Context-Aware Mechanisms for Reducing Interactive Delays of Energy Management in Disks

Igor Crk   Chris Gniady
University of Arizona
{*icrk, gniady*}*@cs.arizona.edu*

Abstract

Aggressive energy conserving mechanisms can maximize energy efficiency, but often have the negative tradeoff of simultaneously reducing system responsiveness due to the switching of component power modes. This side-effect is especially prominent in hard disk drives, where the time required to switch power modes is dictated by the latency of the mechanical elements of the drive. Existing disk activity prediction schemes provide solutions for eliminating transition delays in the presence of non-interactive applications and processes, but perform poorly on systems dominated by interactive applications. The key idea in eliminating transition delays exposed to users in interactive applications is that the users are responsible for placing energy and performance demand on the systems through interactions with applications. Therefore, monitoring user interactions with applications provides an opportunity for predicting upcoming power mode transitions and, as a result, eliminating the delays associated with these transitions. In this paper, we propose a set of user behavior monitoring and prediction mechanisms that significantly reduce delays in interactive applications while minimizing energy consumption.

1 Introduction

Energy is a critical system resource for both portable and stationary systems. The need for energy efficiency in portable systems is clear: batteries have a limited energy capacity and users are expecting not only higher performance but longer battery life with every new system they buy. Recently, researchers have realized the positive financial and environmental implications of energy conservation for stand-alone servers and server clusters [3, 4, 12, 22]. The challenge of designing energy efficient systems lies in understanding the role of user interactions in energy consumption and in providing an energy/performance schedule that accommodates user demand. Furthermore, by understanding user behavior we can optimize system performance by tailoring it to a user's patterns of interaction.

Performance and energy consumption are tightly coupled where higher performance is usually achieved at the cost of increased power demand. Likewise, decreasing energy consumption by decreasing the performance level of a component can significantly increase interactive delays. This is particularly apparent in the case of hard disk drives, where the retrieval of data from a spun-down disk results in a significant delay when platters are spun up to operational speed and during which the system may become unresponsive. Keeping the disk spinning and ready to serve requests eliminates interactive delays, but wastes energy. Stopping or slowing the rotation of disk platters during periods of idleness, i.e. periods during which I/O requests are absent, is the most effective means of reducing the energy consumed by a hard drive. While prior research has focused on predicting the upcoming idle periods in order to place the disk in a lower power mode. Little has been done in predicting the arrival of I/O activity, especially in the arena of interactive user applications, where user-generated I/O requests alone do not generally exhibit discernible patterns.

Timeliness of power mode transitions affects not only the system's performance and the overall system's energy consumption but also user perception of the system's responsiveness. Significant delays are associated with the transition to a higher performance state. For example, waiting for I/O requests to arrive before switching to a higher power level may degrade system performance, keep the system processing the task longer and as result increase overall energy consumption. Switching too early wastes energy, since the demand for high performance is not present. Therefore, timely transitions to the appropriate performance level are critical for achieving both best performance and energy efficiency. Monitoring user behavior provides not only the necessary context of execution that was previously unavailable to the

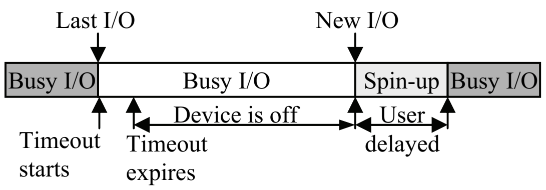


Figure 1: Anatomy of an idle period.

predictors, but enables timely predictions before the need for high performance arrives [8, 1].

In this paper, we show that user interactions can be easily monitored and exploited to increase both the timeliness and accuracy of prediction mechanisms. More specifically, we propose and apply the Interaction-Aware Spin-up Predictor (IASP) to reducing the interactive delays of hard disk power management. We propose a set of mechanisms for capturing user actions and predicting the upcoming device state, and provide a detailed design and implementation. The proposed mechanisms gather contextual information from user's mouse interactions within a GUI and use it in predicting an upcoming I/O request. The idea is motivated by the observation that with a majority of common interactive applications, the user fully interacts with the application through its graphical user interface (GUI). In this context, a simple action such as opening a file requires a sequence of mouse events. By correlating the sequence of steps to the resulting I/O we can predict future I/O occurrences when the user initiates the same set of operations again.

In this paper, we make following set of contributions: (1) we are the first to apply interaction aware prediction to spin up a hard drive, (2) we are able to successfully apply our mechanisms to predicting disk spin-ups in interactive applications, (3) we design, implement, and evaluate our design, showing significant improvements in delays exposed to the user, (4) we extend ALT mechanisms to predict the length of idle periods and spin-up the disk accordingly. Furthermore, the interaction-aware approach can easily be extended to manage other system resources and peripheral devices whose activity is dependant on user behavior.

2 Background

High performance hard drives are a significant source of energy consumption and timeout mechanisms have gained wide popularity due to the simplicity of implementation and the energy savings they provide to disks that would otherwise be spinning needlessly. Figure 1 shows an example of a timeout mechanism shutting down the device once a timer expires. The disk remains powered down until a new I/O request arrives and the disk has to be powered up before servicing the new request, potentially exposing several seconds of delay to the users.

2.1 Shutdown prediction techniques

Interestingly, spinning down the disk is not always beneficial. Accelerating the platters requires more energy than keeping them spinning while the disk is idle. Therefore, the time during which the device is off has to be long enough to offset the extra energy needed for the shutdown and spin-up sequence. This time is commonly referred to as the *breakeven-time*, and is usually on the order of a few seconds. Eliminating wrong shutdowns that not only waste energy but also significantly delay user requests is critical to conserving energy and reducing interactive delays. Simple timeout-based mechanisms gained wide popularity but they waste energy while waiting for a timeout to expire. As a result, various selections and dynamic adjustments of the timeout value have been proposed [18, 10, 14, 16] to reduce the amount of energy consumed during the timeout period. Consequently, dynamic predictors that shut down the device much earlier than the timeout mechanisms have been proposed to address energy consumption of the timeout period [6, 17, 27]. Stochastic modeling techniques have also been applied to model the idle periods in applications and shut down the disk based on the resulting models [2, 5, 23, 26].

To improve accuracy, energy management can be delegated to programmers, since they have a better idea of what the application, and potentially the users, are doing at a given time [11, 15, 20, 29]. To reduce the burden of hint insertion on the programmer, automatic generation of application hints was proposed [13] to exploit the observation that I/O activity is caused by unique call sites within applications. Finally, operating systems can concurrently evaluate multiple predictors and select the best one for the current workload [28].

2.2 Reducing spin-up delays

The goal of shutdown mechanisms powering down the disk is to improve energy savings. However, every shutdown requires a corresponding spin-up to serve future requests. It is important to note that even correct shutdowns can expose spin-up delays to the application or the user as shown in Figure 1. There are two approaches for reducing the impact of spin-up delays. First, we can prefetch and cache the data either in main memory [19, 24, 7] or an alternate storage device such as flash memory [21, 25], however disk accesses for uncached data will inevitably occur. Second, we propose waking up the disk early by spinning up the platters before the request arrives and serving the request without any delays. Both approaches are complementary since the disk will have to be spun-up at some point even if the caching techniques are very efficient.

2.3 Predicting spin-up time

In this paper, we focus on spin-up prediction, which can be achieved in two ways. First, we can predict the length of the idle period and spin-up before the end of the predicted period. Second, we can predict spin-up itself by observing system events, such as user interactions. Prediction of the idle period lengths was previously proposed by Adaptive Learning Tree (ALT) [6]. The ALT approach is to predict the best current power mode based on a sequence of idle periods. Idle periods are discretized according to the time spent idling, and in relation to the number of available sleep states and device specifications. Previously observed states or sequences of states are encoded in a tree, the paths of which are matched according to newly observed sequences of discretized idle periods. Each leaf node in the tree constitutes a prediction and the most likely prediction is selected to transition the disk to the matching power state. ALT has shown significant improvement for power mode prediction in static, non-interactive applications and motivated us to adapt the design to predict the length of idle times and spin-up the disk before the predicted idle time ends.

ALT's discretization of idle periods depends on the number of available power states of the disk, and the prediction of the period lengths allows transition into shallower sleep states, where the disk's RPMs are reduced, but not halted, and the time to ready the disk is lessened. These periods are on the order of seconds and correspond to the breakeven time of each state. We can extend the design of ALT to predict longer idle periods and for the purpose of differentiation we will refer to our modified design as ALT+. The discretization of idle periods in ALT+ results not in the prediction for the best power mode, but rather for the duration of the current idle period. In order to spin up the disk in ALT+, we consider thresholds to be multiples of the disk's breakeven time, where the multiple is given by the discretization and encoded in the same tree structure as found in ALT. With this modification, ALT+ generates a likely time to ready the disk in anticipation of upcoming I/O activity, allowing the disk to be spun up on time to service the request.

2.4 Challenges in predicting spin-up time

Accurately spinning up the disk is challenging since the idle period has to be predicted very accurately. There are three possible situations that can occur following the prediction. The first and best scenario is when the prediction is accurate and the disk is powered up just before the request arrives. In this situation, there is no energy wasted in waiting for the request to arrive and also the spin-up delay is hidden from the user. The second scenario occurs when the predicted idle period is longer than actual idle period. In this case, the device is powered-up upon I/O arrival and the latency of the spin-up is exposed to the user. The last scenario occurs when the predicted idle period is much shorter than the actual idle period. In this case, the disk is powered up and subsequently shut down without serving any disk requests. The disk is shut down to prevent it from remaining in the powered-up state for long idle periods. As a result, energy is wasted performing the unnecessary spin-up and shutdown transitions and spin-up delay is exposed when I/O requests do arrive.

Predicting the exact length of idle periods in interactive applications is difficult since it depends on the constantly changing frequency of user interactions with the application. Therefore, we propose to observe user interactions and infer from them the impending arrival of I/O activity, since users are responsible for the majority of I/O activity in interactive applications. Our mechanisms reconstruct the user's interaction context from mouse events directed at the application's GUI, thereby providing the necessary hints transparently and without application modification. The captured user context results in high accuracy and prediction timeliness in the proposed IASP.

3 Design

The key idea in IASP design is that the correlation between user interactions and disk I/O activity can be transparently exploited to predict I/O activity ahead of time and perform a timely disk spin-up to serve the request. Subsequently, our design faces several requirements:

- User interactions have to be captured transparently without modification of applications.
- Capture and prediction should be efficient to prevent excessive energy consumption by the CPU to train and generate predictions.
- The system should handle multiple applications in a graphically rich environment.
- User behavior correlation and classifications should be performed online and without direct user involvement.

The first three items are addressed by a novel implementation we propose in this paper. The last item is addressed by the proposed predictor design.

3.1 The Naïve Predictor

The observation that user interactions are responsible for the majority of disk I/O, in the interactive applications, leads us to a proposal of a simple mechanism that spins up the disk upon mouse clicks. The intuition dictates that if the user actively interacts with an application, which may require disk I/O, the disk should stay on to satisfy user requests. If the user is not actively interacting with the application the likelihood that the disk will be needed drops and the disk can be shut down. Therefore, our naïve All-Click Spin-Up mechanism (ACSU) spins the disk up upon each mouse click and keeps it spinning as long as the user is interacting with the application. Once the user stops interacting, the disk shuts down after a timeout period.

ACSU mechanisms act on all mouse clicks and spin up the disk as soon as possible, with the downside of unnecessary spin-ups for clicks that are not followed by any disk I/O. It is important to note that user interactions that require disk I/O are a small subset of all user interactions with the application. Therefore, ACSU mechanisms have the greatest potential for reducing spin-up delays at the expense of energy consumption caused by unnecessarily spinning the disk up and keeping the disk spun up without serving any disk I/Os. ACSU mechanisms set a lower bound on the spin-up delays for the proposed IASP and also illustrate the need for more intelligent prediction schemes that decide when the disk should be spun up to improve energy efficiency.

3.2 Capturing user interactions

The basis for the energy efficient design of IASP is the accurate and detailed monitoring of user activity. Most interactive applications are driven by simple point-and-click interactions. All operating systems targeted for consumers offer a Graphical User Interface (GUI) to facilitate uniform interfaces for interactions between users and application. As a result, virtually all interactions can be accomplished through mouse clicks [9]. Users interact with an application to accomplish specific tasks, like opening or saving a file. Many tasks can be accomplished by a single point and click, but other tasks require sequences of interactions. For example, to save a new file, the user commonly clicks on the File menu then selects the Save option and is presented by a directory selection menu, once the filename is entered and the user clicks OK, the file is saved and disk I/O may be requested. We argue that all GUI interactions resulting in disk I/O activity can be accurately captured and correlated to the activity they initiate, however, it should be noted that this does not always account for all occurring I/O activity.

User activity can be captured at various levels of detail. The simplest method is to capture the coordinates of mouse clicks relative to the application window and approximate the graphical interface features from clusters of clicks [1]. However, this method either requires off-line processing or complicated on-line mechanisms for clustering of incoming clicks. Clustering is necessary since the only available information about user interactions are the relative coordinates of mouse events. K-means clustering is an effective approach, but suffers from several inefficiencies. First, the number of clusters has to be known a priori, meaning that for each application, we are faced with pre-determining the number of interactive elements. Second, assigning newly observed clicks to their respective clusters has a high processing overhead. Finally, each time the layout of the window changes, the mechanisms will generate mispredictions and also require retraining. This method is clearly far from the goals we set above, since it is neither very transparent nor computationally efficient. In order to address these problems, we turn to monitoring the GUI protocol streams and directly identifying the interactive elements.

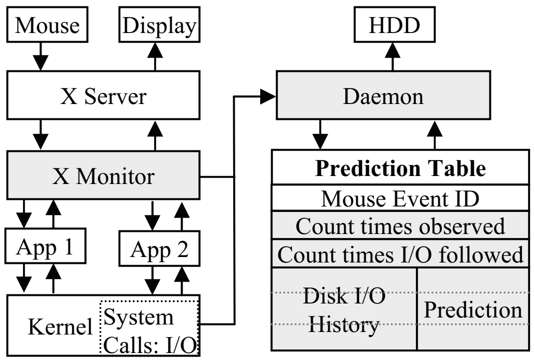


Figure 2: IASP architecture.

On Unix-like systems, the X Window System is the common display protocol built on the client-server model. It is responsible for accepting graphical output requests from and reporting user input to clients. The stream of data from the client to the server contains the information about the window layout, while the data sent from the server to the client applications contains the information about user interactions. By adding an intermediary layer, as shown in Figure 2, between the server and its clients, we can observe the exact sequence of requests and events. This layer allows for transparent monitoring of user behavior. No modification of applications is necessary. Furthermore, user interactions are captured exactly, eliminating both the excessive computational overhead of computing a clustering and the inaccuracies associated with the clustering present in the previously described solution [1]. Since the need for cluster formation and behavior detection is eliminated, the offline processing needed in the clustering approach is eliminated, fully allowing for detection, correlation, and prediction to be performed online.

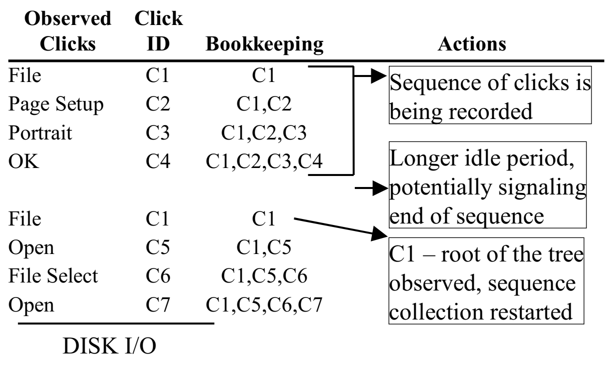
The X Window System tracks GUI windows in trees, whose structure remains the same across executions of an application. Mouse event IDs are generated by storing and traversing the trees, generating a single unique integer for each node. The IDs generated by tree traversals are augmented with the size and location of the visual element that the mouse event occurs in. Information regarding the application's window tree structure is obtained from the X Window server. Internally, the server tracks window nodes and their children with identifiers that are unique to the application during an execution. Upon subsequent executions these server-side identifiers may change, but the structure of the tree remains the same. By performing a tree traversal and labeling all visited nodes in turn, we generate IDs based on a structural representation of the tree. This allows us to reuse the training across multiple executions of the application.

3.3 Monitoring & correlating I/O activity

Our mechanisms monitor each application individually for mouse clicks and file I/O. This allows a more accurate correlation of file I/O activity to user interactions with an application. We use two levels of correlations. First, the application's file I/O activity is captured by the kernel in the modified I/O system call functions that check for file I/Os. For example, we modified *sys_read* to check if the I/O call that entered *sys_read* is indeed file I/O since *sys_read* can be used for many types of I/O. This stage does not consider buffer cache effects since file I/O activity is captured before the buffer cache. As a result, we obtain a more accurate correlation between file I/O and mouse interactions. Second, once potential file I/O activity is detected, we follow the call to see if it resulted in an actual disk I/O or it was satisfied by the buffer cache. We use this information to correlate the user interactions that invoke file I/O to the actual disk I/O. We argue that the usage patterns in the buffer cache will also correlate to the user interactions, since user behavior is repetitive, and we show that IASP is able to predict actual disk I/Os with a high degree of accuracy.

3.3.1 Correlating file I/O activity

We record correlation statistics in the prediction table that is organized as a hash table indexed by the hash calculated using the mouse event IDs. Figure 2 shows the prediction table organization and the content of the table entry that is maintained by the IASP daemon. The click IDs are unique to the window organization and therefore do not result in aliasing between different applications



Observed Clicks	Click ID	Bookkeeping
File	C1	C1
Page Setup	C2	C1,C2
Portrait	C3	C1,C2,C3
OK	C4	C1,C2,C3,C4
File	C1	C1
Open	C5	C1,C5
File Select	C6	C1,C5,C6
Open	C7	C1,C5,C6,C7

Actions
Sequence of clicks is being recorded
Longer idle period, potentially signaling end of sequence
C1 – root of the tree observed, sequence collection restarted
DISK I/O

Figure 3: Example of interaction sequences.

and windows as explained earlier. The data stored by the prediction table contains only the unique event ID, the number of times the event was observed, and the number of times I/O activity followed. The counts are a simple, but efficient means of computing an empirical probability for future predictions. The table resides globally in a daemon and is shared among processes to allow table reuse across multiple or concurrent executions of the application. Furthermore, the table can be easily retained in the kernel across multiple executions of the application due to its small size.

In addition to the global prediction table, IASP records the history of recent click activity for each process in the system. Consider a typical usage scenario shown in Figure 3 where a user is editing a file in a word processor. After a while, the user clicks through a file menu to change properties of the edited file. The recorded history of clicks is C1, C2, C3, C4. At this point, the user decides to work on the file again. If the time is long enough we can consider the clicks to be uncorrelated and the history of clicks is cleared. Alternatively, the user may immediately proceed to open a new file with click sequence of C1, C5, C6, C7. In this case, the history is also reset when the user clicks on C1. Since all menus are organized as trees in the application, clicking on C1 signifies return to the root of the File menu tree. Therefore, when IASP detect a repeated click ID in the history, the history is restarted with the current click. It is still possible to record uncorrelated clicks in the history. For example, user interacts with the Edit menu and subsequently opens a file. In this case, the history will contain clicks for edit menu interactions and the file open interactions. However, the uncorrelated clicks will have low probability and will eventually be made insignificant to the predictor with further training.

IASP uses very simple training where the observed count is updated every time a particular click is detected. In order to correlate file I/O activity, the history of clicks that lead to file I/O is traversed and the I/O count for every click present in the history is incremented. Ratio of both counts gives us the probability of file I/O following the particular click.

3.3.2 Correlating disk I/O activity

File I/Os issued by the application can be satisfied by the buffer cache and as a result may not require any disk I/O and the disk can remain in a power saving state. Since not all file I/Os result in disk I/O, we introduce an additional correlation step to correlate the mouse click to the disk I/O. We use a history of file I/Os generated by the particular click IDs to predict the future disk I/O generated by the particular click. We add a 2-bit history table with a 2-bit saturating counter to record the history of file I/Os that resulted in disk I/O after a given mouse click was observed. We update the prediction table using the history of file I/Os and the resulting disk I/O and the current outcome of the file I/O. Combination of the file I/O probability and the resulting I/O prediction results in a final decision about disk spin-up.

We are relying on file I/O prediction and disk I/O prediction to separate application behavior from the file cache behavior. By considering both the probability of a particular click being followed by file I/O and the behavior of the resulting I/O in the buffer cache, IASP can accurately discern whether that click will result in actual disk I/O. Separating the predictor's training into file I/O and buffer cache behavior allows accurate correlation of clicks to the application's file I/O, which is the fundamental goal of this paper. Mouse interactions with the application's GUI are strongly correlated to file I/O, so, intuitively, the goal of the described implementation is to filter all uncorrelated clicks first before the buffer cache impact on disk I/O is considered. Finally, we use a simple 2-bit history to predict buffer cache behavior, which provided sufficient accuracy. However, more sophisticated buffer-cache behavior prediction can be potentially employed to further improve IASP accuracies.

3.4 To predict or not to predict

The critical issues that we are addressing in our design are timeliness and accuracy, which turn out to be competing optimizations. Many application functions can be invoked with just a single click, however certain operation may require several steps. In case of multiple clicks, the last click initiates a system action that is a response to the user's interaction. More specifically, we can observe only the last click just before disk I/O occurred and correlate the click to the particular disk behavior with high accuracy. While this approach is very accurate, it is not very timely. Correlating disk I/O to the last click occurring before the I/O request was observed does not

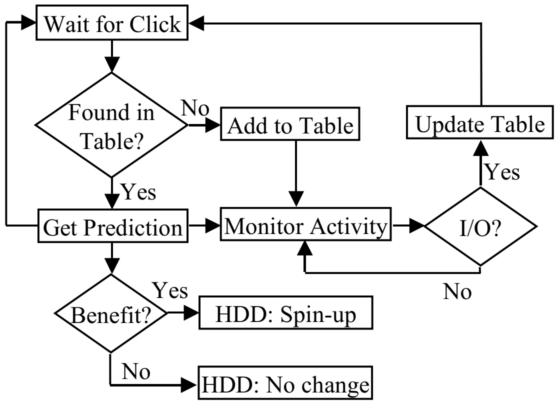


Figure 4: IASP decision flowchart.

provide adequate time before the I/O arrives to offset a significant portion of the spin-up latency, and so has a negligible impact on reducing the associated interactive delays. This scenario is illustrated in Figure 3. Clicking on C7, which is the final Open button in file open sequence, will result in an I/O system call. However, the click is immediately followed by I/O and waiting for prediction until last click will provide little benefit in reducing delays exposed to users. Spinning up the disk upon C1 click provides sufficient time to reduce delays; however, it may result in erroneous spin-ups, since the user may perform other operations that do not lead to file I/O.

3.5 Predicting upcoming activity

Figure 4 shows the decision-making process. Upon each mouse click, we use the ID of the current click to calculate the prediction table hash index. The daemon performs a prediction table lookup with three possible outcomes: the entry is found indicating that the interaction leads to file I/O with probability above the threshold, the corresponding entry is found but contains a low probability of upcoming file I/O, or the entry is not found. Events which are not found in the table are added and updated accordingly to our training routine described earlier. Once the entry is found that satisfies the desired probability threshold, we use the current history of disk I/Os for the given mouse click and perform final lookup into history prediction table for the selected click. The prediction from the history table dictates the outcome of the disk state and the disk is transition to the predicted state.

In our experimental implementation, we consider only two possible states for the disk, standby and sleep. Therefore, the decision to spin-up the disk is binary, i.e. the determination is made that either I/O activity is forthcoming following the mouse event or it is not. However, our binary decision predictor can easily be ex-

Appl.	Number of I/O Periods	Read (MB) *Without* Cache	Write (MB) *Without* Cache	Read (MB) *With* Cache	Write (MB) *With* Cache	Number of Clicks	Number of IDs
Firefox	814	1903.35	350.8	851.91	120.39	3857	130
Writer	1385	2043.62	2186.28	1434.05	2120.47	5755	195
Impress	1485	1230.42	263.6	517.06	60.44	25375	194
Calc	2846	1840.4	116.7	1280.7	59.67	9102	35
Gimp	844	1443.32	957.3	796.9	936.54	8465	157
Dia	6362	174.31	65.3	123.64	10.28	46864	118

Table 1: Applications and execution details

tended to devices with multiple sleep states. The only required modification is the discretization of the probabilities computed from the prediction table. Consider the case of 2 sleep states, one having the platters fully spun-down, and the other having the platters spinning at a reduced RPM. The third possible state is full idle. This scenario is easily handled by adding a second threshold. With two thresholds, probability values falling beneath the lower one generate a prediction favoring the halted platters sleep state. Probability values falling between the two thresholds would cause the disk to enter the reduced RPM sleep state. Finally, any probability values above the higher threshold would fully spin-up the disk. Clearly this modification can be extended to an arbitrary number of sleep states.

3.6 Multiprogramming environment

As described, our mechanisms work in a multiprocess environment since training and prediction are made independently of other processes. The described monitoring mechanism allows us to uniquely identify windows from multiple processes and allow for accurate correlation without any aliasing from other applications. The prediction performed by IASP is also easily integrated into a multiprocess environment since as soon as IASP predict spin-up for a single process the disk is spun up without considering other processes. This is opposite of the shutdown mechanisms which have to consider other processes that are currently ruining and may need the disk. In case of spin-up, once the disk is needed it has to be spun-up.

4 Methodology

We evaluate the performance of ACSU and IASP mechanisms, comparing them to ALT+. In order to fully evaluate the effectiveness of our proposed mechanisms, we use a trace-based simulator as well as an implementation of the mechanisms that replays the traces in real time with an actual disk. We focus on predicting spin-ups

State	**WD2500JD**	**40GNX**
Read/Write Power	10.6W	2.5W
Seek Power	13.25W	2.6W
Idle Power	10W	1.3W
Standby Power	1.8W	0.25W
Spin-up Energy	148.5J	17.1J
Shutdown Energy	6.4J	1.08J
State Transition		
Spin-up time	9 sec.	4.5 sec.
Shutdown time	4 sec.	0.35 sec

Table 2: Disk energy consumption specifications.

and as a result, we use a simple timeout based shutdown mechanism with the timeout set to 20 seconds which is comparable to the breakeven time of both disks. This means that the disk is shut down after 20 seconds of idleness. This also applies to erroneous spin-ups, where when the disk is spun up, it waits for the timeout to expire before subsequently shutting down.

Detailed traces of user-interactive sessions for each application were obtained by a modified `strace` utility over a number of days. The modified `strace` utility allows us to obtain the PID, access type, time, file descriptor, as well as the amount of data that is fetched for each I/O operation. The specifications of the simulated disks belong to Western Digital Caviar WD2500JD and Hitachi Travelstar 40GNX hard drives and are shown in Table 2. The WD2500JD has a spin-up time of about 9 seconds from the sleep state, the surprising duration of which appears to be remarkably common for high-speed commodity drives. The 40GNX is designed for portable systems and as such has much lower energy consumption and spin-up time than the WD2500JD.

Table 1 shows six popular desktop applications chosen for our evaluation: *Firefox*, *Writer*, *Impress*, *Calc*, *Gimp*, and *Dia*. *Firefox* is a web browser with which a user spends time reading page content and following links. In this case, I/O behavior depends on the content of the page and user behavior. *Impress* (presentation editor), *Writer* (word processor), and *Calc* (spreadsheet editor),

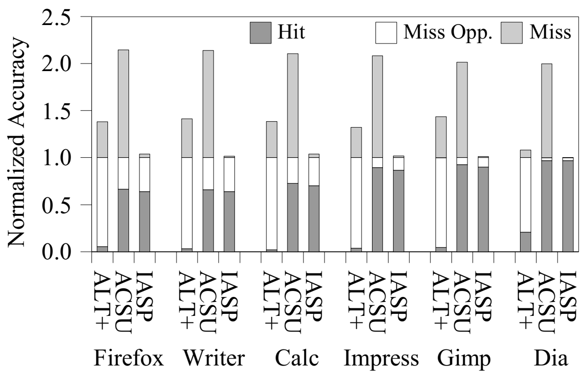


Figure 5: Prediction accuracy normalized to the total number of disk spin-ups *without* the buffer cache.

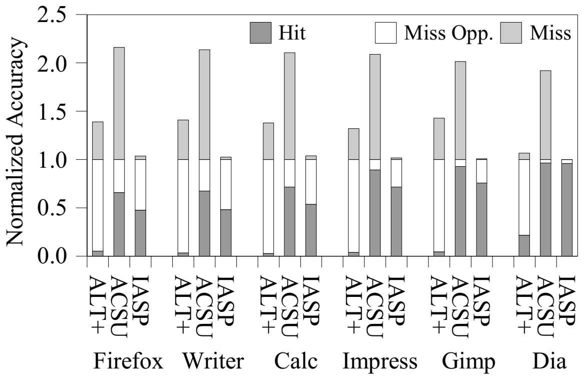


Figure 6: Prediction accuracy normalized to the total number of spin-ups *with* the buffer cache.

are part of the Open Office suite of applications. All three are interactive applications with both user driven I/O and periodic automated I/O, i.e. autosaves. *Gimp* is an image manipulation program used to prepare and edit figures, graphs, and photos. Finally, *Dia* is an application used for drawing diagrams for papers and presentations.

Table 1 also lists the total number of idle periods for which a potential shutdown and a corresponding spin-up are required, the total amount of read and write activity, a total number of mouse click interactions and the number of unique click interactions encountered in the studied applications. We show statistics for I/O requests that are generated by the application (shown as '*Without* Cache') and I/O requests that are not filtered by the buffer cache and are send to the disk (shown as '*With* Cache'). In our experiments, we use an LRU managed buffer cache of size 512MB, which is representative of current systems' capabilities.

5 Results

5.1 File I/O Correlation Accuracy

Accurate prediction ensures that the disk is not spun-up needlessly, when no activity is forthcoming. We first consider the accuracy of correlating mouse clicks to file I/O at the application level, before it is filtered by the buffer cache. Figure 5 shows the breakdown of correct and incorrect spin-ups, i.e. hits and misses, for ACSU, IASP and ALT+ that result from predicting file I/O when the system does not employ buffer caching. Hits are counted when the prediction to spin-up the disk is made and it is followed by file I/O. Misses are those spin-ups which were not followed by any I/O, and Missed Opportunities are periods for which the mechanism failed to provide a prediction, but a spin-up was needed. Each missed opportunity results in the disk being spun up on demand, essentially spinning up when an I/O request arrives. ACSU mechanisms keep the disk powered up while a user is interacting with the application, minimizing the interactive delays. While it provides an upper bound for the number of I/O periods that may be predicted by clicks (i.e. the number of periods covered by IASP can equal but not exceed the number of periods covered by ACSU), it naïvely spins-up the disk for all clicks, resulting in excess misses. Low coverage and high inaccuracies in ALT+ illustrate the behavior of the mechanisms that solely rely on observing system events without considering user interactions.

ACSU on average covers 81% of all file I/O periods, while IASP correctly covers an average of 79% of periods. The lack of contextual information and the random nature of idle period duration results in ALT+ correctly covering an average of 7% of periods. ACSU shows the greatest number of misses for all applications, 52% of spin-ups are misses. This miss rate reflects the number of existing mouse clicks that do not correlate to any I/O. When the disk is spun-up in ACSU, it will remain spinning as long as new clicks are observed and the idle threshold is not reached between any two clicks. IASP consistently results in the fewest misses, averaging 2%, which mostly occur while the predictor is warming up.

Whereas ALT+ considers solely I/O patterns when generating predictions, coverage by the ACSU and IASP mechanisms is contingent on the availability of mouse events preceding I/O activity. *Firefox*, *Writer*, and *Calc* show the greatest number of misses and missed opportunities for both ACSU and IASP, meaning that there is a good deal more ambiguity in the mouse events available for prediction generated by these applications than the others. In the case of *Firefox*, most mouse activity occurs within the window displaying the visited web pages. As such, the constantly changing structure of the window increases the number of mouse IDs that are encountered resulting in a high misprediction rate for ACSU. IASP, on the other hand, does not spin-up the disk for these clicks, since their IDs are not observed as often as those that belong to the static part of the GUI. In the case of *Writer* and *Calc*, the relatively low coverage by both ACSU and IASP mechanisms is caused by lower availability of clicks preceding I/O. While most or all functionality of these applications is accessible through the GUI, the interaction is made simpler through the use of keyboard shortcuts. As we are only considering mouse events, any I/O that occurs in response to a keyboard shortcut is not predicted by the mechanisms. The applicability of keyboard events to I/O prediction will be explored in future research.

The applications for which both ACSU and IASP perform best are *Impress*, *Dia* and *Gimp*. These applications have more complex GUIs for which extensive keyboard shortcuts are not intuitive to an average user. This is the case with *Dia* and especially *Gimp*. All three of these applications also depend heavily upon the mouse, due to the graphical nature of their content and usage. Manipulating images, graphs, and figures is done most easily with the mouse and in our traces the user depended more heavily on the mouse for all interactions with these applications.

5.2 Impact of the Buffer Cache

High file I/O prediction accuracy shown in Figure 5 represents the strong correlation between mouse clicks and file I/O. However, file I/O may also be satisfied by a buffer cache access, making a disk spin-up unnecessary. Hence, we have to consider the impact of the buffer cache on prediction accuracy. From this point on, all figures show the mechanism with the buffer cache enabled. We set the buffer cache to 512MB, which is representative of current systems' capabilities. The buffer cache can satisfy many file I/Os resulting in fewer required disk accesses. In addition to introducing additional randomness into file I/O patterns, the buffer cache also increases the training time of prediction mechanisms due to both inclusion of the access history and and fewer spin-ups encountered in the system. Similarly to Figure 5, Figure 6 shows hits, misses, and missed opportunities, but those metrics now realistically reflect the actual required disk I/O.

The aggressiveness of our ACSU mechanism again makes it a top performer when we consider the amount of periods covered with correct spin-ups. This behavior can be expected since the mechanism just keeps the disk on no matter if the buffer cache satisfies the request or not. Therefore, this mechanism's behavior is not impacted by the presence of the buffer cache. The different fraction of period misses, and hits, as compared to Figure 5 are due to a change in the periods' composition since we have fewer and longer periods due to filtering of I/O by the buffer cache. In this case, ACSU is able to spin up the disk ahead of time for 66% of required periods, while incurring misprediction rates as high as 54% with the average of 52%. The ALT+ mechanism is also not impacted much by the buffer cache since the randomness observed by the file I/O in the interactive application is already large rendering this mechanism not very useful in either case. The coverage is as low as 3% with an average of 7%, incurring a high misprediction rate of 24% on average.

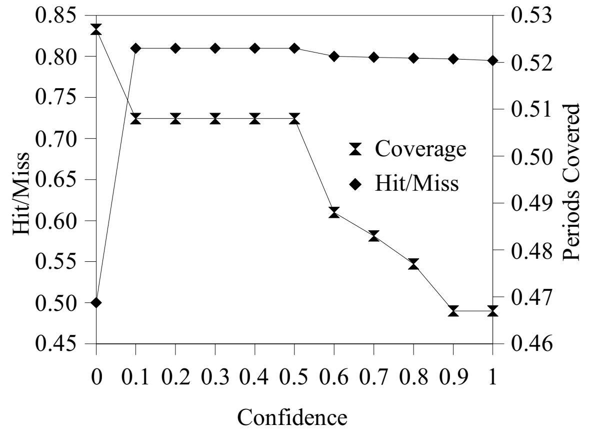


Figure 7: Hit/Miss ratio and I/O activity period coverage vary as the acceptable confidence level is increased.

The impact is more pronounced in the case of IASP, since IASP uses contextual prediction selectively to predict what user activity will result in disk I/O. Therefore, the introduction of any randomness by the buffer cache affects the accuracy of our history-based IASP disk I/O prediction. IASP remains the most accurate mechanism, resulting in only 2% mispredictions while achieving 65% of correct spin-ups, on average. Low misprediction rate indicates that the randomness introduced by the buffer cache is insignificant and the history-based prediction is able to capture correctly the behavior of the disk I/O. Lower coverage indicates that the fewer I/O periods increase the fraction of learning time. It is worth noting that the significance of learning time decreases the longer the system stays on.

In the case of all applications except for *Dia*, the lower coverage of the IASP mechanism as compared to the coverage of the uncached I/O is due to learning, since predictions followed by an absence of disk I/O due to caching result in fewer learning opportunities. Interaction with GUI elements results in the requisite file data being stored in the cache. In the absence of a cache, even the infrequently used elements would generate disk

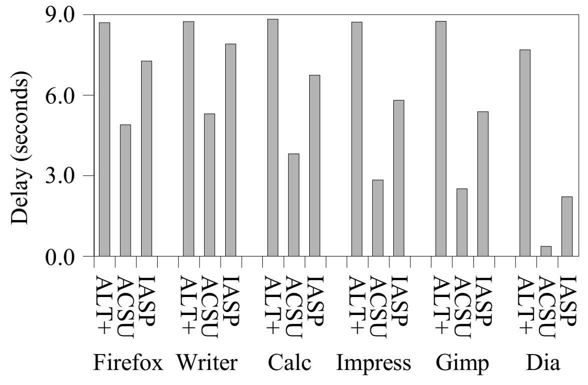


Figure 8: Average delay in seconds, WD2500JD.

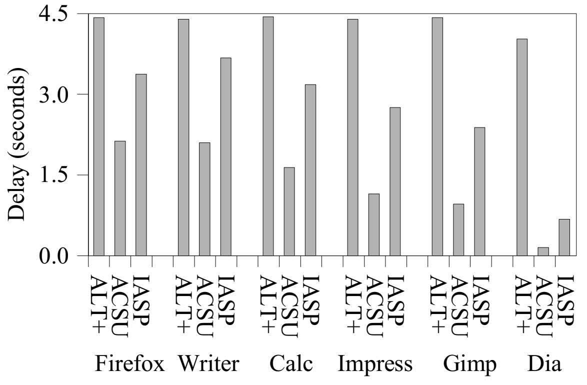


Figure 9: Average delay in seconds, 40GNX.

I/O, but not so with the cache. In general, IASP greatly reduces the number of unnecessary spin-ups that are present in ACSU, at the cost of lower coverage, due to more energy-efficient spin-up policies.

In the case of *Dia*, the type of interactions encountered during tracing were limited to very simple actions, such as opening, creating and saving a number of files containing various simple figures, meaning that the availability or absence of I/O was quickly learned by IASP. Creating even the simplest diagrams may require a large number of clicks. ACSU therefore exhibits a large number of mispredictions in this case, while IASP easily filters out the events that cause the program to, for example, draw a triangle rather than open a file.

5.3 Confidence Levels

Confidence levels are dynamically set thresholds for prediction within IASP. Recall that confidence levels associated with the mouse events represent the ratio of how many times the event was observed and the number of times the event was followed by file I/O. A given confidence level dictates the amount of predictions made and the prediction accuracy as illustrated in Figure 7. Confidence of 1 means that the click is always followed by I/O activity, confidence of .9 means that the click is followed by I/O activity 90% of the time, and so on. If the confidence level is set too low, the predictor may spin-up the disk early in response to events that rarely lead to I/O activity. For example, the user clicked on File menu but interacted with options that did not involve disk I/O. However, early spin-ups hide more latency if the interaction leads to disk I/O. Setting the confidence level too high, however, may delay the disk spin-up and potentially expose the entire spin-up latency to the users. It is therefore important to set confidence levels such that the energy consumption caused by early-erroneous spin-ups and the delay reduction offered by the early spin-ups are in balance.

Figure 7 illustrates impact of confidence on Hit/Miss ratios and file I/O period coverages. We define coverage as a fraction of correctly predicted spin-ups in the applications. The ratio of hits to misses shows the average accuracy over all applications in predicting upcoming I/O activity. We see that due to optimistic prediction, the hit/miss ratio declines slightly as the acceptable confidence level increases past 0.5, but overall remains steady at just over 80%. The sharp increase and steady behavior in the hit/miss ratio indicates quick convergence during training and stable behavior for each mouse event. Increasing confidence level past 0.5 results in longer training and the predictor's coverage drops sharply, since it attempts to predict fewer and fewer disk spin-ups.

5.4 Delay Reduction

Figures 8 and 9 show the average spin-up delays that are exposed to the user in the case of each of the two disks. Each missed opportunity seen in Figure 6 results in delay equal to the average spin-up time, 9 seconds in the case of WD2500JD and 4.5 seconds in the case of 40GNX. On the other hand, hits described in Figure 6 are predictions that result in the disk spinning up correctly and may arrive either early enough to allow the disk to spin-up before the I/O arrives, resulting in no delay, or late, where the disk is in the process of spinning up when I/O arrives. The demand based spin-up exposes full spin-up delay to the application, during every spin-up, and therefore the average delay in demand based system is full 9 seconds in the case of WD2500JD and 4.5 seconds in the case of 40GNX.

ACSU is very aggressive in reducing spin-up delays at the expense of increased energy consumption. High coverage of file I/O periods in Figure 6 results in an average spin-up delay reduction from 9 seconds to 3.3, which is only 37% of the spin-up delay exposed by the

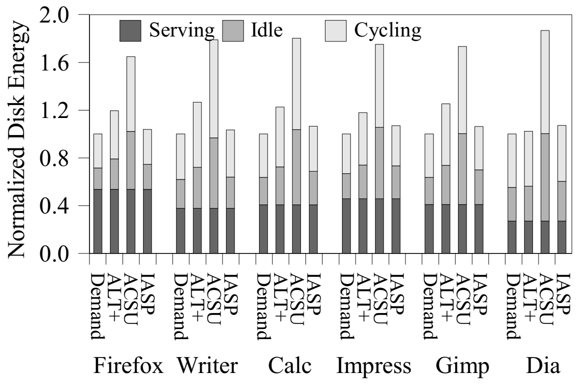


Figure 10: Energy consumption, WD2500JD.

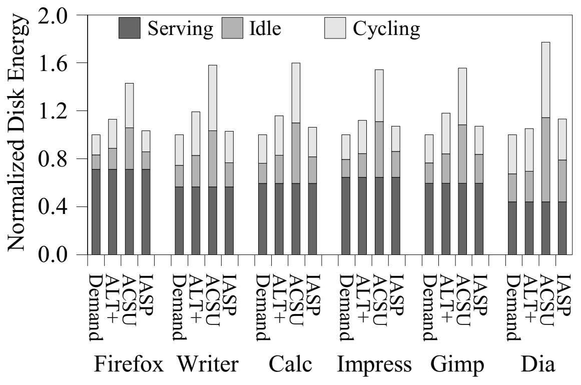


Figure 11: Energy consumption, 40GNX.

demand-based spin-up for the WD2500JD. In the case of the 40GNX disk, shown in Figure 9, the average spin-up delay is reduced to 1.36 seconds, which is 30% of demand-based spin-up delay. As expected from Figure 6 ALT+ performs poorly exposing high delays of 8.58 seconds and 4.35 seconds for the WD2500JD and 40GNX, respectively. The exposed delays in ALT+ are comparable to demand based spin-up delays, since most periods require on-demand spin-up.

IASP is able to shorten interactive delays exposed to the users down to 5.89 seconds and 2.67 seconds for WD2500JD and 40GNX, respectively, while maintaining high accuracy and low energy consumption. IASP exposes 2.6 seconds and 1.3 seconds more delay than ACSU for WD2500JD and 40GNX, respectively. ACSU sets the lower bound on the spin-up delay for mechanisms that utilize mouse interaction since it spins up or keeps the disk on for all mouse interaction as shown by the higher coverage in Figure 6. ACSU not only captures more I/O periods, but also does so earlier than IASP, since it is not governed by the confidence requirement set in IASP to prevent erroneous spin-ups. ACSU is therefore most effective in situations where low delay is desired, assuming that of course energy-efficiency is also a desired attribute, but to a lesser extent. The higher accuracy of IASP makes it the most desirable choice when energy efficiency is important and users are willing to tolerate slightly higher delays than ACSU provides, which are still much lower than delays exposed by the demand-based spin-ups.

Highest delay reduction is present in *Dia*, where the delay is reduced by 93% and 85% by ACSU and IASP for 40GNX, and 96% and 85% for the WD2500JD, indicating that there is plenty of user think time to overlap spin-up delays. On the other hand, *Writer* shows the lowest reduction in spin-up delays. The most significant factor contributing to the low reduction in spin-up delays in case of Writer is single button interaction with toolbars, which results in I/O activity. For example, if the user clicks on the spell-check button in the toolbar rather than finding spell-check in the Tools menu, the resulting activity arrives quickly following the single mouse event that predicted it.

Reduction in delay is generally accompanied by increase in energy consumption, since we need the disk to remain on in order to minimize the delay. For example, if we allow the disk to remain spinning for the entirety of an application's run, the interactive delays are eliminated, but at the cost of vastly increased disk energy consumption. On the other hand, simple demand-based mechanisms are often the lowest energy solution, due to the fact that they do not have extraneous spin-ups, but they incur delays each time the disk is spun up. We have seen that delay can be significantly improved using our proposed ACSU and IASP mechanisms and now we turn to a discussion of energy consumption.

5.5 Energy

Figures 10 and 11 show the details of energy consumption of the two disks. The energy consumption is divided into I/O serving energy, power-cycle energy, and idle energy. I/O serving energy is consumed by the disk while reading, writing, and seeking data. I/O serving energy is the same for all mechanisms, since the amount of I/O served is the same. Power-cycle energy is consumed by the disk during spin-up and shutdown and is directly related to number of spin-ups which also include erroneous spin-ups. Finally, idle energy is the energy consumed by the disk while it is spinning but not serving any I/Os. Idle energy is dependent on the number of I/O periods and the timeout before the disk is shutdown after an I/O period, additional idle energy consumption occurs in ACSU, IASP, and ALT+ due to mispredictions, during which the disk idles before shutting down when I/O does not arrive. In addition, early spin-ups result in additional energy being consumed by the disk between the time when the disk is ready to serve data and the arrival of the first I/O, which is most prevalent in ACSU.

Due to a large number of mispredictions, ACSU consumes significantly more idle and power-cycle energy than IASP. On average, IASP consumes 30% less energy idling than ACSU, and 40% less energy cycling power modes when using WD2500JD. In 40GNX's case, IASP consumes 27% less idle energy than ACSU, and 25% less cycling energy. On average, with the WD2500JD, IASP consumes 6% more energy than the on-demand mechanism due to waiting after early spin-ups and the few mispredictions that result in the consumption of energy not present in the on-demand mechanism. Similarly, in the 40GNX case, IASP consumes 7% more energy than the on-demand mechanism. Keeping the disk always on has the effect of increasing idle energy consumption to levels that are prohibitively large for energy constrained systems. Overall, the energy consumed by WD2500JD using ACSU is 49% lower than keeping the disk always on, 70% lower in case of IASP, and 65% lower for ALT+. The energy consumed by 40GNX when using IASP is 64% lower, 60% lower for ALT+. Differences in relative energy consumption result from the different power profiles of the two disks in question.

5.6 Overheads

The computational and storage overheads of any power management mechanism have to be taken into consideration, since improving the energy consumption of one device while equally increasing that of another does not result in energy-efficiency. Therefore, it is critical to keep computational requirements to minimum to avoid the excess energy consumption in the processor. Additionally, the storage overheads of a power management mechanism's data should be low enough to be considered insignificant, since storing a large amount of data could potentially impact the execution of interactive applications by polluting data caches in the processor.

Considering those requirements, ACSU has a clear advantage since it does not have to store or compute anything. It simply spins the disk up when the disk is shutdown and a click arrives or resets the timeout variable when the disk is active. IASP, on the other hand, computes IDs that uniquely identify user mouse interactions and store the interaction predictions in a prediction table. Due to the efficiency of hash tables, the only measurable computational overhead is incurred when the unique window ID is computed. *Firefox* has the deepest tree of 27 levels in the studied applications. Therefore, we setup an experiment to measure the average overhead of traversing 27 levels of tree hierarchy and we found the overhead to be negligible. Furthermore, this overhead can be almost eliminated by modifying the X-Window Server to automatically generate mouse click IDs as it itself builds the window tree, rather than building a separate representation as shown in Figure 2.

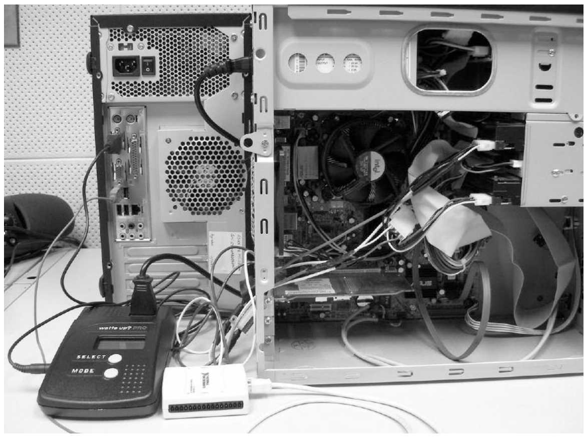


Figure 12: The experimental setup used for measuring power.

The storage overhead is likewise relatively low in IASP. For each unique mouse event we are storing its ID (32 bits), the number of times the event was observed (32 bits), the number of times it was followed by I/O activity (32 bits), and the two-bit history table (8 bits) with two bit saturating counters for the prediction outcome (8 bits). The resulting table entry is 14 bytes. The number of unique click IDs in the studied applications ranged widely from 35 in *Calc* to 195 in *Writer*. Therefore, in the worst case *Writer* requires 2.67KB to store 195 entries. An 11.3KB table would suffice for storing all entries from every one of the six applications we have studied.

5.7 Experimental Evaluation

Experiments were conducted using a setup of two desktop machines with dual-core 3.0GHz processors and 2GB of memory. As shown in Figure 12, a multi-channel data acquisition board (DAQ) from NI was connected to the power cable of a WD2500JD hard drive dedicated to replying the traces. To measure the power consumed, a 0.1 ohms resistor was placed in series with the hard disk power supply and the voltage drop across the resistor was fed to the DAQ. The second machine, running Windows XP and the DAQ drivers, ran the LabView setup sampling measurements at 1000Hz from the DAQ. The simulated trace-driven prediction mechanisms were ported to a driver that replays the traces on the measured hard drive.

Figure 13 shows a selected portion of the *Dia* trace, as replayed on the hardware and captured through the experimental setup. We show several activity periods for on-demand spin-up, ACSU, and IASP. We omit ALT+

due to its low accuracy. Figure 13 shows several Lines C, E, and G which show the spin-ups that were initiated on-demand when the I/O arrived. The period of disk activity beginning at A, as initiated by ACSU, illustrates ACSU's aggressive spin-up policy. At A the user begins a series of interactions which cause the disk to spin up and remain spinning until the interaction ends and I/O is served. The I/O arrived at C and the disk was spun up on-demand to serve the request. IASP, on the other hand, spun-up early at B, in response to a user interaction that predicted that I/O will follow.

Similarly, we see that the next interaction at D, caused both IASP and ACSU to spin-up the disk ahead of E, where the disk was spun up on-demand. Another example occurs at G, with on-demand spin-up shortly following the IASP and ACSU spin-ups. Matching up the trace replay to the simulator output, we have verified that this behavior is indeed expected and due to space constraints we include only limited results from the hardware replay.

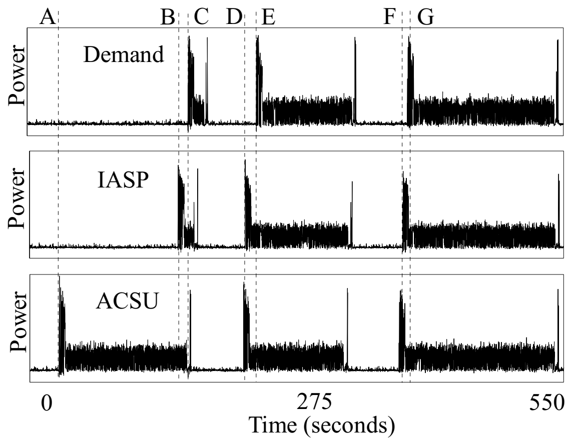


Figure 13: Power consumption in a selected 550 second period from *Dia* under Demand-based spin-up, ACSU, and IASP.

6 Conclusion

In this paper, we proposed two disk spin-up mechanisms: ACSU that simply keeps the disk powered when users are interacting with the application and IASP that accurately and efficiently monitors user behavior. Both mechanisms reduce interactive delays exposed to the users due to energy management in hard disk drives. Hard disks contribute significantly to the overall energy consumption in computer systems. Therefore, aggressive energy management techniques attempt to maintain the hard drive in a low power state as much as possible exposing long latency spin-up delays to the users. Reducing the spin-up delays provides twofold benefit. First, the users are less irritated by constant lags in the responsiveness of the system due to disk spin-ups. Second, shorter delays allow the system to accomplish tasks quicker resulting in less energy being consumed by other components that have to wait for the disk spin-up.

The key observation used in our design is that users are responsible for the demand placed on the system through interactive applications. Therefore, monitoring user interaction patterns with applications provides the opportunity for predicting I/O requests that follow the interactions and use this to spin-up the disk ahead of time, reducing delays. Our evaluation of proposed ACSU and IASP shows significant improvement over modified ALT+ mechanisms in terms of predicting upcoming disk I/O activity and thereby shortening the interactive delay associated with energy management. ALT+ mechanisms are not able to accurately predict I/O activity in interactive applications resulting in an average misprediction rate of 25% that increases energy consumption in the system without providing any benefit of reducing delays with only 7% of periods correctly predicted spin-ups, on average. ACSU mechanisms are very aggressive and achieve 81% of accurate predictions that reduce delays at a cost of 52% misprediction rate. As a result, ACSU is able to reduce spin-up delays on average by over 60% (over 5 seconds), albeit at the cost high energy consumption. Finally, IASP is much more accurate since it is monitors user interactions. IASP on average achieves 79% of accurate predictions that reduce delays with only 2% of mispredictions. As a result, IASP is able to reduce spin-up delays on average by 35% (over 3 seconds), while maintaining low energy consumption.

The primary goal of this paper was to reduce interactive delays due to disk spin-up exposed to the users, while maintaining the energy efficiency of the shutdown mechanism. Spin-up mechanisms do not reduce energy consumption of the individual device, however they have a side effect of making the system more energy efficient by accomplishing tasks quicker and reducing the energy consumed by the system waiting for the disk to spin-up.

References

[1] ALBINALI, F., AND GNIADY, C. CPM: Context-aware power management in wlans. In *Proceedings of the Eighteenth Innovative Applications of Artificial Intelligence Conference (IAAI)* (2006).

[2] BENINI, L., BOGLIOLO, A., PALEOLOGO, G. A., AND MICHELI, G. D. Policy optimization for dynamic power management. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 18*, 6 (June 1999), 813–833.

[3] BIANCHINI, R., AND RAJAMONY, R. Power and energy management for server systems. Tech. Rep. DCS-TR-528, Department of Computer Science, Rutgers University, June 2003.

[4] CHASE, J., ANDERSON, D., THACKAR, P., VAHDAT, A., AND BOYLE, R. Managing energy and server resources in hosting centers. In *Proceedings of the 18th Symposium on Operating Systems Principles* (October 2001).

[5] CHUNG, E.-Y., BENINI, L., BOGLIOLO, A., LU, Y.-H., AND MICHELI, G. D. Dynamic power management for nonstationary service requests. *IEEE Transactions on Computers 51*, 11 (November 2002), 1345–1361.

[6] CHUNG, E.-Y., BENINI, L., AND MICHELI, G. D. Dynamic power management using adaptive learning tree. In *Proceedings of the International Conference on Computer-Aided Design* (Novemebr 1999), pp. 274–279.

[7] CRAVEN, M., AND AMER, A. Predictive reduction of power and latency (purple). In *MSST '05: Proceedings of the 22nd IEEE / 13th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST'05)* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 237–244.

[8] CRK, I., BI, M., AND GNIADY, C. Interaction-aware energy management for wireless network cards. In *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems* (2008).

[9] DIX, A., FINLEY, J., ABOWD, G., AND BEALE, R. *Human-computer interaction (3rd ed.).* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2003.

[10] DOUGLIS, F., KRISHNAN, P., AND BERSHAD, B. Adaptive disk spin-down policies for mobile computers. In *Proceedings 2nd USENIX Symp. on Mobile and Location-Independent Computing* (1995), pp. 381–413.

[11] ELLIS, C. S. The case for higher-level power management. In *Workshop on Hot Topics in Operating Systems* (Rio Rico, AZ, USA, March 1999), pp. 162–167.

[12] ELNOZAHY, M., KISTLER, M., AND RAJAMONY, R. Energy conservation policies for web servers. In *Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems* (March 2003).

[13] GNIADY, C., HU, Y. C., , AND LU, Y.-H. Program counter based techniques for dynamic power management. In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation* (Dec. 2004).

[14] GOLDING, R. A., II, P. B., STAELIN, C., SULLIVAN, T., AND WILKES, J. Idleness is not sloth. In *Proceedings of the USENIX Winter Conference* (1995), pp. 201–212.

[15] HEATH, T., PINHEIRO, E., HOM, J., KREMER, U., AND BIANCHINI, R. Application transformations for energy and performance-aware device management. In *Proceedings of the 11th International Conference on Parallel Architectures and Compilation Techniques* (September 2002).

[16] HELMBOLD, D. P., LONG, D. D. E., AND SHERROD, B. A dynamic disk spin-down technique for mobile computing. In *Mobile Computing and Networking* (1996), pp. 130–142.

[17] HWANG, C.-H., AND WU, A. C. A predictive system shutdown method for energy saving of event driven computation. *ACM Transactions on Design Automation of Electronic Systems 5*, 2 (April 2000), 226–241.

[18] KARLIN, A. R., MANASSE, M. S., MCGEOCH, L. A., AND OWICKI, S. Competitive randomized algorithms for non-uniform problems. In *Symposium on Discrete Algorithms* (1990), pp. 301–309.

[19] LARKBY-LAHET, J., SANTHANAKRISHNAN, G., AMER, A., AND CHRYSANTHIS, P. K. Step: Self-tuning energy-safe predictors. 125–133.

[20] LU, Y.-H., MICHELI, G. D., AND BENINI, L. Requester-aware power reduction. In *Proceedings of the International Symposium on System Synthesis* (2000), pp. 18–24.

[21] NIGHTINGALE, E. B., AND FLINN, J. Energy efficiency and storage flexibility in the blue file system. In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation (OSDI)* (December 2004).

[22] PINHEIRO, E., BIANCHINI, R., CARRERA, E. V., AND HEATH, T. Load balancing and unbalancing for power and performance in cluster-based systems. In *Proceedings of the Workshop on Compilers and Operating Systems for Low Power* (September 2001).

[23] QIU, Q., AND PEDRAM, M. Dynamic power management based on continuous-time markov decision processes. In *Proceedings of the Design Automation Conference* (New Orleans, LA, USA, June 1999), pp. 555–561.

[24] RYBCZYNSKI, J. P., LONG, D. D. E., AND AMER, A. Expecting the unexpected: adaptation for predictive energy conservation. In *StorageSS '05: Proceedings of the 2005 ACM workshop on Storage security and survivability* (New York, NY, USA, 2005), ACM Press, pp. 130–134.

[25] SAMSUNG. Samsung teams with microsoft to develop first hybrid hard drive with nand flash memory, 2005.

[26] SIMUNIC, T., BENINI, L., GLYNN, P., AND MICHELI, G. D. Dynamic power management for portable systems. In *Proceedings of the International Conference on Mobile Computing and Networking* (2000), pp. 11–19.

[27] SRIVASTAVA, M. B., CHANDRAKASAN, A. P., AND BRODERSEN, R. W. Predictive system shutdown and other architecture techniques for energy efficient programmable computation. *IEEE Transactions on VLSI Systems 4*, 1 (March 1996), 42–55.

[28] WEISSEL, A., AND BELLOSA, F. Self-learning hard disk power management for mobile devices. In *Proceedings of the Second International Workshop on Software Support for Portable Storage (IWSSPS 2006)* (Seoul, Korea, Oct. 2006), pp. 33–40.

[29] WEISSEL, A., BEUTEL, B., AND BELLOSA, F. Cooperative I/O—a novel I/O semantics for energy-aware applications. In *Proceedings of the Fifth Symposium on Operating System Design and Implementation* (December 2002).