being expended. As long as a fixed spin-down policy is employed, the system is vulnerable to encountering such unexpected problem workloads, whether they be deliberately or naturally pathological.

One method to avoid such problematic workloads is to develop fully adaptive strategies for managing the disk subsystem. By continually and dynamically adapting the disk spin-down and prefetching policies in response to the current workload, such strategies provide a system that utilizes the best policy available in the face of the current workload, regardless of how such a workload may vary. In this case we look at the problem of conserving disk energy consumption, but in doing so we are also effectively minimizing total disk motion and reducing unnecessary spin-ups and spin-downs. This in turn implies a reduction in overall mechanical wear and an increase in the reliability and availability of the disk and the system as a whole. In this manner, a policy aimed at reducing disk energy consumption and activity can mitigate the effects of malicious and pathological workloads, as well as increasing the overall longevity of hard drives.

While processors are still the main consumer of system power, it has been shown that the hard disk can use up to 30% of the total system energy [9], making the disk sub-

[†] Supported in part by the National Science Foundation under award CCR-0204358

[‡] Supported in part by the National Science Foundation under award ANI-0325353.

StorageSS'05, November 11, 2005, Fairfax, Virginia, USA.
Copyright 2005 ACM 1-59593-223-X/05/0011 ...$5.00.

system a prime candidate for energy conservation. Much research has been dedicated to conserving energy, particularly in mobile environments. We use prediction and its application to energy conservation as a means to illustrate the benefits of adaptive and self-optimizing strategies in the face of varying workloads. We contend that even in applications where prediction is the goal, such adaptive management is advantageous in handling the inevitably unpredictable and variant.

2. PREDICTORS AND DISK POWER

Disk systems use a significant amount of energy. Unlike most electronics in a computer, the disk has mechanical components. The spinning disk platters and the actuator arm require a considerable amount of energy to start operation. Powering down the disk to conserve energy is therefore only worthwhile if the disk can remain idle long enough to conserve as much as the additional energy that would be needed to spin the disk back up again. Aside from intelligently deciding whether to spin the disk down for each idle period (dynamic disk spin-down), another technique is to actively reshape the workload by prefetching data that will be requested in the future, which would result in longer periods of inactivity between such bursts. Different access prediction policies can meet with varying success for different workloads, and it is interesting to note that the most accurate predictors are not necessarily the most beneficial for conserving energy. Our *Best Shifting* policy provides an effective energy-conserving predictor, while automatically updating its prediction policy in light of the current workload.

2.1 Accurate Prediction is Not Optimal

To allow our disk spin-down policy the longest idle periods, a *perfect* predictor will need to both prefetch read data, as well as judiciously delay or accelerate write-backs of modified data back to the disk. To test our dynamic policy, and competing predictors, we evaluate the performance of such a perfect oracle for each test workload.

Simply prefetching the next *N* items (even if you are perfectly accurate) is not always the best strategy. Assume that we have an access pattern which contains items *A*, *B*, *C* and *D*, and we are trying to create long disk idle periods so we can spin the disk down. Now, assume the following access sequence: *ABABCDDDBBBB* (shown in Figure 1) and that we have a cache with a capacity of 2 items. If we assume that all items are predictable, then predicting the next items that fit in the cache will get us an access pattern as shown in Figure 1(b), with three idle periods of length three. As we can see from the sequence, if we fetch D and prefetch B towards the end of the trace (*i.e.*, delaying the eviction of D), then we can reduce this to only two idle periods, one of length three and the other of length six (Figure 1(c)). Through dynamic programming we identify the behavior of a perfect oracle, rather than relying on simply prefetching the next *N* items which, as was shown in Figure 1(b), is not an optimal strategy.

The oracle works on the principal that all accesses are in one of three categories, either they are predictable, unpredictable, or may be delayed. Predictable accesses are for data that has been requested before, and assuming a prefetching algorithm was smart enough, could be predicted and prefetched. An unpredictable disk access, whether it is a write that needs to happen immediately or just a file that has not been requested before, refers to accesses that cannot be predicted and will always result in disk activity. Accesses that can be delayed are a special case. These are accesses which can be postponed for at most a given time period. After that time period expires, if the data has yet to be written to disk or read, it becomes equivalent to an unpredictable disk access that must happen immediately. An example of an access that can be postponed is a write request stored in a write buffer and waiting for a flexible time-out before being written to disk. This allows the system to batch writes with other requests based on the current state of the disk so that they can all go to the disk at the same time, creating a busier burst period and a possibly longer idle period. Weissel *et al.* [18] have also shown that flexible write time-outs can be used to batch disk requests and save disk power.

The oracle is also optimal in terms of the spin-down policy. If the idle period is long enough to make a spin-down efficient, then this perfect oracle assumes a spin-down occurs at the start of the idle period. This allows the oracle to represent not only optimal prefetching and request batching, but also optimal spin down strategy as well. The energy value estimated for our oracle is therefore a strict lower bound on how much disk energy a given trace would require.

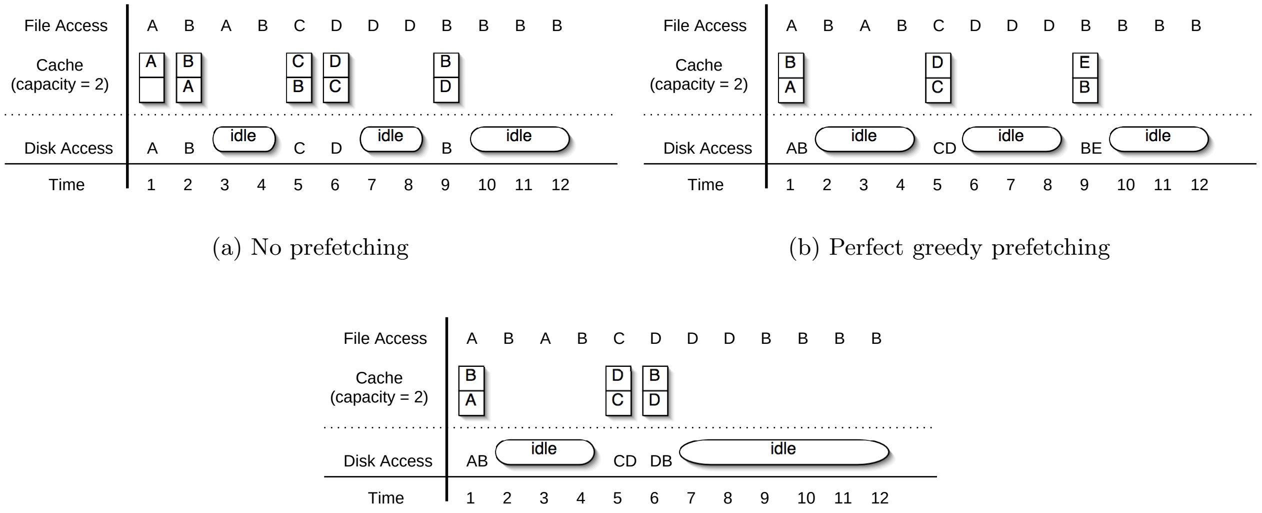
2.2 Best-Shifting Prediction

To demonstrate the effectiveness and feasibility of dynamically adapting a predictive disk power management mechanism to the current workload, we present the *Best Shifting* policy. Since we will demonstrate how different prefetching policies can perform best for different workloads, and that access patterns can vary substantially, it would be best to use an algorithm that would automatically switch to the best prefetching policy in response to the current workload. This is the basis of our *Best Shifting* prefetching policy, which adjusts which prefetching algorithm it uses based on which is likely to conserve the most energy for the current access pattern.

The *Best Shifting* policy uses machine learning techniques to choose which policy out of six implemented component algorithms works best for the workload at that point in time. The Best Shifting policy dynamically chooses the best policy, not based upon hit ratio performance, but rather, based on dynamically estimated energy savings for each component policy.

The six component predictors we evaluate in comparison to our *Best Shifting* policy, are: *Unmodified*, *Last Successor* [1, 2], *First Successor* [1, 2], *Stability* [1, 2], *FMOC* [11], and *EPCM* [12]. Each of these prefetching policies use past access events to predict future accesses. The unmodified policy simply leaves the original workload unchanged, while the Successor and Stability predictors are based on simple pair-wise associations. The FMOC and EPCM predictors are based on data compression and context modeling.

To keep track of the performance of the different predictors, each policy has its own virtual cache, containing the data it would have in the cache if it were the system's prefetching policy. The virtual cache then allows us to derive which data accesses would cause disk activity for the different policies. Each policy then stores these disk accesses in its own Disk Access Window, which is a snapshot of all the disk accesses a given policy would have created in the past *N* seconds had it been the system's prefetching policy. From this window, we can directly estimate potential energy consump-



(a) No prefetching

(b) Perfect greedy prefetching

(c) Optimal Prefetching

Figure 1: The initial request pattern with even spaces between each request and the corresponding disk accesses for: no prefetching, greedy prefetching, and optimal prefetching.

tion. Virtual caches were used previously by Ari *et al.* [3] and Gramercy *et al.* [8]. Our use of virtual caches differs in that we evaluate each policy's performance based on the estimated energy cost of using that policy, and not the simple hit ratios that it would have achieved. *Best Shifting* uses the virtual caches to determine the disk accesses each policy would have created had it been the system's policy. Then, it periodically calculates the idle durations and estimates the energy used by each policy. The policy with the lowest estimated energy usage is adopted as the policy for the cache. This selection is actually based on a relative weighting of the components, and the use of a machine-learning algorithm to dynamically adjust these weights.

When *Best Shifting*'s policy changes, there are two strategies that we can employ to realize this change. First we can simply change the policy without affecting the contents of the cache. This changes the way future predicted data will be prefetched, though it does nothing to the data already in the cache. The second strategy is to "roll over" the cache. This is the process of synchronizing the virtual cache of the winning policy with the actual cache. This operation, however, can take many disk accesses to perform, and so is only to be attempted if the disk is in the active state. If the cache policy changes while the disk is down, roll-over does not take place. If the disk is already active, then we can fetch the data we need to synchronize the cache in the background without causing an unnecessary disk spin-up. Roll-over can quickly help cache performance after the policy is switched due to a workload change that favors the new policy.

3. EXPERIMENTAL RESULTS

To test *Best Shifting*, our dynamic prefetching policy, we used file system traces from a varied selection of sources. A cache emulator is used to model the system cache while keeping track of the demand-fetched and prefetched files, along with cache statistics. If a trace entry asks for a file that is not in the cache, a disk request is created. Our cache emulator records the timing of file accesses that result in cache misses and require physical disk activity. This output is then used as the input to a spin-down algorithm, and disk energy usage can then be calculated. For our tests, we used a cache emulator with a typical 30 second write-buffer timeout. The output for this emulator was then run through a dynamic spin-down algorithm, implemented as described by Helmbold *et al.* [10].

The difference among policies was the prefetching algorithms, which were used to predict and prefetch possible future data requests. The prefetched files are then placed in the system cache, alongside normal demand fetched files. The cache then uses LRU to decide which files should be evicted. Thus prefetching incorrect files can adversely affect the performance of the cache and reduce the length of idle periods. That is why it is important to use prefetching policies that are accurate and effective. We have implemented six different prefetching policies. Each also uses an underlying LRU cache eviction algorithm while predicting and prefetching files.

Different prefetching policies can be seen to work better for different workloads, and even for the same workload observed over different days. Figures 3(a) and 3(b) show how different prefetching algorithms work better when using day long traces from instructional computers at the University of California, Berkeley [16]. Figure 2 shows the difference in performance for workloads observed on a Windows PC at the University of California, Santa Cruz during early 2005. The energy usage presented in these figures is depicted as a ratio of the estimated energy used by the given policy against the ideal energy usage of the oracle, which has the benefits of perfect prescience and perfect (instant and infallibly judicious) spin-down decisions. Both sets of traces demonstrate that a single prefetching policy does not always perform best. This is typical of practically all traces

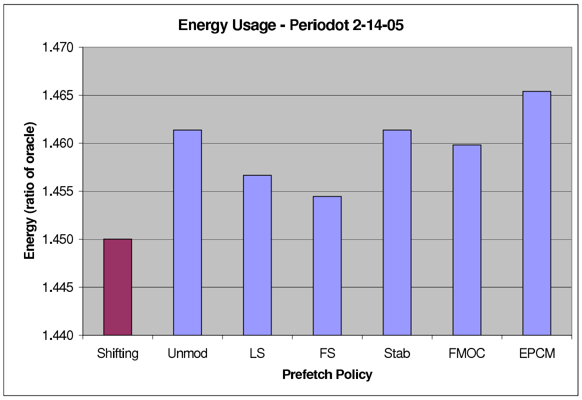
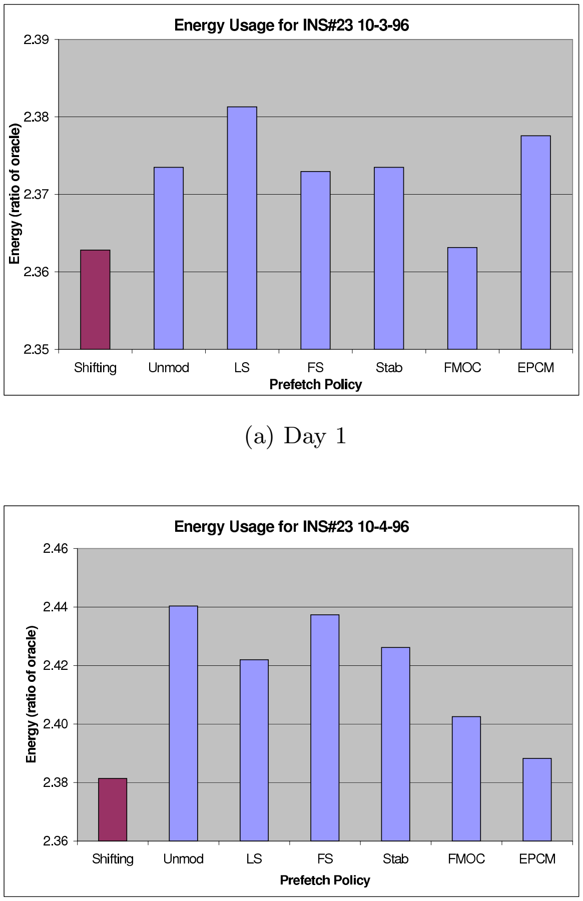


Figure 2: The energy usage for host *Periodot* at the University of California, Santa Cruz on February 14, 2005.



(a) Day 1

(b) Day 2

Figure 3: The energy usage for host *INS#23* at the University of California, Berekely on October 3 and 4, 1996.

we have observed, and suggests that it would be best to use management algorithms which automatically switch to the best policy in response to the current workload. The *Best Shifting* policy, which aims to do just that, can be seen to consistently offer the best performance compared against all other fixed and non-adaptive policies.

4. RELATED RESEARCH

Effectively spinning down the disk drive can save valuable energy in a mobile environment [13, 7]. We have used a dynamic spin-down algorithm which adjust the timeout value based on past disk request history. Helmbold *et al.* [10] described another dynamic spin-down timeout algorithm that employed a machine learning algorithm to adjust the timeout. Bisson and Brandt demonstrated the practicality of implementing such an algorithm [4]. One of the earliest proposals for the incorporation of prediction to dynamically save energy in storage systems was offered by Wilkes [19].

The nature of the access workload, and its interaction with the underlying cache and disk, is crucial for for effective disk power management. Zhu *et al.* [20] showed that simply minimizing cache misses does not necessarily result in the minimum energy usage for a given cache replacement policy. They proposed four different power-aware caching policies that can save up to 16% disk energy over a traditional LRU cache policy. Creating busier burst periods and longer idle periods allows the disk to be spun down for longer periods of time. The exploitation and promotion of such bursty behavior has been explicitly attempted by Weisel *et al.* [18], and Papathanasiou and Scott [15]. They found that traditional OS resource management policies tend to "smooth out" these burst and idle periods. *The Milly Watt Project* [5] contended that because application needs are the driving force behind power management strategies, it is useful to propagate energy efficiency information to the application. Nobel's implementation, *Odyssey* [14], showed a factor of five increase in performance over three different benchmarks. More specifically, Flinn *et al.* [6] showed that collaboration between the operation system and applications can be used to achieve longer battery life and less energy consumption. Their implementation in the Linux kernel monitored energy supply and demand to select a tradeoff between energy conservation and performance. Others have also used dynamic collaboration between the operating system and applications to increase energy efficiency and performance [18, 17].

5. CONCLUSION & FUTURE RESEARCH

Increasing system longevity, and increasing the overall reliability a system is an important goal when providing secure and available storage. Resources can be wasted, and availability threatened, by unforeseen changes in workload, as well as potentially deliberate problem workloads. One aspect of increasing availability in the face of such a threat is workload reshaping with the goal of reducing disk energy consumption. By creating increasingly bursty disk access patterns and longer disk idle periods through prediction and prefetching, such active workload reshaping can increase disk energy savings. But while some predictors are better than others, there is rarely a universal single choice of algorithm that is best for all possible workloads. This is particularly true if the nature of workload change is a result of deliberate attempts to produce a problematic, though light, workload. In such a situation, a dynamically self-optimizing algorithm, such as our *Best Shifting* prefetcher, has the potential to leverage the best strategy regardless of the encountered request patterns. This particular prefetcher, when combined with a dynamic spin-down mechanism, results in the use of 15% to 35% less energy than traditional predictive prefetching and spin-down policies.

Here we have focused solely on prefetchers and disk energy, but we aim to investigate the benefits of dynamic adaptation for other subsystems, and in the face of more deliberate adversaries. While we have shown that by dymanically selecting the prefetching strategy we can lengthen disk idle periods and save energy, our oracle results also show that there is possibly more energy yet to be saved. By implementing different prefetching strategies, with the goal not only to make more accurate predictions but also to create busier burst periods, we aim to come closer to optimal disk energy savings, regardless of the workload our algorithms may encounter.

6. ACKNOWLEDGEMENTS

We are grateful to all the members of the Storage Systems Research Center who are a constant source of stimulating comments, especially Prof. David Helmbold and to our sponsors (Engenio, Hewlett-Packard Laboratories, Hitachi Global Storage Technologies, IBM Research, Intel Research, Microsoft Research, Network Appliance and Veritas) for their financial support. We are also grateful to our colleagues at the University of Pittsburgh's Department of Computer Science, especially members of the Storage Research Group, and particularly Prof. Panos K. Chrysanthis, of the Advanced Data Management Technologies Laboratory, for valuable discussions. Jeffrey Rybczynski wishes to express his special thanks to Prof. Patrick Mantey for his support during the author's graduate studies, and for his insightful comments on this research.

7. REFERENCES

[1] Amer, A., and Long, D. D. E. Noah: Low-cost file access prediction through pairs. In *Proceedings of the 20th IEEE International Performance, Computing and Communications Conference (IPCCC '01)* (Apr. 2001), IEEE, pp. 27–33.

[2] Amer, A., Long, D. D. E., Pâris, J.-F., and Burns, R. C. File access prediction with adjustable accuracy. In *Proceedings of the International Performance Conference on Computers and Communication (IPCCC '02)* (Phoenix, Apr. 2002), IEEE.

[3] Ari, I., Amer, A., Gramacy, R., Miller, E. L., Brandt, S. A., and Long, D. D. E. ACME: adaptive caching using multiple experts. In *Proceedings in Informatics* (2002), vol. 14, Carleton Scientific, pp. 143–158.

[4] Bisson, T., and Brandt, S. A. Adaptive disk spin-down algorithms in practice. In *Proceedings of the 2004 Conference on File and Storage Technologies (FAST)* (2004).

[5] Ellis, C. S. The case for higher-level power management. In *HOTOS '99: Proceedings of the Seventh Workshop on Hot Topics in Operating Systems* (Washington, DC, USA, 1999), IEEE Computer Society, p. 162.

[6] Flinn, J., and Satyanarayanan, M. Energy-aware adaptation for mobile applications. In *Symposium on Operating Systems Principles* (1999), pp. 48–63.

[7] Golding, R., Bosch, P., Staelin, C., Sullivan, T., and Wilkes, J. Idleness is not sloth. In *Proceedings of the Winter 1995 USENIX Technical Conference* (New Orleans, LA, Jan. 1995), USENIX, pp. 201–212.

[8] Gramacy, R. B., Warmuth, M. K., Brandt, S. A., and Ari, I. Adaptive caching by refetching. In *Advances in Neural Information Processing Systems 15* (2003), MIT Press, pp. 1465–1472.

[9] Greenawalt, P. M. Modeling power management for hard disks. In *Proceedings of the 2nd International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '94)* (Durham, NC, Jan. 1994), IEEE, pp. 62–66.

[10] Helmbold, D. P., Long, D. D. E., Sconyers, T. L., and Sherrod, B. Adaptive disk spin-down for mobile computers. *ACM/Baltzer Mobile Networks and Applications (MONET) 5*, 4 (2000), 285–297.

[11] Kroeger, T. M., and Long, D. D. E. The case for efficient file access pattern modeling. In *Proceedings of the 7th IEEE Workshop on Hot Topics in Operating Systems (HotOS-VII)* (Rio Rico, Arizona, Mar. 1999), pp. 14–19.

[12] Kroeger, T. M., and Long, D. D. E. Design and implementation of a predictive file prefetching algorithm. In *Proceedings of the 2001 USENIX Annual Technical Conference* (Boston, Jan. 2001), pp. 105–118.

[13] Li, K., Kumpf, R., Horton, P., and Anderson, T. A quantitative analysis of disk drive power management in portable computers. In *Proceedings of the Winter 1994 USENIX Technical Conference* (San Francisco, CA, Jan. 1994), pp. 279–291.

[14] Noble, B. D., Satyanarayanan, M., Narayanan, D., Tilton, J. E., Flinn, J., and Walker, K. R. Agile application-aware adaptation for mobility. In *Sixteen ACM Symposium on Operating Systems Principles* (Saint Malo, France, 1997), pp. 276–287.

[15] Papathanasiou, A. E., and Scott, M. L. Energy efficient prefetching and caching. In *Proceedings of the 2004 USENIX Annual Technical Conference* (Boston, MA, June 2004), pp. 255–268.

[16] Roselli, D., and Anderson, T. E. Characteristics of file system workloads. Research report, University of California, Berkeley, June 1996.

[17] Vahdat, A., Lebeck, A., and Ellis, C. Every joule is precious: The case for revisiting operating system design for energy efficiency, Sept. 2000.

[18] Weissel, A., Beutel, B., and Bellosa, F. Cooperative I/O: A novel I/O semantics for energy-aware applications. *SIGOPS Oper. Syst. Rev. 36*, SI (2002), 117–129.

[19] Wilkes, J. Predictive power conservation. Technical Report HPL-CSP-92-5, Hewlett-Packard Laboratories, Feb. 1992.

[20] Zhu, Q., David, F. M., Devaraj, C. F., Li, Z., and Zhou, Y. Reducing energy consumption of disk storage using power-aware cache management. In *10th International Symposium on High Performance Computer Architecture (HPCA'04)* (Madrid, Spain, Feb. 2004), Cisco Systems Inc., pp. 118–129.