PREFiguRE: An Analytic Framework for HDD Management

FENG YAN, College of William and Mary
XENIA MOUNTROUIDOU, Jacksonville University
ALMA RISKA, NetApp Corporation
EVGENIA SMIRNI, College of William and Mary

Low disk drive utilization suggests that placing the drive into a power saving mode during idle times may decrease power consumption. We present PREFiguRE, a robust framework that aims at harvesting future idle intervals for power savings while meeting strict quality constraints: first, it contains potential delays in serving IO requests that occur during power savings since the time to bring up the disk is not negligible, and second, it ensures that the power saving mechanism is triggered a few times only, such that the disk wear-out due to powering up and down does not compromise the disk's lifetime. PREFiguRE is based on an analytic methodology that uses the histogram of idle times to determine schedules for power saving modes as a function of the preceding constraints. PREFiguRE facilitates analysis for the evaluation of the trade-offs between power savings and quality targets for the current workload. Extensive experimentation on a set of enterprise storage traces illustrates PREFiguRE's effectiveness to consistently achieve high power savings without undermining disk reliability and performance.

Categories and Subject Descriptors: C.4 [**Computer Systems Organization**]: Performance of Systems

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Performance modeling, scheduling, disk drives, power savings, reliability, histograms

ACM Reference Format:
Feng Yan, Xenia Mountrouidou, Alma Riska, and Evgenia Smirni. 2016. PREFiguRE: An analytic framework for HDD management. ACM Trans. Model. Perform. Eval. Comput. Syst. 1, 3, Article 10 (May 2016), 27 pages.
DOI: http://dx.doi.org/10.1145/2872331

1. INTRODUCTION

Storage systems in data centers host thousands of disk drives. Despite emerging new storage technologies, such as solid state drives (SSDs), it is the hard disk drives (HDDs) that continue to store the overwhelming majority of corporate data [Vasudeva 2011; Grupp et al. 2012; Narayanan et al. 2009]. Specifically, HDDs are expected to store aging data (from a few weeks old to several years old) that are expected to grow in size over the years. Given the characteristic of data stored in HDDs, it is expected that not all data in a vast data center is accessed simultaneously. Consequently, a

This work was supported by the National Science Foundation under grants CCF-0937925 and CCF-1218758.
Authors' addresses: F. Yan, Room 140, McGlothlin-Street Hall, Department Computer Science, College of William and Mary, Williamsburg, VA, 23187; email: fyan@cs.wm.edu; X. Mountrouidou, Wofford College, Department of Computer Science, 445 Melbourne Ln., Spartanburg, SC, 29301; email: xenia.mountrouidou@gmail.com; A. Riska, 57 Pine Plain Rd., Wellesley, MA, 02481; email: alma.dimnaku@gmail.com; E. Smirni, P.O. Box 8795, Department of Computer Science, College of William and Mary, Williamsburg, VA, 23187; email: esmirni@cs.wm.edu.

DOI: http://dx.doi.org/10.1145/2872331

compelling approach for reducing power consumption in data centers is to spin down idle HDDs. This approach is routinely deployed in storage systems that serve as archival or backup systems [Colarelli and Grunwald 2002] and is being exploited even in high-end computing environments [Narayanan et al. 2008].

Spinning down disk drives to save energy in a high-end environment *transparently* to the end user and *reliably* to the disk drive's lifetime is a challenging open problem for a host of reasons. First, in enterprise environments, requests that arrive while the drive is in a power saving mode are to be inevitably delayed during the time it takes for the disk drive to reactivate (e.g., to be physically ready to serve jobs again). Second, idle times can be highly fragmented while the overall drive utilization is very low, and therefore idle periods that are long enough to be used effectively for power savings may be very few [Riska and Smirni 2010]. Third, every power up/down wears out the disk drive, which implies strict limitations on the number of times a disk drive can be placed into a power saving mode without affecting its reliability.

Common practice methods try to address these challenges by idle waiting for a fixed amount of time or use the past utilization to guide future scheduling decisions. However, these common practice methods cannot provide performance guarantees nor take into consideration disk reliability. To overcome these shortcomings, we present PREFiguRE, a framework that uses as input user- or system-level constraints (e.g., the number of allowable power ups/downs of a disk within a time period (strict constraint) and the user acceptable potential performance degradation of future IOs (soft constraint)) and estimate the projected power savings as well as provide a strategy on *how* these power savings should occur. PREFiguRE uses as a basic tool the histogram of past idle times and projects future power savings based on statistical information that is monitored or extracted from this histogram. Probabilistic interpretation of all of the preceding information leads PREFiguRE to define robust schedules for power saving modes. As the workload changes in the system, the histogram of idle times and information about the sequence of idle times are updated. Such updates enable the adjustment of the schedules of power saving activation to the workload dynamics.

The core of PREFiguRE is a robust, accurate, and computationally efficient analytic model that enables the identification of effective, user-transparent schedules of power saving modes in disk drives. Most importantly, the analytic model that is encapsulated in PREFiguRE encompasses a strong reliability component to comply with the restrictions on the number of times a hard disk can go into a specific power saving mode during its lifetime [Kim and Suk 2007]. In addition, thanks to the excellent prediction accuracy of the model, it is possible to answer a wide range of questions regarding the power saving capabilities of the current disk workload. For instance, if the power gains are projected to be marginal, then it may not be worth engaging the system in any power savings mode or it may signal that part of the workload should be offloaded (to a buffer or to another disk) such that idle times, and consequently, power savings, are increased.

Although the main contribution of our framework lies in its theoretical aspect, we also conduct trace-driven simulations to verify its practical benefit. We drive the evaluation of PREFiguRE via a set of enterprise disk drive traces with a wide range of idleness characteristics. The excellent agreement between the results from PREFiguRE's analytic estimations and the trace-driven simulations suggests that our analytic methodology can achieve good accuracy and robustness even under the real-world workloads.

This article is organized as follows. Section 2 summarizes the power saving opportunities in disk drives and storage systems. In Section 3, we present the methodology that we propose to identify and estimate the power saving opportunities in a system under a given workload. We validate the effectiveness of the approach and illustrate its robustness in Section 4 using trace-driven analysis and simulations. Section 5

Table I. Characteristics of Power Saving Modes

Operation Mode	Description	Power Savings	Penalty (seconds)
Level 1	Serving IOs	0%	0.0
Level 2	Active (but) idle	40%	0.0
Level 3	Unloaded heads	48%	0.5
Level 4	Slowed platters	60%	1
Level 5	Stopped platters	70%	8
Level 6	Shut down	95%	25

positions our contributions relatively to related work. Conclusions and future work are given in Section 6.

2. POWER SAVING MODES IN DISK DRIVES

Disk drives represent the overwhelming majority of the storage devices deployed in large data centers where power conservation is a priority. Individual disk drives consume moderate amount of power compared to other components in a computer system. However, disk drives tend to be more idle than other system components. This is particularly true in large data centers that deploy thousands of disk drives and host terabytes and petabytes of data, which are not all accessed simultaneously.

Disk drives are complex hardware devices that consist of both mechanical and electronic components. The mechanical components, such as the platters that rotate at high speeds, or the positioning arm that is kept at a specific distance away from the platters, continue to consume power even when not accessing data. Similarly, the electronics in a disk drive consume power even during periods of idleness. Overall, disk drives consume less power when they are idle than when they serve IOs.

Beyond the moderate power savings when an active disk is idle (i.e., the "active idle" state), additional power can be saved by slowing down components in a disk drive, such as platter rotation, or by unloading and parking the heads (and the positioning arm) on the side instead of flying them at constant height over the platters. Finally, completely shutting down the disk drive eliminates almost the entire power consumption from the disk drive. Slowing or shutting down the disk comes nonetheless with a performance cost to user IOs, because bringing the disk back to its active state takes time, which ranges from hundreds of milliseconds to tens of seconds. This required time period to reactivate a disk drive can be viewed as an unavoidable performance *penalty* paid by those IOs that by arrival find the disk drive that stores their data in an inactive (i.e., power saving) mode.

There are several levels of power consumption depending on the state of the disk's mechanical and electronic components. Each power consumption level or mode is characterized by the amount of *power* it consumes and the amount of *time* it takes to get out of the power saving mode and become ready to serve IOs. The exact amount of power saved in a given power saving mode, or the amount of time it takes to become ready again, differs between disk drive families and manufacturers. Table I presents a coarse description of the possible power saving modes focusing on the components that are slowed down or shut off and the penalties associated with each power saving mode. The reported penalty values are within representative ranges published by disk drive manufacturers [Seagate Technology 2012, 2014; Hitachi Global Storage Technologies 2007]. For example, the penalty (in seconds) for Level 6 is between 23 (typical) and 30 (max) [Seagate Technology 2014].

Note that during the process of bringing a disk drive out of a power saving mode, the consumed power surges before settling to a normal consumption level. As with the

power savings in Table I, this power surge during reactivation depends on the drive family and manufacturer.

The time it takes a disk to become active following a power saving mode make obvious the need to account for the performance penalty before deciding on a disk operation mode for power savings. One could argue that putting the disk into an idle mode immediately after any idleness is detected could maximize power savings. Given the stochastic nature of the length of idle times and the penalty to bring the disk up to active mode, it is important to use idle intervals that are sufficiently large (i.e., longer than the reactivation time) for power savings. In storage systems, it is very common to not put the system automatically in a power saving mode when an idle interval is observed. Instead, the system waits for a time period in anticipation of future IO arrivals.

In addition to the performance penalty associated with reactivating a disk drive that is put in a power saving mode, there is a reliability penalty as well. The latter is not straightforward to quantify, because it is associated with the wear-out of the disk drives during power ups (i.e., in the spin-up phase) or reactivation of individual components. In disk drives, the spin-up/down (Levels 5 and 6 in Table I) involves certainly more components than loading/unloading heads (i.e., Level 3 in Table I) or spinning platters slower while heads are parked on the side (i.e., Level 4 in Table I). While spin-up and spin-down have been analyzed for years as part of the disk drive wear-out process [Li et al. 1994], the heads load/unloading in disk drives is more recent and is introduced solely for the purpose of power savings [Kim and Suk 2007]. As is discussed in the following sections, in the enterprise environment, loads/unloads (Level 3) are expected to occur more often because the penalty to bring the HDD into the active state is smaller than the other power saving levels. During its lifetime, a disk drive is expected to survive well beyond 300,000 loads/unloads [Kim and Suk 2007], which is used as a threshold in the methodology in the article.

In the following section, we present a framework that determines when and for how long a disk drive should be put into a power saving mode without violating a predefined quality of service target. The framework takes into consideration both the performance and reliability penalties associated with disk drive power saving modes.

3. ALGORITHMIC FRAMEWORK

Here, we develop an algorithmic framework that determines the schedule of the periods when a disk drive is placed in power saving modes such that predefined targets of system quality metrics are met. There are three system quality metrics used in the framework. They include the performance degradation D, the portion of time the disk is placed in power saving modes S, and the reliability constraint X. A definition of these metrics and other notations used in the framework are given in Table II. Note that it is not necessary to have all three system quality metrics set. For example, if only the performance target D and the reliability target X are set, then the framework can meet those targets while the third one (i.e., power saving S) is maximized. It is also possible to set all three metrics, but whether all targets can be met depends on the viability of the workloads. Note also that the application performance can be impacted by many factors (e.g., CPU, memory, networking), and thus for an unbiased analysis, we focus only on the disk performance itself, which is measured by the average response time of IO requests.

In addition to the system quality targets, our framework bases its calculations on a set of monitored (or predefined) input metrics. In particular, it uses the time penalty P that is necessary to bring a disk drive out of a specific power saving mode. Recall that different power saving modes have different penalties P. However, because P depends on the disk drive model, the correct Ps for a given disk drive can be either received

Table II. Notation Used in Section 3

Input Parameters	
D	*Quality metric—performance*: Relative average response time increase due to power savings (in percentage).
S	*Quality metric—power savings*: Portion of time in power savings (in in percentage).
X	*Quality metric—reliability*: Number of reactivations per time unit a disk can have without impacting its lifetime.
P	Penalty due to power savings (i.e., time to reactivate a disk from a specific power saving mode).
Monitored Metrics	
$p(j)$	Probability of idle interval of length j.
$CDH(j)$	Cumulative probability of an idle interval of length at most j.
$E[idle]$	Average idle interval length.
RT	Average IO request response time.
Intermediate Metrics	
W	Average additional wait time IO requests experienced due to the disk in a power saving mode.
w_i	Additional waiting time affecting IOs in the i^{th} busy period following a power saving mode.
$Prob_i(w)$	Probability of w waiting time for the IOs in the i^{th} busy period following a power saving mode.
i_j	Length of the j^{th} idle interval following a power saving mode.
$Prob(LL_l)$	Probability of two idle intervals of at least length L to be l lags apart.
Output Parameters and Estimated Metrics	
I	Amount of time that should elapse in an idle disk before it is put into a power saving mode.
T	Maximum amount of time that a disk is kept in a power saving mode.
$D_{(I,T)}$	Achieved average degradation of response time due to power savings.
$S_{(I,T)}$	Achieved time in power savings.

Note: All time units are in milliseconds.

from the manufacturer or measured in offline testing. Note that P is the *extra* delay due to power saving. This delay is in addition to any queuing delays that requests may experience due to bursty or heavy arrivals. Throughout this article, the focus is on estimating and reducing the delay due to power saving. The set of monitored metrics used in our framework include the *cumulative data histogram* (CDH) of idle times observed in the system and the *average response time* RT of IOs (excluding any slowdown effect that previous power saving modes may have had on average IO response time). The CDH is a list of tuples (at most a few thousands of them). We stress that this representation is very efficient both memory-wise and computation-wise. As we show later in this section, the estimation of scheduling parameters to meet the required targets only requires a few scans of the CDH, which can be executed almost instantaneously. Each tuple contains a range of idle interval lengths and their corresponding empirical cumulative probability. Note that the CDH of idle times is used to capture the characteristics of the overall workload in our framework. As a result, the granularity of the CDH bins determines the accuracy of the estimations and calculations. The coarser the CDH, the less accurate is our solution.

The monitored metrics can be easily obtained from the arrival and departure times of IO requests in the system, which are generally monitored or can be monitored without complex instrumentation. The framework adapts its decisions to changes in workload (captured via the histogram of idle times, system utilization, and IO response time) and other inputs. As a result, the output of the framework (i.e., the schedule of the

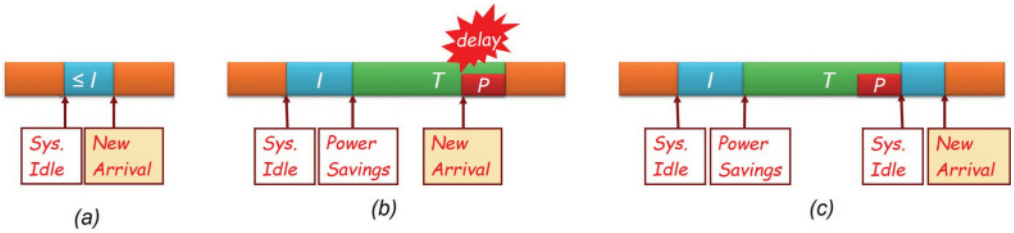

Fig. 1. Examples of relationship between idle periods length, requests arrival, and parameters I, T, and P. Orange represents busy times, blue represents idle times, and green represents power saving mode.

power-saving modes) changes if the workload that arrives in the disk drive changes or if the system quality targets change. For example, in an enterprise storage system, the performance quality target D can be adjusted to be more stringent during the day (i.e., business hours) and less stringent during the night (i.e., nonbusiness hours). Another example is that the framework can estimate for a given performance target D and reliability target X the time in power saving if Level 3 is used or if Level 4 is used. Comparing the resulting time in power saving S allows the system to decide which power saving mode to use (if any) for the current workload.

In our framework, power saving modes *always* take advantage of only the *idle periods* in a disk drive and are not purposely scheduled if user requests are waiting for service in the system. This condition must be satisfied even if the target power saving S is set and not met. Here, we assume that the user workload has always a higher priority, although our framework can be adapted to a situation where power savings have the same priority or higher than the performance of user workload.

Given this consideration, we model the power saving modes as *low-priority tasks* that need P units of time to be preempted. The IO requests arriving in the system are modeled as *high-priority tasks*. Because the penalty P to preempt the low-priority work (i.e., the time to reactivate the disk) is orders of magnitude higher than the expected service and response time of user IOs, the performance impact that power saving modes could have on user IOs may be significant. Our framework schedules power saving modes in disk drives proactively (i.e., average IO slowdown is limited to the performance target D). The framework achieves its targets by scheduling power saving modes according to parameters I and T, where

—I represents the amount of time the system remains idle before a power saving mode starts, and
—T represents the maximum amount of time the disk remains in a power saving mode (i.e., if an IO arrives before T elapses, the power saving mode is interrupted). T includes the penalty P, which implies that $T > P$.

The scheduling pair (I, T) is recalculated every time the monitored metrics are updated or the system quality target changes, adapting the scheduling of power saving modes to the dynamics in the storage system.

Figure 1 demonstrates three examples of the relationship between idle period length, arriving requests, and parameters I, T, and P. Figure 1(a) shows an idle period smaller than I, Figure 1(b) shows an idle period larger than I but smaller than $I + T$, and Figure 1(c) shows an idle period that is larger than $I + T$.

3.1. Modeling Waiting Times Due to Power Saving Modes

In our framework, the scheduling pair (I, T) is calculated such that it guarantees the quality targets (reliability, performance, and/or amount of power savings). To meet

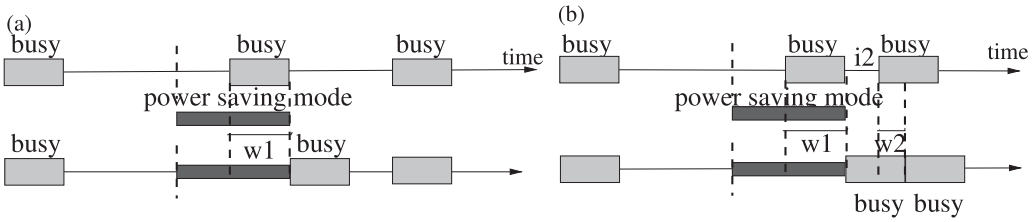

Fig. 2. (a) No delay propagation. (b) Delay propagates two busy periods.

the performance or power saving target, it is critical to estimate correctly the *waiting time* (or delay) caused to IOs arriving during or after a power saving mode. Without loss of generality, we measure the idle interval length as well as the wait within the 1ms granularity. The coarser the granularity, the less the accuracy, but the monitoring overhead is expected to reduce.

Assume that W is the average IO waiting due to the power savings (i.e., $W = RT_{w\ power\ saving} - RT_{wo\ power\ saving}$). Because a disk is loaded upon an IO arrival, W can be at most P (i.e., the time it takes the disk to become active). By denoting a possible delay by w and its respective probability by $Prob(w)$, then

$$W = \sum_{w=1}^{P} w \cdot Prob(w). \tag{1}$$

We define a busy period as the time period when there are one or several IO requests being served without idle time between requests. The power saving mode preemption time P may be longer than the average idle interval. As a result, the delay due to a power saving mode may not be absorbed by the immediately following idle period and may *propagate* to impact multiple user busy periods. Figure 2 shows an example of no delay propagation and one where delay propagates two busy periods. As shown in Figure 2(a), the idle period following the second busy period is longer than the delay caused by power savings; therefore, the delay is absorbed and does not propagate further. Figure 2(b) is an example of when delay propagates two busy periods. The idle period after the second busy period is very short and the delay caused by power savings propagating into the third busy period, and therefore IO requests in both the second and third busy periods are delayed. Although all IOs in one busy period get delayed by the same amount, the delay propagates to multiple busy periods and different delays may be caused to IOs in future busy periods because of the activation of a single power saving mode.

To estimate $Prob(w)$ of a delay w, we identify the events that happen during disk reactivation that result in a delay w and estimate their corresponding probabilities. These events are the basis for the estimation of the average waiting W due to power savings. Without loss of generality, we assume that a disk reactivation affects at most K consecutive user busy periods. The larger the K, the more accurate is our framework. In general, the larger the P, the larger the value of K should be for better estimation accuracy. In our estimations, K is set to be equal to P, which represents the largest practical value that K should take. During disk reactivation, the delay propagates as follows:

—*First delay*: User IOs arrive during a power saving mode or disk reactivation and find an empty queue and a disk that is not ready for service. These IOs would have made up the *first* user busy period if the disk would have been ready. Their waiting due to power saving is w_1 ms (where the index $i = 1$ indicates the first busy period and $1 \leq w_1 \leq P$).

—*Second delay*: User IOs in the "would-be" second busy period in the absence of the power saving mode could also be delayed if the preceding wait w_1 is longer than the idle interval i_2 that would have followed the preceding first busy period. The waiting time experienced by the IOs of the second busy period following a power saving mode is $w_2 = (w_1 - i_2)$.
—*Further propagation*: In general, the delay propagates through multiple consecutive user busy periods until all intermediate idle periods absorb the initial delay w_1. Specifically, the delay propagates for K consecutive user busy periods if $(i_2 + i_3 + \cdots + i_K) < w_1 < (i_2 + i_3 + \cdots + i_K + i_{K+1})$. The waiting times experienced by the IOs due to this power saving mode are w_j for $1 \leq j \leq K$.

Denoting with $Prob_k(w)$ the probability that wait w occurs to the IOs of the k^{th} delayed busy period, we estimate the probability of delay w as $Prob(w) = \sum_{k=1}^{K} Prob_k(w)$. The delay P may occur only to IOs of the first delayed busy period, because for the IOs of the second (or higher) delayed busy period, the intermediate idle interval would absorb some of the delay and would therefore reduce it. The same argument can be used to claim that the delay of $P - 1$ can occur only to IOs of the first and second delayed busy periods. In general, it is true that the delay $w = P - k$ may occur only to IOs of the first $k + 1$ delayed busy periods ($0 \leq k \leq K$).

The preceding fact is used as the base for our recursion that computes $Prob(w)$ for $1 \leq w \leq P$. The base is $w = P$, and $Prob(w = P) = Prob_1(P)$ because the delay P is caused *only* to the IOs of the first delayed busy period. For a scheduling pair (I, T), the delay to the first busy period following a power saving mode is P for all idle intervals whose length falls between I and $I + T - P$. The probability of this event is given as $\text{CDH}(I + T - P) - \text{CDH}(I)$, where CDH(.) indicates the cumulative probability value of an idle interval in the monitored histogram.

The delay w caused to the IOs in the first busy period following a power saving mode may be any value between 1 and P. This delay cannot exceed P, as P is the time that the disk required to revert from power saving mode to serving mode. Using the CDH of idle times, the probability of any delay w caused to the IOs of the first busy period are given by the following equation:

$$Prob_1(w) = \begin{cases} \text{CDH}(I + T - w + 1) - \text{CDH}(I + T - w), & \text{for } 1 \leq w < P, \\ \text{CDH}(I + T - P) - \text{CDH}(I), & \text{for } w = P. \end{cases} \tag{2}$$

If the length i_2 of the idle interval following the first delayed busy period is less than w, then the IOs of the second busy period may be delayed as well by $w - i_2$. The IOs of the second busy period are delayed by $w - i_2$ if (1) the idle interval following the first delayed busy period is i_2, which happens with probability $Porb(i_2)$, and (2) the first delay was $w + i_2$, which happens with probability $Porb_1(w + i_2)$. Since there is independence between the arrival and service processes, the delay propagation is also independent of the process of idle lengths. Therefore, the probability $Prob_2(w)$ is given by the following equation:

$$Prob_2(w) = \sum_{j=1}^{P-w} Prob_1(w + j) \cdot p(j), \tag{3}$$

where $Prob_1(w + j)$ for $1 \leq j \leq P - w - 1$ is defined in Equation (2) and $p(j)$ is the probability of an idle interval of length j.

The delay $P - 1$ can occur only to the IOs of the first busy period with probability $Prob_1(P - 1)$ and to the second busy period with probability $Prob_2(P - 1)$. Using

Equations (2) and (3), we get

$$Prob(P-1) = Prob_1(P-1) + Prob(P) \cdot p(1). \tag{4}$$

This implies that $Prob(P-1)$ depends only on $Prob_1(.)$ and $Prob(P)$, which are both defined in Equation (2), and represents how the base $Prob(P)$ of our recursion is used to compute the next probability, $Prob(P-1)$.

Similarly, we determine the probabilities of delays propagated to the IOs of the busy periods following the power saving mode and establish recursion for all $1 \leq w \leq P$. For clarity, we show how we develop the next step recursion and then generalize. Specifically, delay w is caused to the IOs of the third delayed busy period, and w takes values from 1 to at most $P-2$ (recall that the granularity of the idle interval length is 1ms).

$$Prob_3(w) = \sum_{j=1}^{P-w} Prob_1(w+j) \sum_{j_2=1}^{j-1} Prob_2(j-j_2) \cdot p(j_2). \tag{5}$$

The delay of $P-2$ does not propagate beyond the third delayed busy period, and its probability is given as the sum of probabilities of its occurrence to IOs of the first delayed busy period, $Prob_1(P-2)$, second delayed busy period, $Prob_2(P-2)$, and third delayed busy period, $Prob_3(P-2)$. Using Equations (2), (3), and (5), we obtain

$$\begin{aligned} Prob(P-2) \;=\; & Prob_1(P-2) \\ & + Prob_1(P-1) \cdot p(1) + Prob_1(P) \cdot p(2) \\ & + Prob_1(P) \cdot p(1) \cdot p(1). \end{aligned} \tag{6}$$

Substituting $Prob_1(P-1) + Prob(P) \cdot p(1)$ with $Prob(P-1)$ from Equations (4), we get

$$\begin{aligned} Prob(P-2) \;=\; & Prob_1(P-2) \\ & + Prob(P-1) \cdot p(1) + Prob(P) \cdot p(2). \end{aligned} \tag{7}$$

In general, for the k^{th} delayed busy period, delay w occurs with probability $Prob_k(w)$ given by the following equation:

$$\begin{aligned} Prob_k(w) \;=\; & \sum_{j=1}^{P-w} Prob_1(w+j) \\ & \cdot \sum_{o_2=1}^{j-1} Prob_2(j-o_2) \\ & \cdot \sum_{o_3=1}^{o_2-1} Prob_3(o_2-o_3) \cdot \ldots \\ & \cdot \sum_{o_{k-1}=1}^{o_{k-2}-1} Prob_{k-1}(o_{k-2}-o_{k-1}) \cdot p(o_{k-1}). \end{aligned} \tag{8}$$

Recursion in Equation (7) is generalized using probabilities defined in Equation (8) as follows:

$$Prob(w) = Prob_1(w) + \sum_{j=w+1}^{P} Prob(j) \cdot p(j-w). \tag{9}$$

To estimate the average delay W, first all $Prob_1(w)$ for $1 \leq w \leq P$ can be estimated using Equation (2). Then starting from $w = P$, all probabilities $Prob(w)$ for $1 \leq w \leq P$

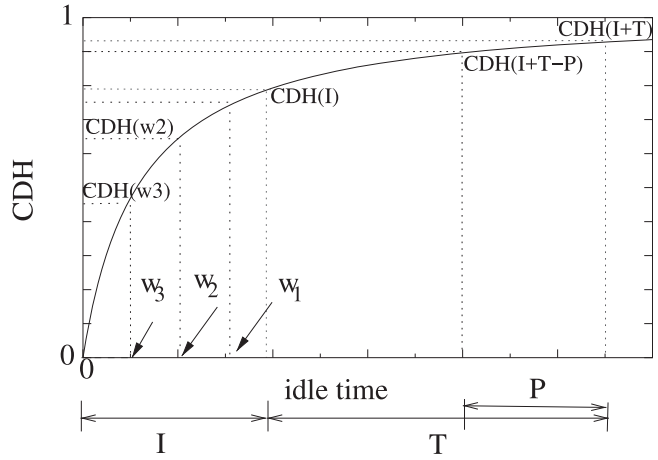

Fig. 3. Estimation of probabilities for propagation delay.

are computed using the recursion in Equation (9). Note that the granularity of the CDH bins determines the granularity of the recursion step. In the preceding presentation, we assumed, without loss of generality, that each bin is 1ms.

We stress that only $Prob_1(w)$ for $1 \leq w \leq P$ in Equation (2) depends on the scheduling pair (I, T). The rest depends on the probabilities of the monitored CDH of idle times (as depicted in Figure 3). This is important to the computational complexity of the framework because the majority of components in the recursion of Equation (9) are computed only once.

3.2. Meeting Performance Target *D*

Here, we develop the method to determine the pair (I, T) for scheduling the power saving modes such that performance does not degrade more than the target percentage D on the average. Because we want to control performance degradation, T, the time that the disk stays in a power saving mode includes the penalty P (i.e., $T > P$) for reactivating the disk and represents a proactive measure to control performance degradation.

To find the best scheduling pair (I, T), we scan the CDH of idle times for (I_l, T_j) pairs that would not violate the target D. Note that I_l and $I_l + T_j$ correspond to successive histogram bins. A pair (I_l, T_j) guarantees the performance target D if

$$D \geq \frac{W_{(I_l,T_j)}}{RT_{w/o\ power\ saving}}, \tag{10}$$

where $RT_{w/o\ power\ saving}$ is monitored and $W_{(I_l,T_j)}$ is computed using Equations (1) and (9).

If (I_l, T_j) satisfies the performance target D, then the corresponding "time in power savings" $S_{l,j}$ can also be computed. Because T_j includes P, for all idle intervals longer than $(I_l + T_j - P)$, the time in power saving is $(T_j - P)$. For all idle intervals with length i between I_l and $I_l + T_j - P$, the time in power saving $i - I_l$ becomes

$$S_{l,j} = \frac{\sum_{o=I_l}^{I_l+T_j-P} p(o) \cdot (o - I_l)}{E[idle]} + \frac{\sum_{o=I_l+T_j-P}^{max} p(o) \cdot (T - P)}{E[idle]}, \tag{11}$$

where *max* is the value of the last bin in the CDH and $E[idle]$ is the average idle interval length.

We choose the scheduling pair (I, T) to be the pair (I_l, T_j) that results in highest time in power saving $S_{l,j}$. Recall that the estimation of $S_{l,j}$ is done only for these pairs (I_l, T_j) that meet the performance degradation target D of Equation (10).

The computational complexity of the procedure to choose the scheduling pair (I, T) is $O(n^2)$, where n is the number of CDH bins. Note that the recursion for estimating W has a time complexity of $O(n)$.

3.3. Meeting Power Target *S*

In this scenario, the system quality target is the time in power savings S, which means that the scheduling pair (I, T) should achieve a time in power saving of at least $S\%$. The scheduling pair (I, T) should satisfy the targeted time in power saving S and degrade performance at the lowest possible minimum (i.e., in this scenario, there is no D defined). Note that if S is larger than the idleness in the system, then our procedure does not estimate an (I, T) pair, because power savings should not be scheduled when there are user requests outstanding.

Here, we need to find the scheduling pair (I, T) that meets the target S and causes the smallest performance degradation D. If every idle interval would be used for power saving, then S can be expressed as the time in power savings per idle interval $\overline{S}$ and would relate to the average idle interval length $E[idle]$ and utilization U according to the following equation:

$$\overline{S} = \frac{S \cdot E[idle]}{1 - U}. \tag{12}$$

However, for an (I_l, T_j) pair, only $(1 - \text{CDH}(I_l))$ idle intervals can be used for power savings. It follows that the target S can be met only if the time in power saving $T_j - P$ for the idle intervals to be used for power saving is such that if normalized over all idle intervals, then it is at least $\overline{S}$, as shown by the following equation:

$$\overline{S} = \frac{S \cdot E[idle]}{1 - U} \leq (T_j - P)(1 - \text{CDH}(I_l)). \tag{13}$$

All possible pairs (I_l, T_j) as defined by the bin values of the CDH of idle times are evaluated against Equation (13) (a scan that requires $O(n^2)$ steps). Those pairs (I_l, T_j) that satisfy Equation (13) meet the power saving target S. Among these pairs, we select the one with the smallest performance degradation D_{I_l,T_j} that is estimated according to Equation (10). The actual anticipated time in power savings for a pair (I_l, T_j) is S_{I_l,T_j} and is estimated using Equation (11).

3.4. Meeting Reliability Target *X*

The reliability target X is another quality target in our framework and is measured as the rate of power saving modes (measured usually at coarse granularity, e.g., 1 day) that the disk can have without impacting its lifetime. This rate is equal to the rate of spin-ups that a disk can tolerate without premature wear-out.

Let us denote utilization as $U = E[busy]/(E[idle] + E[busy])$, where $E[busy]$ is the average busy interval and $E[idle]$ is the average idle interval. Let us denote $\hat{X}$ as the rate of opportunities for power savings, and $\hat{X} = 1/(E[idle] + E[busy]) = U/E[busy] = (1 - U)/E[idle]$. If X is smaller than $\hat{X}$, then an idle interval should be used for power savings with probability $X/\hat{X}$. Otherwise, all idle intervals are to be utilized for power savings. Denote $\overline{X}$ as

$$\overline{X} = \begin{cases} \frac{X}{\hat{X}}, & \text{for } X < \hat{X} \\ 1, & \text{otherwise.} \end{cases} \tag{14}$$

Because a scheduling pair (I, T) uses only (1-CDH(I)) of idle intervals for power savings, the reliability target X is violated only if (1-CDH(I)) is larger than $\overline{X}$. In this case, fewer idle intervals than (1-CDH(I)) should be used for power savings. As a result, the delay W should reflect the potential fewer power saving modes and the resulting lower delay. For this, we redefine Equation (2) to reflect that the delay caused to the IOs of the *first* busy period following a power saving mode happens with probability $\overline{X}$/(1-CDH(I)). Note that if $\overline{X} > 1 - \text{CDH}(I)$, then no correction needs to take place, as X is not violated.

Reflecting the reliability target X in Equation (2) results in the following:

$$Prob_1(w) = C \cdot \begin{cases} \text{CDH}(I+T-w+1) - \text{CDH}(I+T-w), & \text{for } 1 \le w < P, \\ \text{CDH}(I+T-P) - \text{CDH}(I), & \text{for } w = P, \end{cases} \tag{15}$$

where C is defined as

$$C = \begin{cases} \frac{\overline{X}}{1-\text{CDH}(I)}, & \text{for } \overline{X} < 1 - \text{CDH}(I) \\ 1, & \text{otherwise.} \end{cases} \tag{16}$$

Using Equation (15) to estimate the first delay ensures that the average delay W is estimated accurately based on Equation (1) and the recursion of Equation (9). As a result, the framework meets both reliability and performance targets. The reliability target is reflected similarly in the estimation of power savings achieved by a scheduling pair (I, T). Equation (11) is updated to account for the reliability target as follows:

$$S_{l,j} = C \cdot \frac{\sum_{o=I_l}^{I_l+T_j-P} p(o) \cdot (o - I_l)}{E[idle]} + C \cdot \frac{\sum_{o=I_l+T_j-P}^{max} p(o) \cdot (T - P)}{E[idle]}, \tag{17}$$

where C is defined in Equation (16). By using these improved formulas, we can achieve the reliability target.

3.5. Correlation-Based Enhancement

So far, the scheduling pair (I, T) is computed by heavily using the CDH of idle times. As a result, the decisions are made on the probability of an idle interval length assuming that the sequence of idle intervals is a renewal process. However, the utilization of idle time would improve further if the length of idle intervals were predicted more accurately than by using only the marginal distribution (i.e., CDH). Here, we show how to exploit any existing short-term correlation in idle interval lengths.

For this, we define the category of *long idle intervals* as all idle intervals longer than L, where L is defined such that idle intervals of at least length L are observed at a rate close to the reliability target X. We compute online, similar to the CDH of idle times, the probabilities that two consecutive idle intervals, up to G lags apart, are both long. We denote these probabilities as $Prob(LL_l)$ (i.e., two idle intervals of at least length L that are l lags apart).

The lag l with the highest $Prob(LL_l)$ is selected for prediction. Although any $Prob(LL_l)$ value can be used in the framework, only $Prob(LL_l)$ above 0.5 is recommended for a good power savings effect because when $Prob(LL_l)$ is above 0.5, the correlation structure is considered strong and yields a good prediction accuracy. Therefore, once a long idle interval is observed, the upcoming idle interval l lags in the future are also to be used for power savings. This correlation-based prediction is used to enhance the performance of our framework *in addition* to the regularly estimated scheduling pair (I, T).

We argue that if a long idle interval is predicted, then the probability of causing a delay is less than when the regular probabilities in the CDH are used. As a result, we

propose using a shorter I and a longer T without violating the performance target D. Specifically, we denote the scheduling pair that results from such prediction as (I_L, T_L), where I_L is defined such that $\text{CDH}(I_L) = 0.5$ and T_L is defined such that it corresponds to the length of the long idle interval L (i.e., $T_L = L - I_L$). Although we define L such that the occurrence of idle intervals of at least length L is at most X, it is expected that for most enterprise workloads, the number of idle intervals of length at least equal to L should be less than X within a specified time period. For this reason, we generate two scheduling pairs, (I, T) and (I_L, T_L), where the first one is estimated as a regular scheduling pair using the CDH of idle times and is used to "fill up" the quota X left unused by the second pair.

The most important characteristic in our framework is the ability to accurately estimate performance of a scheduling pair (I, T). In the case when two scheduling pairs are used, we combine the estimations of delay W and power savings S for both scheduling pairs. We define

$$W = (1 - Y) \cdot W_L + Y \cdot W_R, \;\; S = (1 - Y) \cdot S_L + Y \cdot S_R, \tag{18}$$

where W_R and S_R are the delay and power savings yield by the regular scheduling pair (I, T), and W_L and S_L are the delay and power savings yield by the predictive scheduling pair (I_L, T_L). The coefficient Y captures the portion of X that is contributed by (I, T). This coefficient is zero if the probability of having long idle intervals is larger than the allowance $A(X)$. We define Y as

$$Y = \begin{cases} A(X) - (1 - \text{CDH}(L)), & \text{for } A(X) > 1 - \text{CDH}(L) \\ 0, & \text{otherwise.} \end{cases} \tag{19}$$

Although W_R and S_R are defined in the previous sections, we need to define W_L and S_L. From the conditional probability $Prob(LL_l)$, we know that we need to have $Prob(LL_l)$ true positives in prediction of idle intervals longer than L and $1 - Prob(LL_l)$ false positives (i.e., the predicted long idle interval is in fact shorter than L). Because this prediction occurs only if a long idle interval is observed, with probability $1 - \text{CDH}(L)$, the (I_L, T_L) scheduling pair causes a power saving mode with probability $(1 - \text{CDH}(L))(1 - \text{CDH}(I_L))$. This means that a delay P is caused with probability $(1 - \text{CDH}(L))(1 - \text{CDH}(I_L))(1 - Prob(LL_l))$, whereas the savings of $T_L - P$ units of time occur with probability $(1 - \text{CDH}(L))(1 - \text{CDH}(I_L))Prob(LL_l)$. We have

$$\begin{aligned} Prob(P)_L &= (1 - \text{CDH}(L)) \cdot (1 - \text{CDH}(I_L)) \cdot (1 - Prob(LL_l)), \\ S_L &= (1 - \text{CDH}(L)) \cdot (1 - \text{CDH}(I_L)) \cdot Prob(LL_l)(L - I_L - P), \end{aligned} \tag{20}$$

where $Prob(P)_L$ is used as the basis for the recursion to compute W_L as given by Equations (9), (10), and (15).

4. EXPERIMENTAL EVALUATION

In this section, we evaluate PREFiguRE with regard to accuracy, robustness, flexibility, and adaptivity in estimating schedules for power saving modes while meeting system quality targets, including the performance slowdown target D, the reliability target X, and the power savings target S. One of the most important aspects of PREFiguRE is making decisions based *only* on metrics that are monitored in real time and does not depend on static models or knowledge of the underlying disk drive characteristics. As a result, for the evaluation of PREFiguRE, we use trace-driven simulations as long as they allow for the calculation of the PREFiguRE input parameters like the histogram of idle times. Recall that PREFiguRE does not interfere with disk request service or scheduling; as a result, we do not need a full-disk simulator. PREFiguRE is computationally lightweight, as it only scans the CDH of idle times, which is at most a

Table III. General Trace Characteristics

Trace: Entries	Util (%)	Idle Length Mean (in milliseconds)	Idle Length CV	Potential Time in Power Savings (%) Lev. 3	Potential Time in Power Savings (%) Lev. 4
Code 1: 379,490	5.6	192.6	8.4	55	48
Code 2: 56,631	0.5	1,681.6	2.3	92	87
Code 3: 286,612	4.8	233.95	22.5	66	55
Code 4: 18,865	0.1	8,293.67	7.8	97	94
File 1: 135,629	1.7	767.5	2.3	70	53
File 2: 44,607	0.7	2,000.2	3.8	94	90
File 3: 44,607	0.1	2,046.51	9.1	87	79
File 4: 14,160	0.1	2,615.74	11.3	95	92

Note: All traces have a duration of 12 hours.

few thousand entries, at a frequency of every few hours. PREFiguRE computes a nearly optimal (as our experiments show) scheduling pair almost instantaneously. In this section, we show the proximity of the scheduling pair (I, T) given by PREFiguRE to the *optimal* pair that is found by exhaustive search (i.e., by simulating and evaluating *all* possible pairs for scheduling power saving modes). In addition, we show how one could use workload patterns in the time series of idle intervals to further improve on power savings without deviating from the preset reliability and performance constraints.

4.1. Performance of PREFiguRE

Our evaluation is driven by a set of disk-level enterprise traces collected at mid-size enterprise storage systems hosting dedicated server applications, such as a development server ("Code") and a file server ("File") [Riska and Riedel 2006]. Each trace corresponds to a single drive in a RAID. For an unbiased treatment, we focus on the performance requirement of each disk. We monitor the workload of each disk drive and determine whether to put it to sleep or not. Storage systems that deploy advanced redundancy schemes may schedule a request such that it avoids the disks that are in power saving modes. However, our method is orthogonal to such solutions, as we monitor the disk workload *after* those policies have been applied. In addition, our framework works with a lower priority compared to the upper-level policies. Therefore, our framework can be applied at individual storage nodes (e.g., single disk drive) without interfering with upper-level power saving policies.

The traces are collected at the disk level and measured using a SCSI or IDE analyzer that intercepts the IO bus electrical signals and stores them. The final traces are produced by decoding the electrical signals. This trace collection method does not require modifying the software stack of the targeted system and does not affect system performance. We stress that our framework only requires knowledge of idleness and is completely independent of the complexity of the arrival and service processes, as well as complex scheduling behavior in the various levels of the storage stack (e.g., the RAID setup). More importantly, they record the arrival and departure time of each disk-level request, allowing for exact calculation of the histogram of idle times.

The traces that we use to evaluate PREFiguRE have varying characteristics (Table III provides an overview). From this table, we notice that these traces are characterized by very low utilization, yet their idleness is highly fragmented. Notice the differences in the mean idle intervals and their coefficients of variation (CVs). The columns labeled "Time in Power Savings" include the percentage of time relative to the duration of the entire trace that is used for power savings if *all* idle intervals that can be used for Level 3 or Level 4 savings are indeed used, and if *perfect* knowledge of

future workload is available. This is of course not practical, but this value represents an absolute upper bound on power savings. The table shows that the eight traces are quite diverse and thus constitute an excellent set to evaluate PREFiguRE's ability to estimate the best scheduling pair (I, T) for any workload. We stress that our traces are measured in enterprise systems with idle intervals that yield power savings only for Levels 3 and 4, whose penalty P is up to 1s, but not Levels 5 and 6, whose penalty P is several seconds. Consequently, we do not show results from Levels 5 and 6 of power saving modes and do not discuss wear-out because of spin-ups/downs. The reliability aspect of power savings is evaluated in association with load/unload cycles that occur when Levels 3 and 4 of power saving modes apply on a disk drive.

We use the first half of each trace as the "training period" during which we construct the CDH of idle times and determine other monitored metrics. PREFiguRE computes the scheduling pair (I, T) using the metrics collected during the training period using the analytic methodology presented in Section 3. The second half of each trace is used as the "testing period," during which we run a simulation that uses the computed (I, T) pair to schedule power saving modes. The testing period validates the accuracy of the PREFiguRE scheduling decision. Specifically in the trace-driven simulation, the power saving modes are activated only after I idle time units elapse. The disk remains in a power saving mode for at most T time units. A new IO arrival *always* preempts a power savings mode and reactivates the disk drive, which takes P units of time.

Table IV gives an overview of the effectiveness of PREFiguRE. All columns labeled "Estim." represent values estimated by PREFiguRE, and the ones labeled "Actual" are obtained via trace-driven simulation. The "Target D" column is the performance target input to PREFiguRE. Performance target D is not violated if columns labeled "Performance Degradation" are less than or equal to "Target D." Finally, S_{max} corresponds to the optimal value found by exhaustive search of all possible (I, T) pairs to identify the one that offers best savings with performance degradation equal to or under the target D. The penalty to reactivate the drive is set to $P = 500$ms (Level 3) [Seagate Technology 2012; Hitachi Global Storage Technologies 2007]. The reliability target X is set to 200 for Level 3 or Level 4 power saving modes per day [Kim and Suk 2007], assuming a lifetime of 4 years.

The main observations from this table are the following:

—The performance D is *never* violated by the scheduling pair computed by PREFiguRE, as validated by multiple simulation experiments.
—PREFiguRE consistently estimates excellent scheduling parameters for maximum power saving while limiting the number of load/unloads per day.
—The time in power savings $S_{(I,T)}$ estimated analytically by PREFiguRE is accurate most of the time; see its proximity to the actual values given by simulation. The errors come from two sources: first, the estimation method relies on past information to predict the future. Consequently, its accuracy depends on the change in workload characteristics used by the framework between future and past. Second, the estimation method is a statistical approach that relies on the granularity and accuracy of characterization measurements (e.g., finer granularity of CDH of idle periods yields better prediction accuracy than coarse granularity).
—High accuracy of PREFiguRE and its ability to estimate the scheduling outcome in the form of $D_{(I,T)}$ and $S_{(I,T)}$ is critical because it suggests that PREFiguRE can be used to drive analysis in the system.
—Monitoring of metrics in the short past ("training period" of several hours) yields good and robust predictions for the near future ("testing period" of several hours).
—For $D > 5\%$, the accuracy of estimations is consistently high. For $D = 1\%$, the accuracy reduces, as it becomes difficult for PREFiguRE to capture the very small

Table IV. Power Savings and Performance Degradation Estimated Using PREFiguRE (Columns "Estim.") and Simulation (Columns "Actual.")

"Code 1"						"Code 3"				
	Performance Degradation		Time in Power Saving		Max Time in Power Saving	Performance Degradation		Time in Power Saving		Max Time in Power Saving
Target D	Estim.	Actual	Estim.	Actual	S_{max}	Estim.	Actual	Estim.	Actual	S_{max}
1	1	0.0	1.68	1.37	2.06	1	0.0	12.54	10.87	17.24
5	3	0.0	2.22	1.94	2.06	2	0.0	15.93	11.69	17.99
10	3	0.0	2.22	1.94	2.06	2	0.0	15.93	11.69	17.24
20	3	0.0	2.22	1.94	2.06	2	0.0	15.93	11.69	17.99
100	3	0.0	2.22	1.94	2.06	2	0.0	15.93	11.69	17.99
"Code 2"						**"Code 4"**				
	Performance Degradation		Time in Power Saving		Max Time in Power Saving	Performance Degradation		Time in Power Saving		Max Time in Power Saving
Target D	Estim.	Actual	Estim.	Actual	S_{max}	Estim.	Actual	Estim.	Actual	S_{max}
1	1	0.0	0.09	0.09	0.33	1.0	1.0	8.18	4.99	12.57
5	5	0.0	0.28	0.32	0.33	4.0	1.0	13.68	8.03	13.07
10	10	2.0	0.29	0.33	0.33	9.0	3.0	21.47	18.89	18.89
20	20	20.0	0.31	0.35	0.35	20.0	10.0	35.73	35.35	35.35
100	22	21.0	0.31	0.35	0.37	31.0	25.0	37.79	37.51	37.57
"File 1"						**"File 3"**				
	Performance Degradation		Time in Power Saving		Max Time in Power Saving	Performance Degradation		Time in Power Saving		Max Time in Power Saving
Target D	Estim.	Actual	Estim.	Actual	S_{max}	Estim.	Actual	Estim.	Actual	S_{max}
1	1.00	0.00	0.50	0.39	0.39	1.00	0.00	2.69	1.77	5.76
5	5.00	3.00	0.73	0.69	0.70	4.00	2.00	6.32	4.42	5.76
10	7.00	4.00	0.75	0.71	0.71	10.00	4.00	8.47	6.98	6.98
20	7.00	4.00	0.73	0.71	0.71	20.00	6.00	12.02	10.79	10.80
100	7.00	4.00	0.73	0.71	0.71	28.00	21.00	13.45	11.17	11.17
"File 2"						**"File 4"**				
	Performance Degradation		Time in Power Saving		Max Time in Power Saving	Performance Degradation		Time in Power Saving		Max Time in Power Saving
Target D	Estim.	Actual	Estim.	Actual	S_{max}	Estim.	Actual	Estim.	Actual	S_{max}
1	1.00	0.00	0.31	0.30	0.87	1.00	1.00	0.44	0.36	2.60
5	5.00	5.00	1.59	1.37	1.55	4.00	3.00	11.78	8.75	8.75
10	9.00	6.00	1.90	1.69	1.87	8.00	4.00	14.67	12.70	12.70
20	19.00	10.00	1.92	1.72	1.75	19.00	19.00	17.38	15.86	15.86
100	18.00	12.00	1.92	1.72	1.75	44.00	44.00	27.08	26.33	26.34

Note: Level 3 savings are used. All values are in percentages (%). For example, for the columns of time, it means the percentage of timecompared to the entire trace duration.

variations in performance. Recall that estimation of delays is the most critical aspect of the framework, and its accuracy depends on the CDH bin granularity. As a result, discrepancies become noticeable for very small performance targets, such as $D = 1\%$.

A phenomenon worth discussing is that PREFiguRE estimates for various target D's are the same for "Code 1" and "Code 3." This happens because PREFiguRE calculates the same (I, T) pair for $D \geq 5\%$. The CDHs of "Code 1" and "Code 3" reveal that these two workloads have many small idle intervals but only a few long ones. Indeed, 95% of "Code 1" idle intervals are smaller than the Level 3 penalty (500ms), and thus they are excluded from PREFiguRE as a scheduling choice. As a consequence, a large idle waiting time I is used to prevent small idle intervals from being used for power savings. Therefore, W in Equation (10) is small and results in the same D that is always less

Table V. Various What-If Scenarios That Can Be Answered Using the Estimation Engine in PREFiguRE to Assist with Making Power Saving Decisions in a Storage System

What-If Question	"Code 1"	"Code 2"	"File 1"	"File 2"
How much should I slow down the user traffic to get power savings of 10%?	33.0%	59.0%	195.0%	27.0%
How much should I slow down the user traffic to get power savings of 20%?	61.0%	104.0%	458.0%	140.0%
How much power savings do I get if I slow the user traffic with 10%?	1.94%	0.33%	0.71%	1.69%
How much power savings do I get if I slow the user traffic with 20%?	1.94%	0.35%	0.71%	1.72%
Which power saving level should I use, Level 3 or Level 4, if I slow the user traffic with 10%?	Level 3	Level 3	Level 4	Level 3
Which power saving level should I use, Level 3 or Level 4, to get power savings 10%?	Level 3	Level 3	Level 3	Level 3
If I relax the X condition for the next 12 hours and slow the user traffic with 10%, how much additional savings will I get and by how much is X violated?	6.59% (50)	3.36% (19)	0.73% (23)	8.15% (285)

than the target Ds we set in Table IV. This results in selecting the same (I, T) pairs for $D \geq 5\%$.

Overall, the table shows that PREFiguRE is robust across all workloads and ranges of performance targets, with excellent accuracy for both power and average delay estimation, without compromising on the reliability constraint X. This makes the case that PREFiguRE can be also used very effectively in analysis to select among power saving options, as shown in the following section.

4.2. "What-If" Analysis

In system design and online resource management, it is critical to be able to know the outcome of features and enable them *only* if beneficial. Specifically, because power savings in disk drives impact both performance and reliability, the disk should be put into power saving modes only if the savings are significant for the system. Because of its analytic core, PREFiguRE has the ability to compute schedules and estimate their outcome. As such, it facilitates the automation of online decisions on disk power savings by giving answers to a wide range of "what-if" questions.

Table V lists a set of what-if questions that could be answered using the PREFiguRE framework. The table shows how PREFiguRE predicts for a given workload if a specific power saving target can be met. For example, in a cluster with the four disks (and workloads), a target of 10% time in power saving can be achieved by Code 1 with a performance degradation of 33.0%.

Similarly, the system can also estimate beforehand if it is worth increasing the performance target D for higher power savings. In this table, we can clearly see that it is not beneficial to increase the performance degradation to 20%, as it does not offer additional savings for any of the workloads in the four disks in the storage cluster. It is obvious in this table that for most workloads when the penalty due to power savings is low (i.e., Level 3), the power savings are better. Finally, we can estimate beforehand whether it is worth relaxing the reliability condition to achieve better power savings. The last what-if scenario presented in this table indicates the power savings when we relax the reliability target X. Given that X captures the wear-out effect that power savings have on disk drives over their lifetime, X can be set higher at times and lower at other times. Specifically, for "Code 2," the savings are considerable and the compromise in reliability is small compared to the original reliability constraint. The system may

decide to relax X for that disk for a while and account for it at a later time when the workload has changed and savings are limited.

4.3. PREFiguRE's Adaptivity and Estimation Capabilities

PREFiguRE is a framework that monitors current workload and updates its scheduling decisions (i.e., the (I, T) pair) accordingly. So far in this section, the learning (or training) has occurred for 6 hours and the computed (I, T) pair is used for the following 6 hours. However, there are various ways to learn the workload characteristics (i.e., the histogram of idle times) and update the corresponding scheduling parameters. Here, we evaluate the robustness of PREFiguRE against the length of the learning window and the granularity of updates in the computed (I, T) pair.

We experiment with two additional learning window sizes (i.e., 3 and 5 hours), and scheduling parameters update every half hour or only at the end of a learning window. Specifically, we evaluate the following variations in learning a CDH:

—*Learning1*: Learning windows are nonoverlapping, and (I, T) is computed only at the end of a learning window.
—*Learning2*: CDH of idle times is accumulated from the beginning, and (I, T) is computed every half hour.
—*Learning3*: The learning window slides with half an hour of granularity, and (I, T) is computed every half hour.
—*Baseline*: This is similar to Learning1, but the CDH is built with the knowledge of idleness in the current learning period, not the previous one. It is included only for comparative purposes as a best case.

We present the results from our trace-driven simulation in Figure 4, where the left column plots the performance degradation in the system validating the accuracy of the framework with regard to a performance slowdown target of $D\%$. The right column of plots in Figure 4 captures the power savings resulting from the scheduling framework.

It is clear that different learning methods and granularity achieve different accuracy. We observe that it is important to learn over longer rather than shorter periods of time (compare the first row of results corresponding to 5-hour learning to the second row of results corresponding to 3-hour learning in Figure 4). Another important observation is that updating the (I, T) pair every half hour yields better robustness to the learning window size changing than updating it less frequently, as it can reduce the impact caused by the changing workload. Recall that the computation complexity of computing the (I, T) pair is minimal, and a frequency of every half hour that we suggest here is expected to have an equally minimal impact on the overall system performance.

4.4. Comparison with Common Practice Methods

The efficiency of PREFiguRE is shown by comparing its performance to common practices used for power savings in storage systems. The most widespread approach is to idle wait for a fixed amount of time before putting a disk into a power saving mode. Usually the fixed amount of time is set to be a multiple of the penalty P to bring back the disk into operational state. Here, we show results obtained when the idle wait I is set to $2P$ [Eggert and Touch 2005]. A second approach is to guide power savings by the current utilization levels in the storage node (i.e., disk drive). Here, we apply the first approach of fixed idle wait only if the utilization in the last 10 minutes is below a predefined threshold (set to the average utilization in the trace).

In Figure 5,we plot the performance degradation and power saving results of PREFiguRE and the preceding two common practice methods. For PREFiguRE, three performance targets (i.e., 10%, 50%, and 100%) are evaluated. For the two common practice methods, the performance target cannot be set beforehand, and the slowdown may be

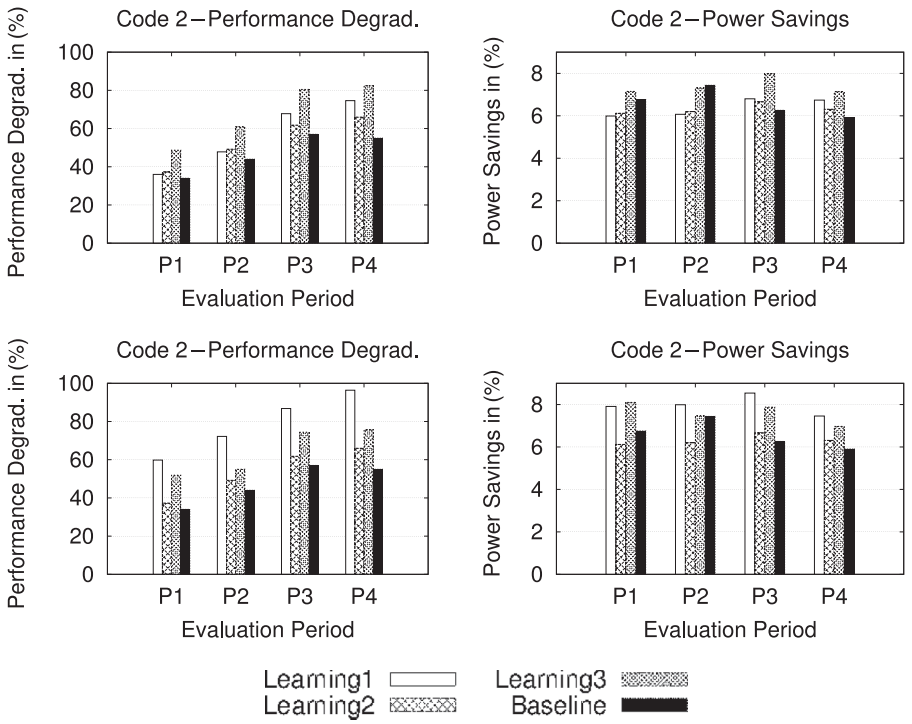

Fig. 4. Performance degradation and time in power savings over time for Code 2, three different learning methods, and two different lengths of learning (the first row of plots corresponds to 5 hours of learning and the second row to 3 hours of learning). The performance degradation target is 50%. P1 is the evaluation period that starts at the fourth hour, P2 starts at the fifth hour, P3 starts at the sixth hour, and P4 starts at the seventh hour. For fair comparison, the evaluation lasts for 5 hours in each evaluation period for both learning length cases.

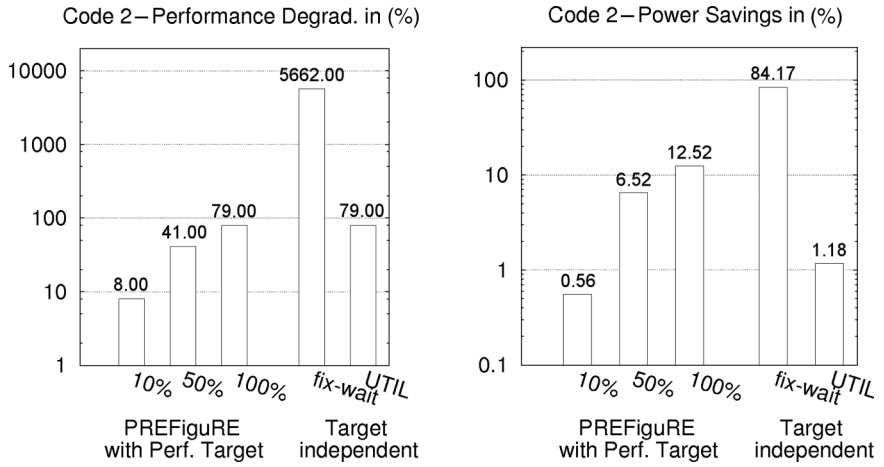

Fig. 5. Performance degradation and time in power savings for Code 2 under PREFiguRE and other common practices (i.e., fixed idle wait and utilization guided). Because the y-axis is in log scale, the y-axis values are shown for each bar.

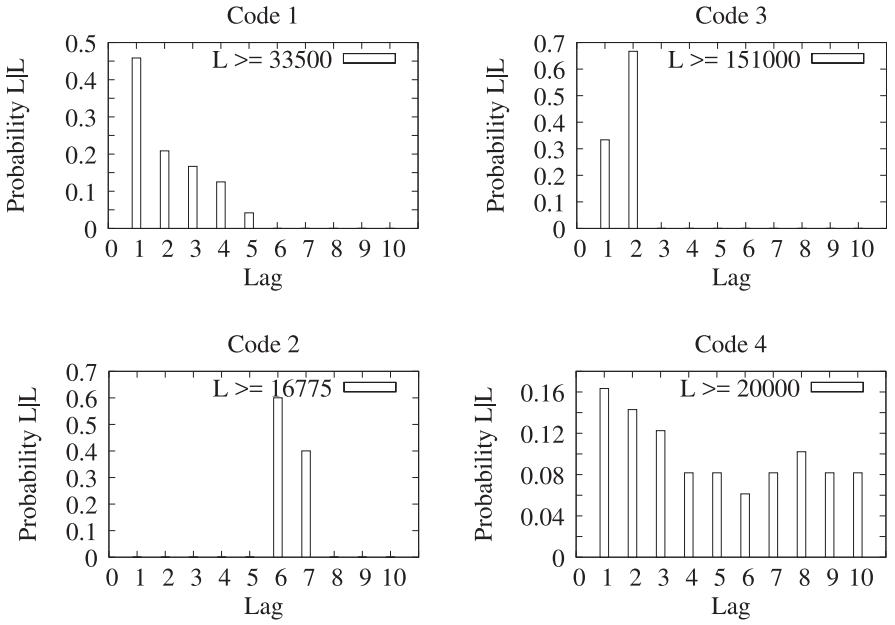

Fig. 6. Conditional probability values of a long interval being followed by another long with k lags apart. The long interval length L is defined in the legend.

unbounded. Often in practice, to limit the performance slowdown, the fixed idle wait and/or the utilization threshold are set such that the system goes into power savings only occasionally.

In Figure 5, the y-axis is in log scale, and the absolute values are shown above each bar. The fixed idle wait method for $I = 2P$ results in a slowdown of 5,662% (i.e., several orders of magnitude more than PREFiguRE for less than 10 times the power savings). The utilization-guided method reduces performance degradation of the fixed idle wait method, but its power savings are 10 times lower than PREFiguRE for similar performance slowdowns.

The results in Figure 5 clearly illustrate that PREFiguRE outperforms common practice methods. By taking into consideration the idleness, which in a way confines in a compact measure the complex interaction of the arrival and service processes, PREFiguRE meets performance targets while achieving high power savings.

4.5. Correlation-Based Enhancement: PREFiguRE-LL

To further extend power savings without violating the performance degradation target, we enhance PREFiguRE with the predictive capabilities of the conditional probabilities of successive idle intervals (Figure 6). We construct conditional probabilities of two idle intervals up to $G = 10$ lags apart being at least of length L, where L represents long idle intervals observed in the system such that the number of such intervals is close to X (i.e., the reliability target). The length L of long idle intervals depends on the workload characteristics (i.e., the average, maximum, and variability in the distribution of idle intervals as captured by the CDH), which means that for more idle workloads, this value is higher than for the busier ones.

In our evaluation of this enhancement, which we call PREFiguRE-LL, we focus on workloads "Code 1," "Code 2," "Code 3," and "Code 4." Figure 6 shows the probability that successive idle intervals of at most 10 lags apart are at least of length L. We

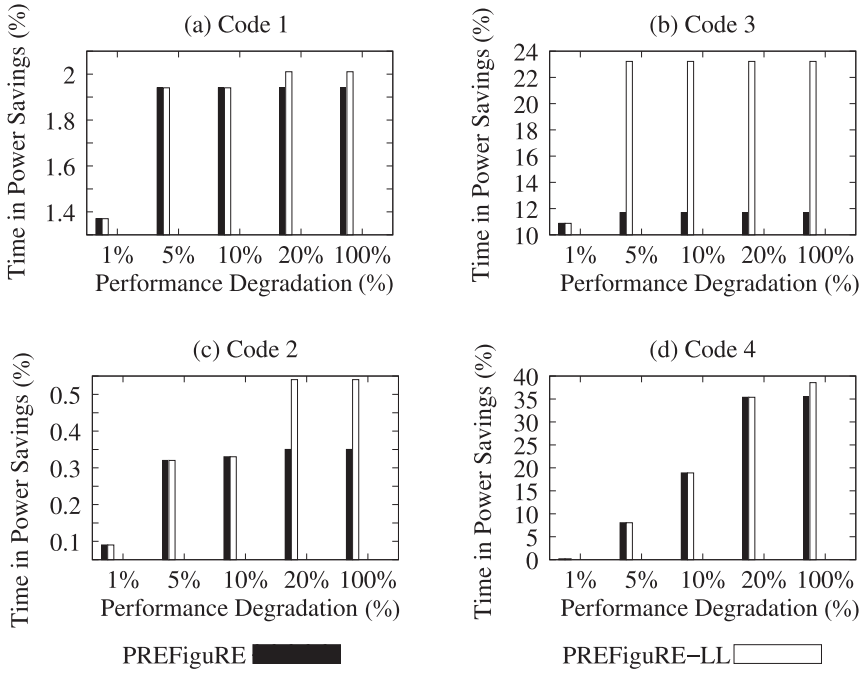

Fig. 7. Power savings (Level 3) for performance degradation targets 1%, 5%, 10%, 20%, and 100% for "Code 1," "Code 2," "Code 3," and "Code 4."

observe that for the "Code 2" and "Code 3" workloads, these conditional probabilities are higher than 0.5 for at least one lag. This suggests that the enhanced PREFiguRE-LL could benefit from the prediction capabilities embedded in these probabilities and harvest these long idle intervals to extend power savings according to the discussion in Section 3.5. For "Code 1" and "Code 4," the conditional probabilities have small values, and therefore PREFiguRE-LL is expected to not increase power savings. However, we stress that in these cases, PREFiguRE-LL reduces seamlessly to PREFiguRE and still meets both reliability and performance targets. For PREFiguRE-LL, we pick among all 10 evaluated lags the one with the highest conditional probability to predict the future long idle interval and define the scheduling parameters as explained in Section 3.5.

Power savings with PREFiguRE and PREFiguRE-LL are shown in Figure 7, whereas the corresponding performance degradations are given in Table VI. Consistently with the expectations set from the probability values in Figure 6, we observe that PREFiguRE-LL extends power savings for "Code 2" and "Code 3" workloads. The high correlation between successive long idle intervals enables PREFiguRE-LL to start early and stay longer in a power saving mode and almost double the overall power savings for several of the performance targets D. For "Code 1" and "Code 4," however, such information does not exist, and as expected, PREFiguRE-LL performs the same as PREFiguRE. As supported by the results in Figure 7, the gains of as much as double in power savings come at *no* cost in performance degradation. PREFiguRE-LL does not violate the performance target D for the entire spectrum of evaluated slowdowns. Note that there are cases when higher performance degradation targets are set, but the actual performance degradation and power savings stay the same. This is because of the reliability targets in the framework. In addition, the policy remains robust, as stochastic information on the sequence of idle times is incorporated in the framework.

Table VI. Actual Performance Degradation under PREFiguRE and PREFiguRE-LL for Level 3 Savings

	"Code 1"		**"Code 2"**	
	Performance Degradation		Performance Degradation	
Target D	PREFiguRE	PREFiguRE-LL	PREFiguRE	PREFiguRE-LL
1	0.0	0.0	0.0	0.0
5	0.0	3.0	0.0	0.0
10	0.0	3.0	2.0	2.0
20	0.0	3.0	20.0	17.0
100	0.0	3.0	21.0	17.0
	"Code 3"		**"Code 4"**	
	Performance Degradation		Performance Degradation	
Target D	PREFiguRE	PREFiguRE-LL	PREFiguRE	PREFiguRE-LL
1	0.0	0.0	1.0	0.0
5	0.0	2.0	1.0	1.0
10	0.0	2.0	3.0	3.0
20	0.0	2.0	10.0	12.0
100	0.0	2.0	25.0	25.0

Note: Values are in percentages.

4.6. Caveats and Limitations

The interplay between the device driver decisions and upper-level policies is an important factor to consider when implementing PREFiguRE. When the framework is implemented at lower levels (e.g., at the device driver level), the interplay between device driver decisions and higher-level scheduling is less likely to happen, as the lower levels are transparent to upper levels. For example, during the periods that the disk is in power saving mode, upper-level policies see that the disk is available and idle. We propose PREFiguRE to be implemented at the lower levels (i.e., at the storage controller or HDDs rather than other levels of the IO hierarchy to avoid the potential interference with upper-level non-FCFS schedulers). If PREFiguRE needs to be deployed in the same level as other non-FCFS schedulers, interference is likely to happen, but such interference is usually harmless for performance. This is because for the non-FCFS disk scheduler, the more the requests to choose from, the better the performance. Therefore, the actual extra delays caused by the waking-up process become even smaller than the values we estimate, as we consider the propagated delay affected up to k consecutive busy periods in our estimation (see Equation (8)). In addition, there are many activities that occur in the path that add variability on measurements more than what we are adding by controlling the sleep times. Sources of variability on higher-level scheduling include multiple interrupts of the communication protocols to communicate with each other: application-OS-client-driver-RAID controller-SAS/PCIcable-HDD. Sources of the HDD variability could be missed rotation, failure in seeks, and failure in rotation. All of these happen regularly and only increase latency at the HDD level. They happen regularly, but for a small portion and are easily discarded in other scheduling decisions. We leave the rigorous estimations (via measurement data and/or simulation) of the exact indirect effect as our future work.

5. RELATED WORK

For the past two decades, there has been a host of work on power efficiency in computer systems and more specifically in disk drives. In general, there are two types of power saving techniques: proactive and reactive. The proactive techniques usually allow users to use less storage to achieve the power saving purpose (e.g., through compression [Kothiyal et al. 2009], deduplication [Costa et al. 2011], and thin provisioning [Edwards

et al. 2008]). Such techniques usually can be applied only to specific applications or content and sometimes are difficult to deploy due to the lack of communication between the system and applications. There are also several works focusing on modeling power saving modes as low-priority tasks [Gandhi et al. 2009; Jaiswal 1968; Gaver 1959, 1962; Keilson 1962], but they do not provide guarantees on power, performance, and reliability. A comprehensive comparison of power saving algorithms for disk drives on personal computers is presented in Douglis et al. [1994]. The algorithms are evaluated based on trace-driven simulation for two known disk drive models. The baseline used for power savings assumes a priori knowledge of the idle interval duration. The compared algorithms vary based on when and for how long a disk is placed in power savings. A fundamental difference with the work presented here is that these algorithms apply to personal, not enterprise, systems, and therefore no power, performance, or reliability guarantees are provided.

In Garg et al. [2009] a Markov model of a cluster of disks is used to predict disk idleness and schedule the spin-down of disks for power savings. This model is based on two states—ON and OFF—and a prediction mechanism that relies on a probability matrix. Simulations using DiskSim with synthetic (exponential and Pareto) and real workloads show that the Markov model has 87.5% prediction accuracy, reduces energy by 35.5%, performs better than other multispeed models, and has a performance penalty that is negligible (less than 1%). Another analytical model is introduced in Greenawalt [1994] and is applied to predict the idle interval duration to spin down a disk for power savings. The Poisson assumption used in this work is questionable, especially given the bursty nature and correlation between interarrivals in real traces [Riska and Riedel 2006]. For this analytical model, a "critical rate" is defined as the number of accesses per unit time at which it is more power efficient to leave the disk active than to spin it down. The preceding models are useful for offline disk spin-down policies but not for anticipating workload changes on the fly that are necessary for the development of online algorithms.

An adaptive algorithm based on the idea of "sessions" is presented in Lu and Micheli [1999]. A session is similar to a busy period. Different sessions are separated by intervals of inactivity of duration τ. The inactivity period is defined by monitoring and adaptation of the algorithm (i.e., increase or decrease τ based on inactivity periods characteristics). The algorithm does not minimize energy consumption compared to other adaptive algorithms, but it reduces power while preserving performance and reliability. However, no specific guarantees are given for the performance and reliability of this power saving algorithm.

A dynamic power management (DPM) algorithm is introduced in Irani et al. [2003] that extends the power savings states from idle and busy to multiple power saving states based on a stochastic optimization. This algorithm has the best power savings (i.e., 25% less) and best performance (i.e., 40% less) compared to other DPM algorithms. It is based on online observations and learning of the probabilistic length of an interval.

The effects of power management on disk request latency for personal computers are studied in Ramanathan et al. [2000]. The authors find the upper bound of IO request latencies to demonstrate the worst-case scenario and how to handle it with efficient system design. A simple adaptive power management algorithm is presented that predicts the duration of the next idle period based on the previous one. Immediate shutdown of disks is studied, and the authors conclude that even though it increases power savings, it also has a negative impact by increasing latency.

A large amount of literature in conserving power in disk drives proposes techniques that alter the way the storage system workload is served such that the work done at a subset of disks is reduced and power is conserved. The work on massive array of idle disks (MAIDs) [Colarelli and Grunwald 2002] uses caches to redirect the workload

to spin down disks. The applicability of MAIDs is limited to back-up or archival storage systems. RIMAC [Yao and Wang 2006] uses a two-level IO cache in addition to the redundancy of RAID5 to reduce disk spin-ups of standby disks and improve on power savings and overall IO response time. Energy-efficient RAID (EERAID) [Li and Wang 2004] achieves power savings while keeping performance degradation under 10%. However, EERAID does not address the reliability concern associated with the additional disk spin-ups. Similar to RIMAC, EERAID requires additional hardware to be efficient. The redundancy in RAID is exploited also in Pinheiro et al. [2006], where power saving is achieved by powering down lightly loaded redundant disks in a RAID setting. Hibernator [Zhu et al. 2005] is another framework that addresses power savings in a storage system setting. In Hibernator, the workload is redirected to active disks that are dynamically deployed with different rotation speeds. Redundancy in storage systems is exploited further in the FS2 framework [Huang et al. 2005], which saves power and enhances performance by serving IOs from the closest replica of data. This scheme reorganizes data by exploiting free blocks based on access patterns.

WRITE offloading [Narayanan et al. 2008] is proposed as a workload shaping technique. It extends idleness in a disk drive by offloading the WRITE traffic elsewhere in the storage system. This method is effective in extending power savings for WRITE-intensive workloads. Similarly, SRCMap [Verma et al. 2010] is a workload shaping technique that uses an energy versus workload intensity proportionality model to determine which disks in the system can be used for power savings and which to serve IO. SRCMap develops an intelligent replication scheme that aims at serving a workload with the optimal number of active disks in the system. Their model is based on the observation that the power drain for a workload increases linearly as the load intensity increases. SRCMap addresses reliability by offloading READs as well as WRITEs to increase the idle period duration. As a consequence, fewer spin-downs are needed for power saving, as few but large idle intervals are created. Both Narayanan et al. [2008] and Verma et al. [2010] use fixed idle waiting periods (in the order of minutes) to limit performance degradation, albeit no guarantees are given on performance degradation because of power savings.

Although the preceding works represent efforts to combine caches and redundancy for power savings in storage systems, Zhu et al. [2004] propose a new power-aware cache retention policy that achieves as much as 16% power savings compared to LRU by sending fewer requests down to the disk. However, any performance impact of the new caching policy is not addressed. DRPM [Gurumurthi et al. 2003] represents a technique that saves power in a disk drive by dynamically changing the rotation speed of disk drives. Although DRPM addresses performance and reliability concerns, its main drawback is on the substantial hardware changes that need to be done in disk drives to support multiple *active* rotation speeds.

Different from all of the preceding works, in Mountrouidou et al. [2011] the authors introduce disk drive reliability as an additional target of the power savings algorithm. The work [Yan et al. 2012] focuses on how to estimate the performance impact of power saving by taking into consideration the propagation delay effects.

The theoretical contribution of PREFiguRE lies in that it provides *both* performance and reliability guarantees by unifying the work presented in Mountrouidou et al. [2011] and Yan et al. [2012], and by improving the online analytic estimations in a way that promotes efficiency. Furthermore, PREFiguRE significantly improves accuracy and robustness via a new probabilistic model that further improves its underlying analytic framework. The correlation-based enhancement of PREFiguRE introduced in this article leads to more power savings under reliability and performance guarantees. Finally, the problem is reversed to achieve specific power savings while keeping

performance degradation at acceptable levels. PREFiguRE nearly instantaneously yet accurately estimates power savings while *always* meeting performance targets and strictly following reliability targets in the form of rigid numbers of spin-ups/downs within a time period. PREFiguRE consistently achieves nearly excellent power savings by judiciously selecting nearly optimal scheduling parameters across very different and challenging workloads.

To the best of our knowledge, PREFiguRE is unique in its focus of simultaneously aiming at maximizing a specific primary target while meeting predefined secondary measures within strict, preset reliability constraints. Any comparison with any of the power saving policies that have been proposed in the literature would not be possible due to this unique characteristic: no other policy offers such performance or reliability guarantees, making potential comparisons uneven or possibly unfair.

6. CONCLUSIONS

We have presented a compact analytic model and its integration into an algorithmic framework that provides given performance and reliability targets, as well as answers to the following difficult questions: "when" and for "how long" idle periods in disk drives can be utilized for putting the system in a specific power saving mode such that the targets are met. A detailed analytic model is also developed that quite precisely determines the respective amount of power savings that is possible to save. The effectiveness of the proposed heuristics of PREFiguRE are demonstrated using a set of traces from enterprise storage systems. PREFiguRE can also be used forworkloads with dynamic idle period distribution if combined with workload forecasting techniques, which we leave as our future work.

REFERENCES

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

L. B. Costa, S. Al-Kiswany, R. V. Lopes, and M. Ripeanu. 2011. Assessing data deduplication trade-offs from an energy and performance perspective. In *Proceedings of the 2011 International Green Computing Conference and Workshops (IGCC'11)*. 1–6. DOI:http://dx.doi.org/10.1109/IGCC.2011.6008567

Fred Douglis, P. Krishnan, and Brian Marsh. 1994. Thwarting the power-hungry disk. In *Proceedings of the 1994 Winter USENIX Conference*. 293–306.

John K. Edwards, Daniel Ellard, Craig Everhart, Robert Fair, Eric Hamilton, Andy Kahn, Arkady Kanevsky, James Lentini, Ashish Prakash, Keith A. Smith, and Edward Zayas. 2008. FlexVol: Flexible, efficient file volume virtualization in WAFL. In *Proceedings of the USENIX 2008 Annual Technical Conference on Annual Technical Conference (ATC'08)*. 129–142.

L. Eggert and J. D. Touch. 2005. Idletime scheduling with preemption intervals. In *Proceedings of the 20th ACM Symposium on Operating Systems Principles (SOSP'05)*. 249–262.

Anshul Gandhi, Mor Harchol-Balter, Rajarshi Das, and Charles Lefurgy. 2009. Optimal power allocation in server farms. *ACM SIGMETRICS Performance Evaluation Review* 37, 1, 157–168.

Rajat Garg, Seung Woo Son, Mahmut T. Kandemir, Padma Raghavan, and Ramya Prabhakar. 2009. Markov model based disk power management for data intensive workloads. In *Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'09)*. 76–83.

D. P. Gaver Jr. 1962. A waiting line with interrupted service, including priorities. *Journal of the Royal Statistical Society. Series B (Methodological)* 24, 73–90.

Donald P. Gaver Jr. 1959. Imbedded Markov chain analysis of a waiting-line process in continuous time. *Annals of Mathematical Statistics* 30, 3, 698–720.

Paul M. Greenawalt. 1994. Modeling power management for hard disks. In *Proceedings of the 2nd International Workshop on Modeling, Analysis, and Simulation on Computer and Telecommunication Systems (MASCOTS'94)*. 62–66.

Laura Grupp, John Davis, and Steven Swanson. 2012. The bleak future of NAND flash memory. In *Proceedings of the USENIX Conference on File and Storage Technologies*.

S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Franke. 2003. DRPM: Dynamic speed control for power management in server class disks. In *Proceedings of the Annual International Symposium on Computer Architecture (ISCA'03)*. 169–180.

Hitachi Global Storage Technologies 2007. Power and Acoustics Management.

H. Huang, W. Hung, and K. G. Shin. 2005. FS2: Dynamic data replication in free disk space for improving disk performance and energy consumption. In *Proceedings of the 20th ACM Symposium on Operating Systems Principles (SOSP'05)*, Vol. 39. 263–276.

Sandy Irani, Sandeep Shukla, and Rajesh Gupta. 2003. Online strategies for dynamic power management in systems with multiple power-saving states. *ACM Transactions in Embedded Computing Systems* 2, 325–346.

Narendra Kumar Jaiswal. 1968. *Priority Queues*. Elsevier.

Julian Keilson. 1962. Queues subject to service interruption. *Annals of Mathematical Statistics* 33, 4, 1314–1322.

Patricia Kim and Mike Suk. 2007. Ramp Load/Unload Technology in Hard Disk Drives. Available at https://www.hgst.com/sites/default/files/resources/LoadUnload_white_paper_FINAL.pdf.

Rachita Kothiyal, Vasily Tarasov, Priya Sehgal, and Erez Zadok. 2009. Energy and performance evaluation of lossless file data compression on server systems. In *Proceedings of the Israeli Experimental Systems Conference (SYSTOR'09)*. ACM, New York, NY, Article No. 4. DOI: http://dx.doi.org/10.1145/1534530.1534536

D. Li and J. Wang. 2004. EERAID: Energy efficient redundant and inexpensive disk array. In *Proceedings of the 11th ACM SIGOPS European Workshop*.

Kester Li, Roger Kumpf, Paul Horton, and Thomas Anderson. 1994. A quantitative analysis of disk drive power management in portable computers. In *Proceedings of the USENIX Winter 1994 Technical Conference*. 22–36.

Yung-Hsiang Lu and Giovanni De Micheli. 1999. Adaptive hard disk power management on personal computers. In *Proceedings of the IEEE Great Lakes Symposium*. 50–53.

X. Mountrouidou, A. Riska, and E. Smirni. 2011. Saving power without compromising disk drive reliability. In *Proceedings of the Workshop on Energy Consumption and Reliability of Storage Systems*.

D. Narayanan, A. Donnelly, and A. I. T. Rowstron. 2008. Write off-loading: Practical power management for enterprise storage. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST'08)*. 253–267.

Dushyanth Narayanan, Eno Thereska, Austin Donnelly, Sameh Elnikety, and Antony Rowstron. 2009. Migrating server storage to SSDs: Analysis of tradeoffs. In *Proceedings of ACM EuroSys Conference*. 145–158.

E. Pinheiro, R. Bianchini, and C. Dubnicki. 2006. Exploiting redundancy to conserve energy in storage systems. In *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'06/Performance'06)*. 15–26.

Dinesh Ramanathan, Sandy Irani, and Rajesh Gupta. 2000. Latency effects of system level power management algorithms. In *Proceedings of the 2000 IEEE/ACM International Conference on Computer-Aided Design (ICCAD'00)*. 350–356.

A. Riska and E. Riedel. 2006. Disk drive level workload characterization. In *Proceedings of the USENIX Annual Technical Conference*. 97–103.

A. Riska and E. Smirni. 2010. Autonomic exploration of trade-offs between power and performance in disk drives. In *Proceedings of the 7th IEEE/ACM International Conference on Autonomic Computing and Communications (ICAC'10)*. 131–140.

Seagate Technology 2012. Constellation ES Product Overview: High Capacity Storage Designed for Seamless Enterprise Integration. Available at http://www.seagate.com.

Seagate Technology 2014. Seagate Enterprise Capacity 3.5 HDD v4 Serial ATA Product Manual. Available at http://www.seagate.com.

Anil Vasudeva 2011. Are SSDs Ready for Enterprise Storage Systems? Available at http://www.snia.org/.

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

Feng Yan, Xenia Mountrouidou, Alma Riska, and Evgenia Smirni. 2012. Quantitative estimation of the performance delay with propagation effects in disk power savings. In *Proceedings of the 2012 USENIX HotPower Workshop*.

X. Yao and J. Wang. 2006. RIMAC: A redundancy-based, hierarchical I/O architecture for energy-efficient storage systems. In *Proceedings of the 1st ACM EuroSys Conference*. 249–262.

Q. Zhu, Z. Chen, L. Tan, Y. Zhou, K. Keeton, and J. Wilkes. 2005. Hibernator: Helping disk arrays sleep through the winter. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP'05)*. 177–190.

Q. Zhu, F. M. David, C. F. Devaraj, Z. Li, and Y. Zhou. 2004. Reducing energy consumption of disk storage using power-aware cache management. In *Proceedings of the International Symposium on High-Performance Computer Architecture (HPCA'04)*. 118–129.

Received December 2014; revised November 2015; accepted December 2015