Adaptive Workload Shaping for Power Savings on Disk Drives

Xenia Mountrouidou, Alma Riska, Evgenia Smirni
College of William and Mary
Williamsburg, VA 23187, USA
xmount,riska,esmirni@cs.wm.edu

ABSTRACT

In order to reduce the amount of power consumption in data centers, it is becoming necessary to shut off or slow down disks that are not actively serving user requests. In addition to exploiting disk drive idleness, system features are in place that shape a disk's workload by redirecting portions of it elsewhere, with the goal to expand the periods of idleness and the potential for power savings. In this paper, we propose several workload shaping techniques that determine which part of the working set to copy elsewhere using temporal and spatial access frequencies in the workload. These workload shaping techniques, used within an analytic estimation methodology, enable a fully automated framework that determines on-line for the current workload which, if any, shaping technique to activate such that the power saving benefits are maximized without violating performance targets. Extensive trace-driven evaluation shows that the proposed workload shaping techniques complement each-other with regard to their abilities to enhance idleness in disk drives for a wide range of workload characteristics. This results to added power savings in a data center even when performance targets are stringent and workloads intensive.

Categories and Subject Descriptors

C.4.1 [**Performance of Systems**]: Design studies; I.5.1.a [**Computing Methodologies**]: Pattern Recognition—*Models*[Statistical]; I.6.4 [**Computing Methodologies**]: Simulation and Modeling—*Model Validation and Analysis*

General Terms

Performance, Algorithms, Design

Keywords

workload shaping, disk drive, power saving, performance, idleness, histogram

ICPE'11, March 14–16, 2011, Karlsruhe, Germany.

1. INTRODUCTION

As data centers grow bigger and central to the IT operations of many organizations, automation and adaptivity becomes imperative to their efficient operation [3]. Beyond that, power consumption in a data center is added to the set of metrics of quality such as performance, reliability, and availability.

While data center power consumption amounts to about 20% of all IT power consumption, as much as 20-40% of data center power consumption is attributed to disk drives [15]. As data grows, so does the number of deployed disk drives, which can reach up to tens of thousands. Because not all the data in a data center is accessed simultaneously, many disks can be idle at any given time. Harvesting disk idle times to reduce power consumption without violating performance targets is the focus of this paper.

To aid with disk drive power consumption, manufacturers are designing disks that can operate in an array of modes with different power consumption characteristics [5, 13]. In addition, system designers aim at serving the incoming workload with the minimum amount of resources, i.e., disk arrays and servers, by bringing resources up on-demand. Most policies that reduce the amount of active storage devices in a system redirect the workload from one set of disk drives to another with the goal of increasing the pool of idle disk drives that can be shut off or slowed down [1, 8, 19, 16]. The reasoning is that although disk drives are generally underutilized [10], idle times are highly fragmented (i.e., idle periods are interrupted by short busy periods) which greatly reduces power saving capabilities. The expectation is that shaping the workload via redirection of a portion of it enables additional opportunities for power savings in disk drives.

The efficiency of workload shaping techniques available in the literature and of those proposed in this paper depends on workload characteristics. Write offloading, where all the incoming writes are redirected elsewhere in real time [8], can be very effective *only* if writes dominate the workload, but is ineffective in a read-dominated workload. Utilizing standard caching policies such as LRU can shape disk workload based on its spatial and temporal characteristics [19] but its effectiveness depends on the working set size, since caches for such purposes tend to be small. In a cluster setting, shifting the entire working set from one disk to another can free up targeted disks but its efficiency depends on the stationarity of working sets [16].

The purpose of this paper is to propose workload shaping techniques whose behavior is determined from the characteristics of the workload observed in the system such as frequency of individual or group of accesses. To this end,

we characterize the observed idle times and busy periods at individual disk drives to determine the portion of the workload that if redirected elsewhere in the system would yield the most power savings without violating performance targets. In this work we strive to increase both the length and the number of large idle intervals by consolidating multiple idle intervals together.

To meet the above goal we extract the temporal and spatial characteristics of the observed workload in the form of histograms of idle intervals and of most frequent disk accesses. In addition, we monitor how consecutive idle times and consecutive busy periods are related and collect probabilities that can assist to identify sets of requests that fragment large idle periods. All the above information is easy to collect and store, thus contributing to a lightweight solution.

Running actively a workload shaping policy in a system whose workload currently does not benefit from it (e.g., write-offloading in a read-mostly workload) is not desirable, because workload shaping techniques generally impose additional work in the system in the form of replicating or moving data. As a result, it becomes critical to activate a workload shaping policy only if a desirable trade-off between power gains and performance degradation caused by the disk power saving modes is possible.

Here, we propose an estimation framework that identifies accurately and efficiently which workload shaping policy (from a pre-defined set) would perform best (e.g., saves the most power without violating performance targets) for the current workload. This estimation methodology is lightweight, because it computes power savings of alternative workload shaping methods based on an analytic expression, thus it evaluates the trade-offs between different workload shaping policies instantaneously and selects the one, if any, that best serves the workload needs.

Extensive experimentation using four workload traces from enterprise systems confirms that the proposed workload shaping techniques succeed at exploiting workload spatial and temporal locality, in order to increase periods of idleness and power savings in disk drives. Temporal locality in workloads, even moderate, consistently leverages power savings by achieving to expand idleness and reduce its fragmentation. Workloads that exhibit both temporal and spatial locality in their IO accesses are excellent candidates for power savings.

Although the workload shaping techniques that we propose are adaptive by nature, their efficiency changes as workload characteristics change. To maximize power savings, the system should alternate between the different workload shaping policies such that the most effective one is active at any given time. Our results show that our analytic estimation framework is the right tool to achieve such full automation of power saving modes in disk drives.

This paper is organized as follows. Section 2 gives an overview of the power saving modes that apply to disk drives and describes an analytic framework used to determine the schedule of power saving modes as well as to assess whether a shaping technique is worth adapting. Then Section 3 presents the workload characterization methodology that guides and motivates the workload shaping techniques in this paper. In Section 4, we propose three workload shaping methods. Results that show the effectiveness of the shaping techniques are included in Section 5. A fully automated framework for power saving in disk drives is laid out in Section 6. In Section 7 we review related work. We conclude in Section 8.

2. BACKGROUND

In this section we give an overview of power saving modes on disk drives that convey the challenges of the problem and motivate the need for workload shaping for power savings. We also give a brief overview of an algorithmic framework for power savings in disk drives that is introduced in [11]. This framework is used here to determine the scheduling parameters for the power saving modes. Furthermore, this power saving framework is used to estimate analytically the effectiveness of the workload shaping techniques, i.e., the amount of power saving for a given performance degradation, as discussed in more details in Section 6.

2.1 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 when compared with 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 do 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.

Operation Mode	Description	Power savings	Penalty (sec)
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

Table 1: Power saving modes in a disk drive and the corresponding re-activation penalty.

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, differ between disk drive families and manufacturers. Table 1 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 two disk drive manufacturers [13, 5].

Note that during the process of bringing up 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 1, this power surge during reactivation depends on drive family and manufacturer.

2.2 Scheduling Power Saving Modes in Disk Drives

The discussion of the previous section obviates the need to account for the performance penalty before deciding on a disk operation mode for power savings. In this section, we give an overview of an algorithmic framework that determines when and for how long to schedule the power saving modes in disk drives such that the trade-off between performance degradation and power savings satisfies system quality targets. For more details on the framework we direct the interested reader to [11].

One could argue that putting the disk into an idle mode immediately after any idleness is detected could maximize power savings, but 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. If there are no arrivals for the first I idle time units of an idle interval, then the system is put into idle mode till the next IO arrival. The algorithmic framework in [11] deviates from this common practice by limiting the amount of time a disk drive stays in a power saving mode to T, in an effort to control the performance degradation caused by the power saving modes.

The basic premise of the algorithmic framework in [11] is to specify "when" (i.e., I) and for "how long", (i.e., T) an idle disk drive should go onto power saving mode, while keeping performance degradation to low levels and maximizing power savings. The algorithmic framework calculates the I and T values as a function of past workload (i.e., the stochastic characteristics of past idle times) and user specifications (i.e., the average performance degradation of future jobs that may be tolerated) in order to maximize power savings.

Central to the calculation of I and T is the cumulative distribution histogram (CDH) of idle intervals that allows for a compact representation of the empirical distribution of the lengths of idle times. This CDH is created by dividing the range of the idle interval lengths into equal-sized "bins". Then, the number of observed idle intervals that fall into each bin is calculated and the frequency of an interval of a specific size is obtained. The CDH value C_j that corresponds to a bin j, indicates the probability that an idle interval has length less than a value t_j, i.e., $C_j = Pr(idle\ interval \leq t_j)$. In addition to the CDH of idle intervals, the framework uses the user-provided average performance degradation D, which is defined as the average relative delay of an IO operation due to power savings. Usually, multiple (I, T) pairs exist for a specific D target but the framework chooses the pair that maximizes the overall amount of time in power savings S.

The algorithm is based on two facts: (1) only idle intervals longer than I would be utilized for power saving, and (2) only busy periods that succeed idle intervals of length between I and $I + T$ are to be delayed due to power saving modes. Consequently, an important measure for the algorithm is E that corresponds to the probability of an idle interval shorter than $I + T$ but longer than I, see Figure 1. E is pivotal for calculating the best I and T values such that delays do not exceed a target performance degradation D and the effective time in power saving S is maximized.

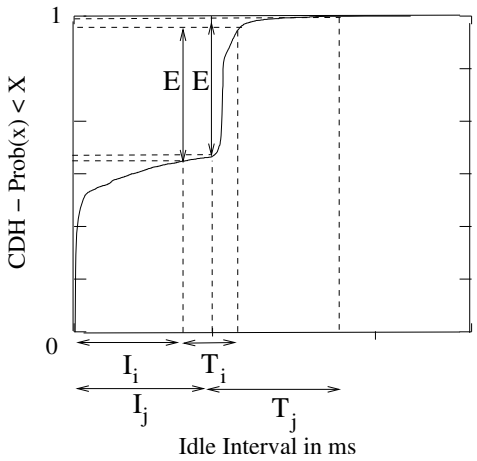

Figure 1: Selecting the scheduling (I, T) pair for the power saving modes in a disk drive whose idleness is captured by the CDH.

Having the CDH of idle times as the main data structure, gives the algorithm the ability to seamlessly and accurately detect opportunities for power saving (i.e., long T). To illustrate this, consider the (I_i, T_i) and (I_j, T_j) pairs in the CDH presented in Figure 1. These two pairs correspond to the same E value, but while I_i and I_j are very close, T_j is much longer than T_i. This feature is a result of the sharp knee of the CDH in the figure, which is indicative of the existence of many long idle intervals of similar length. This property of idle intervals is desirable when it comes to power savings because it means that the behavior of idle intervals of that length is more predictable. As a result, by setting I such that the system goes into power savings only for the idle intervals with the commonly observed length (as captured by the knee) power savings will be higher for less performance degradation. We revisit this observation and what it means for the proposed shaping policies in Section 6.

One of the main features of this algorithmic framework that we capitalize on in this paper is the ability to estimate

1. initialize
  a. $D \leftarrow$ predefined performance degradation target
  b. $E \leftarrow$ portion of idle intervals that may delay IOs without violating D
  c. the penalty P for disk power up
  d. the CDH of idle times (t_j, C_j)
  e. $S_{max} = 0$, $I_{max} = 0$, $T_{max} = 0$
2. for all bins i, j in CDH do
  find (I_i, T_j) such that $CDH(I_i + T_j) - CDH(I_i) = E$
  estimate power savings S for (I_i, T_j)
  if $S \geq S_{max}$ then $S_{max} = S$, $I_{max} = I_i$, $T_{max} = T_j$
3. report final (I_{max}, T_{max})

Figure 2: Scheduling framework algorithm with target performance degradation D.

the effective time in power saving S for a given workload (as captured by the CDH of idle times) and a scheduling pair (I, T). For this, we define the penalty P of a power saving mode as a function of the time the disk needs to become active and the power surge during this process. Consequently, the effective time in a power saving mode is $(T - P)$ every time the disk goes into one of them. The CDH is used to estimate the expected overall effective time in power saving S.

Figure 2 gives an overview of the algorithmic framework that scans the CDH in search of the appropriate I and T values.

3. WORKLOAD CHARACTERIZATION

We use a set of traces measured at the disk level of two enterprise storage systems, an application development server ("Code") and a file server ("File") [10]. These traces record for each request that reaches the disk drive the following metrics: the arrival time, the departure time, the type of the request (i.e., READ or WRITE), the request length in bytes, and the location on the disk. The traces provide the highest level of detail with regard to the length of idle intervals for power savings, because busy periods and idle intervals are captured *exactly*. As we show later in this section, these traces constitute a rich set of workloads that are representative of a wide spectrum of disk workload characteristics.

Table 2 presents a high level characterization of the traces with some metrics pertinent to power savings. The first two columns of the table show the duration of each trace (about 12 hours) and the drive utilization that ranges from 0.5% to 5.6%. The very low utilizations immediately suggest that these traces offer great opportunities for power savings, but close evaluation shows that these traces are highly fragmented, therefore the time that can be used for power savings is a lot less than suggested by the average utilizations [11]. This can be also seen by the mean and standard deviation of idle times. The reported values show large variability in idle times and relatively small idle intervals, motivating the need to devise effective methods that aim at increasing the length of idle times and enable power savings.

The table reports on the trace working set, which here is defined as the total size of different blocks accessed by IO requests in the trace. The table also includes the portion of the working set relative to the total drive capacity. Essentially, the working set shows the range of addresses used in a trace. Except for Code 2, all traces have *working sets that span almost the entire disk capacity*. With regard to workload shaping, this would mean that redirecting the majority of accesses elsewhere would need to mirror *almost* the entire disk. This is not effective in a storage system and it is one of the main reasons why we propose here workload shaping techniques that are based on frequencies of accesses, as well as their spatial and temporal locality characteristics as further explained in Section 4.

3.1 Spatial Locality

Spatial locality is an important characteristic of an IO workload that if present should improve the efficiency of workload shaping. However, most enterprise IO workloads are known to have only moderate spatial locality. Here, in order to strike a balance between book-keeping and workload detail, we define as an "access block" 100 MB of successive IO addresses. Because all traces are READ intensive, we first examine the spatial locality of READs. In addition, we also examine the spatial locality of busy periods, i.e., all IO operations that occur between two idle periods.

For both READs and busy periods we construct the empirical cumulative distribution histograms that capture their respective frequencies. Each bin corresponds to a READ block of 100 MB. The number of accesses to the particular block is placed to the corresponding bin. In a similar way, we construct the CDH of the busy period accesses. This time instead of READ blocks we group a set of blocks to each bin, that are accessed in one busy period.

These two new histograms provide important information about the workload's spatial locality. If a high frequency of accesses concentrates on few READ blocks or busy periods, then it is reasonable to assume that if these high-frequency blocks are moved elsewhere, then idleness can increase. In addition, provided that these histograms are changed online, changes in the workload working set can be easily detected.

Table 3 gives a snapshot of the information provided by the CDH by presenting the portion of the working set that serves the most frequent READ blocks of the workload. The number of READ blocks that corresponds to a particular percentage of the working set is also reported within the parentheses. The table clearly shows that the frequency of READ accesses that corresponds to a particular working set percentage varies significantly for different workloads. Observe that READs for workloads "Code 1", "File 1", and "File 2" correspond to moderate portions of the working set. As we show in the results section, removing READs from such workloads cannot help in achieving high power savings. On the other hand, "Code 2" appears to have good READ locality, since 70% READ load can be removed by redirecting only 5% of the working set. Redirecting only this small part of the working set may be very beneficial for power.

Table 4 presents the portion of workload captured by the most frequent Busy Period accesses that is served by portions of the working set equal to 1%, 5%, 10%, and 20%. For "Code 2", it is remarkable that 92% of the busy period accesses corresponds to only 1% of the working set. This suggests that redirecting more than 1% of the working set cannot be beneficial. The workload that is second highest in spatial locality of busy periods is "File 2" but not nearly as high as "Code 2". The other two workloads exhibit only moderate spatial locality in their busy periods. In general,

Trace	Length of Trace (hrs)	Util. (%)	Idle Length		Disk Capacity (GB)	Working Set (GB)	Working Set (%)
			Mean	Std. Dev.			
Code 1	11.8	5.6	284.02 ms	2287.02 ms	146	135.3	92.7%
Code 2	11.8	0.5	1614.37 ms	3793.50 ms	146	36.2	24.7%
File 1	11.5	1.7	874.99 ms	1559.49 ms	73	51.1	70.0%
File 2	11.5	0.7	2686.85 ms	9066.78 ms	73	60.7	83.2%

Table 2: High level trace characteristics. The working set is measured in GB (column 7) as well as percentage of total disk capacity (column 8).

%WS	1%	5%	10%	20%
Code 1	0.12(13)	0.38(65)	0.56(130)	0.75(260)
Code 2	0.57(4)	0.70(20)	0.77(40)	0.86(80)
File 1	0.12(5)	0.38(25)	0.57(50)	0.81(100)
File 2	0.18(6)	0.43(30)	0.56(60)	0.71(120)

Table 3: Portion of the most frequent READ accesses served by a specific portion of the working set for all traces The number of READ blocks is also given in parentheses.

%WS	1%	5%	10%	20%
Code 1	0.15(14)	0.40(65)	0.52(135)	0.60(260)
Code 2	0.92(10)	0.96(35)	0.97(74)	0.98(177)
File 1	0.06(6)	0.24(35)	0.38(84)	0.62(331)
File 2	0.25(6)	0.58(49)	0.65(106)	0.69(167)

Table 4: Portion of the most frequent Busy Periods served by a specific portion of the working set for all traces. The number of corresponding busy periods is given in parentheses.

the CDH of the busy periods can significantly help in deciding whether there is high spatial locality in the workload. If this property exists, then it may be possible to increase idleness by moving only a small part of the workload.

3.2 Temporal Locality

We now turn to temporal locality in the workload by focusing on the sequence of idle intervals. The purpose here is to explore temporal characteristics such that we are able to first detect (and later predict) whether large idle periods are followed by other large ones. This is essential, because by removing all busy periods between these idle periods, we can significantly enlarge idle intervals.

To characterize temporal locality we use the conditional probability of a long idle interval of size greater or equal to L to be followed by another long interval greater or equal to L. This information is useful not just for successive idle intervals but also for intervals that are separated by k successive busy periods, i.e., that are k lags apart. An important issue here is to identify what consists a "large" idle interval. In this paper, we consider as large, an interval that is 2 to 4 deviations from the mean[1]. Because our purpose is to reduce the fragmentation of idle times by increasing idle intervals, we use these probabilities to guide us whether the busy periods within idle intervals are to be removed.

Figure 3 shows the conditional probability of a long idle interval that is preceded by another long one. The value (in ms) that specifies the long idle interval of size L for each workload is included in the legend. The figure allows us to visualize any structure and/or patterns in the times series of idle intervals. "Code 2" is clearly a code that depicts a distinctive characteristic: the conditional probability is equal to 0.9 for the sixth lag for a long interval of size 11,000 ms. This means that the sixth interval that follows a long one, is highly likely to be long. However, all other lags of "Code 2" are significantly smaller, this suggests low to moderate temporal locality for this workload. "File 1" shows high probability values for a few lags (i.e., for the first three lags) that sum up a value higher than 0.5%, therefore it is beneficial to attempt to remove a *sequence* of two to three busy periods to reduce the fragmentation of idle intervals.

"Code 1" and "File 2" have smaller values of conditional probabilities for successive lags but they remain consistent for multiple lags suggesting a persisting temporal locality that also can be beneficial with regard to workload shaping.

Finally, another way to describe temporal locality is by evaluating the persistence of the entire working set as time elapses. The adaptive workload shaping policy proposed in this paper is built on the ground of the similarity of working set in different time windows. Table 5 shows how much of the working set remains the same in four time windows of three hours each. All workloads have a highly repetitive working set, where more than 75% of it is repeated in the next window. Since the most frequently accessed blocks repeat, addressing the most frequently accessed blocks with workload shaping should be effective. Another important aspect of this characteristic is that past observations serve as good predictors of the future workload, which consequently would result in accurate decisions from our estimation framework.

% same WS	Part 1-2	Part 2-3	Part 3-4
Code 1	0.63	0.51	0.55
Code 2	0.83	0.83	0.91
File 1	0.82	0.79	0.77
File 2	0.87	0.76	0.81

Table 5: Percentage of repeating working set (WS) in 3 hour window.

[1]What constitutes a large idle interval depends on the workload variability and skewness. We have experimented with different values of L for the four workloads and have seen that a good rule of thumb for the definition of a "large" interval tends to be in the range of 2 to 4 deviations from the mean. Our experiments show that the exact value of what is a "large" interval does not seem to matter, as long as it is within this range.

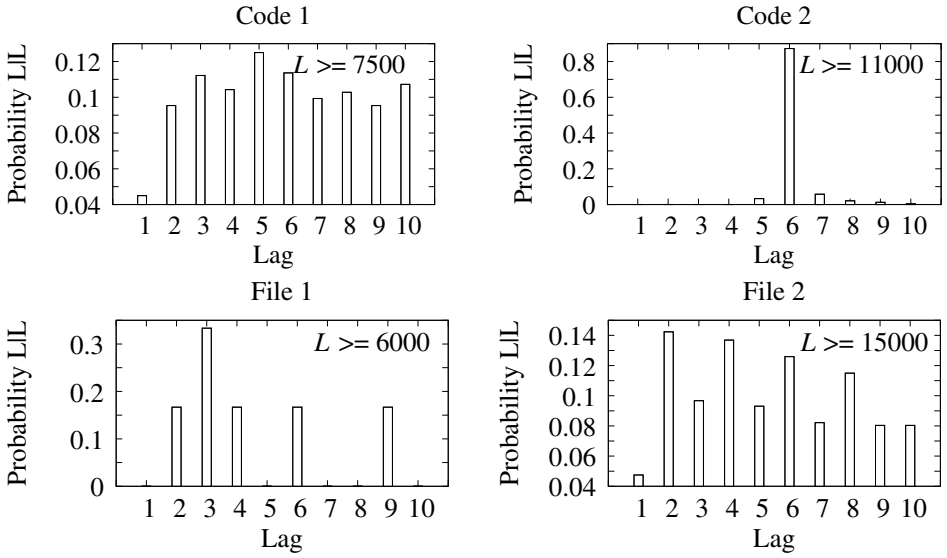

Figure 3: Conditional probability values of a long idle interval being followed by long ($Prob[L|L_{k^{th}\ lag}]$) for all traces as a function of $lag\ k$. A long interval L is defined in the legend for each figure.

4. WORKLOAD SHAPING TECHNIQUES

This section focuses on how to use information on the spatial and temporal localities to design effective workload shaping strategies. A common property of the three techniques that we propose is that we assume the use of a "buffer" to redirect part of the workload. This buffer can be another disk drive or other persistent storage device in the storage system. The goal of the workload shaping techniques proposed here is to redirect only a small portion of the disk capacity elsewhere in the system. Specifically, we focus on buffer sizes that range from 1 GB to 10 GB only, which represent at most 15% of the disk capacities of the traces in Section 3. This restriction is useful because we aim at workload shaping techniques that are lightweight and do not impose significant overhead in the system.

Our framework uses a compact way of storing the workload information that is used for assessing the proposed workload shaping techniques. First we identify in a trace the time-stamps that mark the start and the end of each busy period. Then we apply this information to specify the blocks that correspond to a busy period, in order to build structures for logging access frequencies (i.e., READs) and busy periods frequencies. We use these structures to determine the spatial locality of a workload, i.e., the portion of the workload and its corresponding disk capacity. To assess the temporal locality we use the time-stamps that mark the start and end of a busy period as well, only this time we log the idle interval between two subsequent busy periods and build the conditional probabilities that a long idle interval L is followed by another long one. Finally, we use the spatial and temporal locality information to specify the blocks that should be mirrored elsewhere in the system. We then build the "potential" CDH of this "shaped" workload, as it would become after redirecting these specific blocks and estimate its "potential" power savings. This process repeats every several (4-8) hours so that the CDH of idle times and the frequencies of READs and busy periods reflect changes in the workload.

Based on the above characterization and data structures, we propose the following workload shaping techniques:

READ Offloading: For this shaping technique we evaluate the frequencies of READs and the most active READs are removed until the pre-defined-size buffer is filled. The effectiveness of this technique depends on how READ-intensive is the workload and also on how strong is the spatial locality of the workload. Naturally, larger buffers would store more blocks and potentially more IOs would be redirected there, but as we mention above, we strive for small buffers.

Busy Period (BP) Offloading: Here, the most frequently accessed busy periods (as given by the histogram of busy periods) are removed until the buffer is filled. Spatial locality and buffer size are decisive for the effectiveness of this shaping method. An important difference with READ offloading is that by removing whole busy periods, idle intervals are concatenated with their neighboring intervals.

Probabilistic Offloading: This shaping method removes a group of busy periods among two long idle intervals, aiming at increasing the likelihood that long intervals are concatenated to even longer ones. Deciding what constitutes a long idle interval depends on the observed mean and standard deviation of idle times. The number of busy periods that are removed depends on the conditional probabilities of the sequence of idle intervals. This technique is effective if there is strong persistent temporal dependence over several lags in the time series of idle intervals.

The effectiveness of these shaping techniques naturally depends on the spatial and temporal localities of the workload, as well as the available buffer size. Observe that while READ offloading determines individual blocks to be copied, Busy Period offloading and Probabilistic offloading make decisions upon *sets of blocks* that are accessed together almost

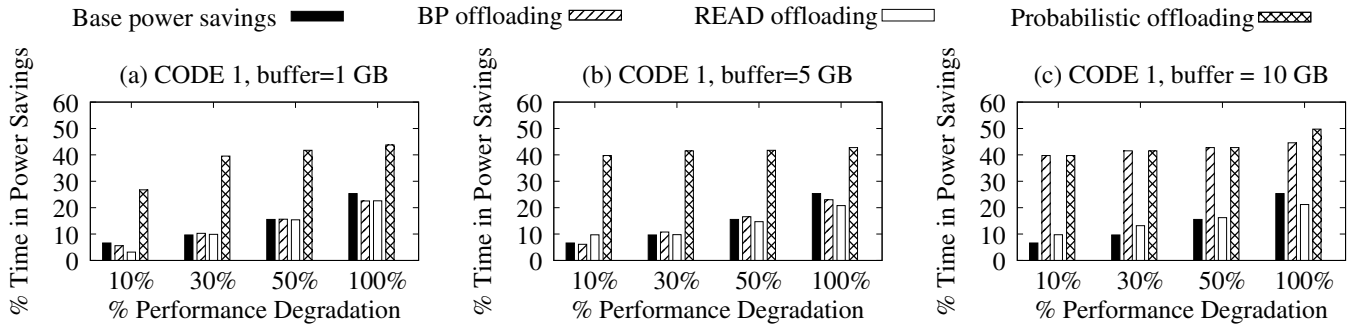

Figure 4: Power savings given user-defined performance degradation equal to 10%-100% for "Code 1" and power savings Level 3. Power savings without any shaping are indicated as "base power savings" (solid black bar). Results are presented for buffer sizes 1-10 GB.

simultaneously (i.e., in the same busy period). The only difference between Busy Period offloading and Probabilistic offloading is that the former makes decisions on frequency of busy periods and the latter makes decisions based on the correlation of the length of idle intervals succeeding the busy periods.

We would like to point out that further idleness fragmentation is a potential side effect of most workload shaping techniques at the disk drive level, including the three proposed here. To illustrate this, consider READ offloading. The expectation is that by removing most frequent READ blocks, the majority of busy periods get shorter (or eliminated) and idle intervals get longer. However, removing individual READs could also partition busy periods into smaller ones and create *more* small idle intervals. The effect of this may lead to counter-intuitive results, e.g., more buffer space may not always lead to larger power savings after workload shaping, because it may increase the fragmentation of idle times in the trace.

While READ and Busy Period offloading offer good power saving when workloads exhibit spatial locality, Probabilistic offloading generally offers good power savings if there is temporal locality among idle interval lengths. The spatial locality of busy periods affects this method indirectly. If the busy periods that are removed based on temporal locality criteria are encountered frequently, then a smaller buffer is needed for the redirected workload. In the following section we present detailed experimentation that further corroborates how the different workload characteristics should guide shaping to maximize power saving.

5. EXPERIMENTAL EVALUATION

In this section, we evaluate in detail the proposed workload shaping techniques via trace-driven simulations. For each workload, each shaping method and buffer size combination, and each disk power saving mode, we use the algorithmic framework presented in [11] and summarized in Section 2.2 to calculate the best scheduling parameters I and T of the power saving mode and the corresponding power savings that would result from that scheduling pair. The I and T values are computed for fixed values of user-provided performance degradation D that we set to 10%, 30%, 50%, and 100%.

Our trace-driven simulation, simulates the collection of data for the first half of the trace and based on analysis of the workload localities, it applies workload shaping for the second half of the trace. Specifically, the simulation collects the necessary data from the trace and populates the data structures described in Section 4 for the first half of the trace. Half of the trace, or six hours (see Table 2) is set as the monitoring window. Upon completion of one monitoring window, the scheduling pair (I, T) is determined, and the resulting "mirrored" blocks from applying a workload shaping technique are marked as such. For the second half of the trace, the simulation puts the disk into a power saving mode every time an idle interval is detected and I time units elapse. The system remains in power savings for T time units or less, if a new IO request arrives for a non-mirrored block and finds the disk in a power saving mode. Each simulation calculates the portion of time that the system is put in a power savings mode.

For all experiments presented here we assume Level 3 power savings. Results with Level 4 are qualitatively similar to those of Level 3 across all workloads and are not presented here in the interest of space. As explained in Section 3, our traces are measured in enterprise systems with limited idleness and few opportunities for power savings. As such Level 5 and 6 of disk power saving modes with high penalties do not yield any savings and are not evaluated here. We also present results for buffer size equal to 1, 5, and 10 GB, that represent at most 15% of available capacity. For all shaping policies, if the buffer fills, then no data are further moved there (i.e., suggested data mirroring from the workload shaping stops). Results for the four workloads are summarized as follows.

"Code 1": In Figure 4 we show the power savings of "Code 1". This is a workload with low spatial and moderate temporal locality. The figure shows the base workload power savings (black bar), and the power savings achieved with Busy Period offloading, READ offloading, and Probabilistic offloading. The base power savings correspond to power savings if the power savings framework presented in [11] is applied on the original workload, i.e., there is no workload shaping.

The results in Figure 4 confirm that across all buffer sizes, the highest power savings are accomplished with Probabilistic offloading. For a large buffer size equal to 10 GB, see Figure 4(c), Busy Period offloading is comparable to Probabilistic offloading. In this case, more busy periods are moved to the buffer, therefore opening up more opportunities for power savings. For small buffer sizes, READ offloading is

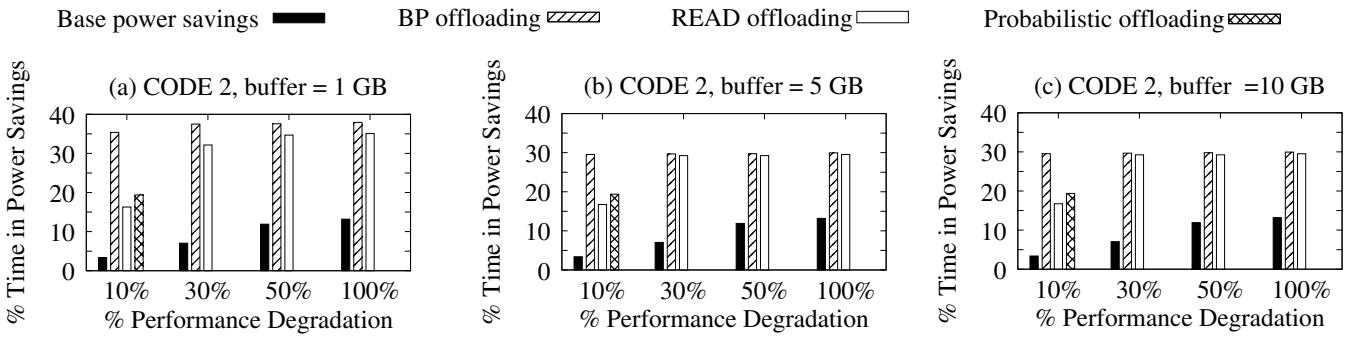

Figure 5: Power savings given user-defined performance degradation equal to 10%-100% for "Code 2" and power savings Level 3. Power savings without any shaping are indicated as "base power savings" (solid black bar). Results are presented for buffer sizes 1-10 GB. Probabilistic offloading requires a buffer size up to 1 GB for this workload, i.e., the results presented for larger buffer sizes do not fill the available buffer.

similar to Busy Period offloading. If only a small buffer is available, then Probabilistic offloading is preferable, because it exploits not only the spatial but also the temporal locality in the workload.[2]

Note that "Code 1" represents a "typical" IO enterprise workload with low spatial locality. And it comes as no surprise that exploiting temporal locality, such as mirroring sets of blocks rather than individual ones, as well as the temporal locality in the length of idle intervals, is effective. Furthermore, selecting the busy periods to be removed based on temporal affinity of idle intervals rather than pure frequencies of the busy periods yields significant power saving.

For a workload of low spatial and moderate temporal locality like "Code 1", the best shaping method is Probabilistic offloading. If a large buffer is available, then Busy Period offloading is also a good choice. However, as we argued previously, we strive for small buffers and exploiting temporal locality as Probabilistic offloading does, succeeds at increasing power savings for a challenging workload such as "Code 1" with *high IO rates* and short original idle intervals (see Table 2).

This workload represents a clear case where a fast analytical method that estimates the power savings for a target performance degradation can be beneficial before choosing a shaping technique as well as the buffer size that should be used to redirect parts of the workload. In all cases here, the analytic methodology of Section 2.2 correctly predicted the relative ranking of the various shaping policies.

"Code 2": In contrast to the previous workload, "Code 2" has high spatial locality and this gives many options for shaping. Table 4 shows that almost 92% of the busy period accesses are removed by redirecting only 1% of the working set. For a small buffer size equal to 1GB, Busy Period offloading gives excellent power savings.

Characterization of frequencies of accesses for "Code 2" suggests that larger buffers of 5 and 10 GB offer marginally better power savings, as they direct to the buffer up to an additional 5% of the working set (see Table 4). Instead, we see that larger buffers can hurt power savings. READ offloading and Busy Period offloading result in a workload with more fragmented idle times than the case of the 1 GB buffer. The result is less power savings for workload shaping policies with large buffers (i.e., 5 GB and 10 GB) than 1 GB, as depicted in Figure 5.

Note the suboptimal performance of the Probabilistic offloading policy. This performance is a direct result of the low persistent temporal locality captured in Figure 3 for "Code 2". Although after five short idle intervals there is a long one, the fact that the two long idle intervals are six lags apart means that a large set of busy periods need to be removed before large idle intervals can be concatenated. Removing the intermediate busy periods results in significant fragmentation of the idleness in the system and consequently low power savings.

However, results from "Code 1" and "Code 2" in Figures 4 and 5, respectively, point out that when the characteristics change orthogonally (i.e., from low spatial locality and moderate temporal locality to high spatial locality and low temporal locality), so does the effectiveness of a specific workload shaping technique. Probabilistic offloading becomes ineffective while Busy Period offloading becomes effective. This observation emphasizes the need for a fully automated framework that estimates which is the most effective, if any, workload shaping technique and activates that one. We address this issue in detail in Section 6.

"File 1": This workload has low spatial but high temporal locality. Temporal analysis, see Figure 3, shows that if a long idle interval is observed, then there is a high probability that another long idle interval follows if we remove two successive busy periods. Figure 6 shows that the Probabilistic offloading is the most efficient shaping method for "File 1". This method provides very large improvement of the original power savings that are reported in [11] (see solid black bars). Probabilistic offloading increases power savings up to 30% for large degradation and it is preferable to READ or Busy Period offloading across all buffer sizes.

Table 2 shows that the spatial locality of the busy periods is low, i.e., 10% of the working set corresponds to only 38% of the busy periods, thus a large buffer is needed to be effective under Busy Period offloading. The results of READ offloading are similar to those of Busy Period offloading.

Although not identical, the characteristics of "Code 1" and "File 1" fall in the same categories when evaluated for power savings and the recommendation is similar, i.e., to use Probabilistic offloading. This confirms that temporal locality

[2]As seen in Figure 3 the conditional probabilities indicate moderate temporal locality for "Code 1".

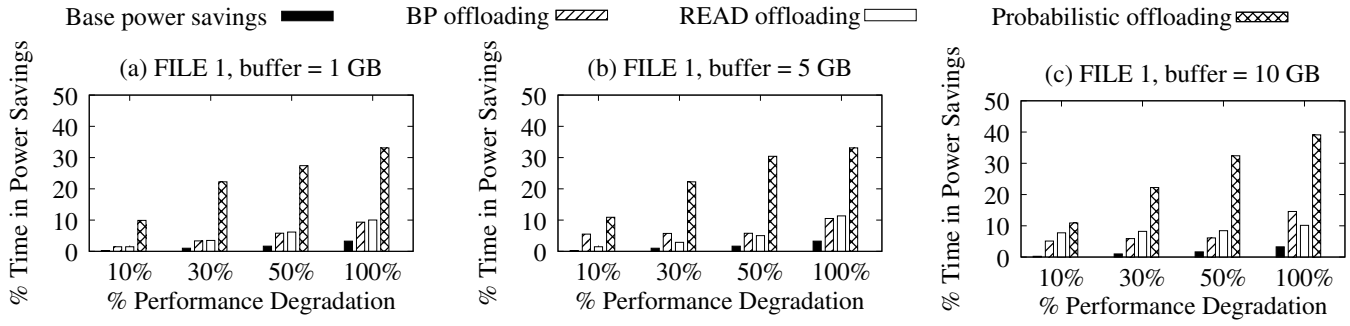

Figure 6: Power savings given user-defined performance degradation equal to 10%-100% for "File 1" and power savings Level 3. Power savings without any shaping are indicated as "base power savings" (solid black bar). Results are presented for buffer sizes 1-10 GB.

even if moderate, should be exploited if spatial locality is low.

File 2: This workload has good spatial locality and moderate temporal locality. The power savings for "File 2" are shown in Figure 7. In this case Probabilistic offloading performs comparable to Busy Period offloading. This indicates that when both spatial and temporal locality are present, then selecting a workload shaping technique that exploits either characteristic will result in maximum power savings in the system.

An interesting observation is that READ offloading has negative effect on the power savings for "File 2". Figure 7 shows that this type of offloading reduces power savings capabilities of the original workload in most cases. The spatial locality of READs is lower than spatial locality of Busy Periods (see Tables 3 and 4). In addition, the results of READ offloading for "File 2" emphasize that it is not enough to redirect multiple independent blocks but rather sets of blocks that are accessed consecutively in a busy period. When removing READS in this case, we remove blocks without ensuring that we steer away from further fragmentation of idle times.

6. DISCUSSION

In the previous sections, we developed several workload shaping techniques, that aim at redirecting portions of the disk workload elsewhere in the system, with the goal of increasing idleness and opportunities for power savings without violating performance targets. However, the main conclusion from the workload characterization presented in Section 3 and the evaluation in Section 5 is that the effectiveness of a shaping technique strongly depends on the stochastic characteristics of the workload.

Although all shaping techniques proposed here are adaptive in nature, i.e., their parameters, such as the pair (I, T) for scheduling the power saving periods or the amount of buffer available for redirecting the workload are adjusted to the changing workload, our evaluation showed that not one shaping technique would work best for all workloads. From the perspective of increasing power savings in a disk drive, our goal is to enable the system to make a decision not only on the parameters of an active workload shaping technique but also determine which workload shaping technique to activate for the current workload. Because workload shaping techniques do impose extra work in the system, only one should be active at any given time.

As we strive to propose a fully automated framework for power savings, we utilize the analytic estimation capability of the algorithm described in Section 2.2 to determine at any given time:

- the highest amount of potential power savings for the current workload or any workload shaping technique;
- the parameters to schedule power saving modes.

The analytic framework of Section 2.2 provides to the system an accurate estimation of benefits and penalties associated with any combination of workload, power saving mode level, and shaping technique.

Since enterprise systems are sensitive toward additional delays in the system (as it is the case with the delays caused by activating disk drives in power saving modes), we recognize that for certain workloads (e.g., the workload during business hours) no power saving mode or workload shaping should occur. However, we may enable the system to detect power saving opportunities as the workload changes and act upon these opportunities.

This analytic estimation and decision making framework for power savings in disk drives is complementary to the workload shaping techniques available in the literature. For example the effectiveness of WRITE offloading [8] or working set offloading[16] for a given workload can be assessed similarly to the effectiveness of the workload shaping techniques proposed here, adding even more flexibility and adaptivity in the system.

Our framework allows us to figure out which are the disks that cannot be shut down. Since these disks have to be active, then perhaps we can send some of the redirected workload to them instead of a sending it to a "buffer" such as a cache. The extra work due to redirection is very small given that the goal of the shaping techniques here is to identify a small portion of the workload only. Serving of the redirected workload by these target disks, from the power consumption perspective in the cluster remains the same because this work would have to be served anyway somewhere in the cluster.

6.1 Algorithmic Framework

In order to be able to estimate the power saving potential for the current workload or the available workload shaping

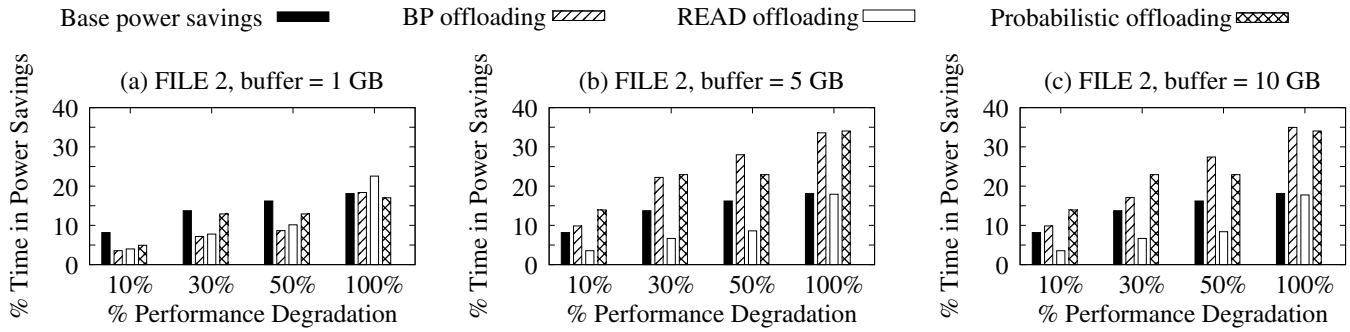

Figure 7: Power savings given user-defined performance degradation equal to 10%-100% for "File 2" and power savings Level 3. Power savings without any shaping are indicated as "base power savings" (solid black bar) . Results are presented for buffer sizes 1-10 GB. Probabilistic offloading requires a buffer size up to 4 GB for this workload, i.e., the results presented for larger buffer sizes do not fill the available buffer.

techniques and make a decision on how the system should operate, several metrics should be monitored in the system. Specifically the system should monitor the CDH of idle times and the frequencies of accessed blocks, individually and in busy periods. From these metrics it is fairly simple to derive the CDH of idle times if a specific workload shaping technique (e.g., READ offloading) is activated.

To illustrate how the CDHs of the shaped workloads can assist in identifying opportunities for power savings, we show the CDH of the original workload for "Code 2" in Figure 8(a) as well as the CDHs for the "shaped Code 2" for the three shaping strategies, see Figure 8(b)-(c) for a 1 GB buffer. Figure 8(b) shows that the CDHs of Busy Period offloading and READ offloading are very similar – both CDHs have more distinctive "knees" and higher CDH values for larger intervals than the Original "Code 2" CDH, suggesting good opportunities for power savings. Indeed, for degradation greater than 30% the power savings of these two policies are close. Probabilistic offloading for this workload results in aggressive creation of very large idle intervals, observe the x-range of the plot in Figure 8(c), but also many small ones. Yet, for "Code 2", Busy period offloading is more effective as shown in Figure 5. When it comes to power savings, the "knee" shapes in the CDH of idle times represents the idleness that yields the most power savings, recall the in Section 2.2.

For each CDH of idle times, i.e., the monitored and the generated ones, the power savings are estimated using the estimation procedure in Section 2.2 using as additional input the user-provided acceptable performance and/or power saving targets. However, since there are multiple workload shaping techniques and multiple power saving levels in a disk drive, the number of combinations for all potential power saving settings can grow significantly. In this case, even with the very compact and efficient procedure of Section 2, there is a risk of overloading the system with estimations.

In order to target only a few potentially effective scenarios we use CDHs that we have already built to guide our decision making. While the benefits of power savings and workload shaping depend on the individual system's quality of service goals, we believe that the majority of systems aim to redirect only a small portion i.e., 5% of the available capacity to a remote buffer. We believe that if more data needs to be copied elsewhere, then the problem changes from shaping the workload to creating multiple copies of the same data and activating them on demand. The latter problem is more complex and imposes more overhead in the system. Here we aim at identifying power saving opportunities with limited overhead.

With the above goals we define a workload to exhibit

- high spatial locality if most frequent accesses span for less than 10-20% of the capacity and low otherwise;
- high temporal locality if the sum of probabilities of several consecutive long idle intervals is more than 0.5 and low otherwise.

Using the above definitions to quantify spatial and temporal locality and based on our evaluations of the individual workload shaping techniques in Section 5, we establish the following relations between workload's spatial and temporal locality and the workload shaping technique that holds potential for power savings:

- if the workload has high temporal locality then Probabilistic offloading with small buffers should be evaluated further,
- if the workload has high spatial locality then READ offloading and Busy Period offloading with small buffers should be evaluated further, and
- if the workload does not exhibit spatial or temporal locality then Busy Period and Probabilistic offloading with large buffers (i.e., 15% of the available capacity) should be evaluated further.

Once we determine which set of workload shaping techniques to evaluate further, then we use the algorithm in Figure 2 to estimate the potential power savings for the set of selected workload shaping techniques for different buffer sizes and power saving modes. In this process, we determine the scheduling parameters as well. Once all estimations are done, the framework selects the set of parameters that achieve the highest power saving for the smallest buffer size without violating system performance targets. The accuracy of our estimations on power savings is demonstrated in [11], therefore we are confident that the decisions on selecting a shaping technique would be correct and yield the best possible power savings without violating performance targets.

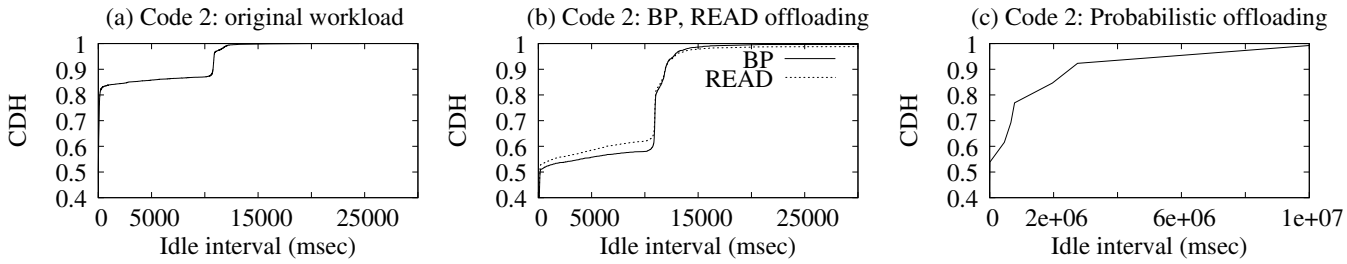

Figure 8: CDHs of idle times for "Code 2". We report on the CDHs of the original workload (no shaping), Busy Period and READ offloading, and Probabilistic offloading. We assume that the buffer is 1 GB.

The estimations are done once in several hours (or when the workload structures including the CDH and the frequencies of READs and busy periods change significantly). For a description of each algorithmic step, see Figure 9.

1. If in *characterization state* do
  a. update IOs and busy / idle periods traces
  b. update the CDH of idle times
  c. update the CDH of busy period accesses
  d. update the CDH of READ accesses and
  e. calculate the conditional probabilities of a long idle interval followed by another long up to 20 lags
    set $long = mean + 3 \cdot standard\ deviation$
2. If system in *decision making state* do
  a. If temporal locality is high (> 0.5 for multiple lags)
    continue with Probabilistic offloading
    else if spatial locality high (working set $< 20\%$ *capacity*)
    continue with Busy Period offloading
    else if no spatial or temporal locality
    continue with Probabilistic or Busy Period offloading
  b. Use selected workload shaping technique
    i. generate the would-be CDHs of idle times from traces of **1.a** and buffer sizes 1, 5, 15% of capacity
    ii. use Algorithm in Figure 2 to calculate power saving with the would-be CDHs for all buffer sizes
  c. Use the CDH generated in **1.b** with Algorithm in Figure 2 to calculate base power savings
  d. Select shaping with the highest power savings

Figure 9: Algorithm of workload shaping framework.

7. RELATED WORK

Workload characterization has established that idleness and burstiness in disk drive workloads remain invariant across different applications [10]. In contrast, the READ/WRITE ratio, workload sequentiality, arrival rate, and request characteristics are largely environment-specific [10]. Temporal and spatial characteristics in disks have been examined in the literature [18, 4]. In [18] the authors characterize a wide range of storage workload traces from Windows servers regarding the IO rate, average system and disk inter arrival, and average IO request size. This characterization aims at giving a complete picture of storage workload to improve modeling, simulation, and storage firmware. In [4] a study on storage workload characterisation in virtualized environments is presented. The access locality, IO sizes, and READ/WRITE ratio of storage workloads in virtualized environments are measured and characterized. The goal in [4] is to reduce latency and improve performance using workload characterization.

Redirecting the entire workload (or parts of it) to other disks or other parts of the system for archival and backup storage systems, are explored in [1]. Offloading the WRITEs of a workload for power savings to another persistent storage space in a cluster is presented in [8] and its effectiveness is shown to be increasing power savings by 45-60% for workloads that are WRITE intensive. The techniques we propose here are more general than WRITE offloading because we offload the most active parts of the workload (READs and/or WRITEs) and those may change as the workload changes across time.

In [19] a theoretical and on-line version of redirecting workload to a cache is presented. The authors consider that power savings are present all the time, which means that user response times can degrade significantly as there is no mechanism to restrain performance degradation.

SRCMap [16] 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 drawn for a workload increases linearly as the load intensity increases. SRCMap offloads the entire working set to increase the idle period duration. Here, our focus is at the disk drive level and on moving only a part of the working set in order to enlarge idle intervals and still achieve high power savings with limited book-keeping. Another difference in SRCMap and our shaping framework is that there is no mechanism to keep performance degradation within certain levels, and that the elapsed time before power savings are activated is *always* a constant value – instead we estimate this value I and this changes depending on the workload and the acceptable performance target. SRCMap uses replication in order to achieve the minimum amount of active physical volumes, as well as dynamic adaptation to workload popularity.

In [2] context aware power savings mechanisms for interactive applications that are based on monitoring user behavior are proposed. The goal is to limit delays due to powering up the hard disk. The authors approach this by using a simple predictor as well as by capturing correlations between user interactions.

In [17] PARAID exploits the unused space in a disk to replicate in a skewed fashion and sets of disks are organised in a hierarchical way. Each set represents a gear similar to automobiles that offers different parallelism and bandwidth. PARAID conserves energy by exploiting this disk parallelism to spin down disks. More disks are used for better performance, less for power savings. In [9] a technique to

increase workload burstiness to disks is applied for power efficiency, by aggressive pre-fetching. A trace based approach that studies the queue depth and the inter-arrival process of requests in order to save power in data centers is presented in [12]. A system with two priority queues is devised in [7] where burstiness and interleaving of workload is used in order to increase power efficiency. Dynamic placement of free blocks to improve disk performance and power savings with a runtime component is presented in [6]. How to optimize file system options for power savings is considered in [14].

In this paper we focus on workload characteristics using monitoring to capture temporal and spatial locality. The target of our workload shaping policies is to enlarge idle times that can be used for power savings using the most effective workload shaping technique as suggested by workload characterization. The main difference from the related work above is that the techniques proposed are *always* adjusted to the temporal and spatial characteristics via continuous workload monitoring, while abiding by the user-set performance targets. The methodology is lightweight because it relies on simple calculations on collected histograms that reflect how the workload changes across time.

8. CONCLUSIONS AND FUTURE WORK

We present methods for workload shaping that are based on learning from past workload temporal and spatial characteristics. Our evaluation shows that we manage to predict a good shaping method that increases power savings across a variety of workloads. Remarkably, even when a trace exhibits only moderate temporal locality, probabilistic offloading can significantly increase the potential for power savings even by directing only a portion of the workload to a small buffer. If spatial locality is also present, then power savings have the potential to increase even further.

In the future, we intend to extend the workload shaping mechanism proposed here in order to target specific power savings (i.e., how and how much should we "shape" the workload in order to achieve certain power targets). An interesting extension would be also on workload multiplexing, i.e., how to redirect workload on other disks such that the performance of the disk where workload is redirected remains intact and power savings are maximized.

Acknowledgments

This work is supported by NSF grants CCF-0811417 and CCF-0937925. The authors thank Seagate Technology for providing the enterprise traces used for this work.

9. REFERENCES

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

[2] I. Crk and C. Gniady. Context-aware mechanisms for reducing interactive delays of energy management in disks. In *Proceedings of USENIX Annual Technical Conference*, pages 71–84, 2008.

[3] S. Dobson, R. Sterritt, P. Nixon, and M. Hinchey. Fulfilling the vision of autonomic computing. *IEEE Computer Society*, pages 35–41, January 2010.

[4] A. Gulati, C. Kumar, and I. Ahmad. Storage workload characterization and consolidation in virtualized environments. In *Workshop on Virtualization Performance: Analysis, Characterization, and Tools (VPACT)*, 2009.

[5] Hitachi Global Storage Technologies. Power and acoustics management. White paper at: http://www.hitachigst.com, 2007.

[6] H. Huang, W. Hung, and K. G. Shin. 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)*, volume 39, pages 263 – 276, 2005.

[7] L. Lu and P. Varman. Workload decomposition for power efficient storage systems. In *Proceedings of the 2008 conference on Power aware computing and systems (HotPower'08)*, pages 13–18, 2008.

[8] D. Narayanan, A. Donnelly, and A. I. T. Rowstron. Write off-loading: Practical power management for enterprise storage. In *Proceedings of the USENIX Conference on File And Storage Technologies (FAST)*, pages 253–267, 2008.

[9] A. E. Papathanasiou and M. L. Scott. Energy efficiency through burstiness. In *Proceedings of 5th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 44–54, 2003.

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

[11] A. Riska and E. Smirni. 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)*, pages 131–140, 2010.

[12] S. Sankar and K. Vaid. Addressing the stranded power problem in datacenters using storage workload characterization. In *Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering*, pages 217–222, 2010.

[13] Seagate Technology. Constellation ES: High capacity storage designed for seamless enterprise integration. Product overview at: http://www.seagate.com, 2009.

[14] P. Sehgal, V. Tarasov, and E. Zadok. Evaluating performance and energy in file system server workloads. In *Proceedings of 8th USENIX Conference on File and Storage Technologies (FAST'10)*, pages 19–33, 2010.

[15] L. Smarr. Project Greenlight: Optimizing cyber-infrastructure for a carbon-constrained world. *IEEE Computer Society*, pages 22–27, January 2010.

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

[17] C. Weddle, M. Oldham, J. Qian, and A. Wang. PARAID: A grear-shifting power-aware raid. In *Proceedings of 5th USENIX Conference on File and Storage Technologies (FAST'07)*, pages 245–269, 2007.

[18] B. Worthington and S. Kavalanekar. Characterization of storage workload traces from production windows servers. In *Proceedings of the International Symposium on Workload Characterization (IISWC)*, pages 119 – 128, 2008.

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