Efficient Management of Idleness in Storage Systems

NINGFANG MI
College of William and Mary
ALMA RISKA
Seagate Research
QI ZHANG
Microsoft
EVGENIA SMIRNI
College of William and Mary
and
ERIK RIEDEL
Seagate Research

Various activities that intend to enhance performance, reliability, and availability of storage systems are scheduled with low priority and served during idle times. Under such conditions, idleness becomes a valuable "resource" that needs to be efficiently managed. A common approach in system design is to be nonwork conserving by "idle waiting", that is, delay the scheduling of background jobs to avoid slowing down upcoming foreground tasks.

In this article, we complement "idle waiting" with the "estimation" of background work to be served in every idle interval to effectively manage the trade-off between the performance of foreground and background tasks. As a result, the storage system is better utilized without compromising foreground performance. Our analysis shows that if idle times have low variability, then idle waiting is not necessary. Only if idle times are highly variable does idle waiting become necessary to minimize the impact of background activity on foreground performance. We further show that if there is burstiness in idle intervals, then it is possible to predict accurately the length of incoming idle intervals and use this information to serve more background jobs without affecting foreground performance.

This work is partially supported by NSF grants CNS-0720699 and CCF-0811417, and by Seagate Research.

Authors' addresses: N. Mi, Department of Computer Science, College of William and Mary, Williamsburg, VA; email: ningfang@cs.wm.edu; A. Riska, Seagate Research, 1251 Waterfront Place, Pittsburgh, PA 15222; Q. Zhang, Microsoft, One Microsoft Way, Redmond, WA 98052; E. Smirni, Department of Computer Science, College of William and Mary, Williamsburg, VA; E. Riedel, Seagate Research, 1251 Waterfront Place, Pittsburgh, PA 15222.

DOI 10.1145/1534912.1534913 http://doi.acm.org/10.1145/1534912.1534913

Categories and Subject Descriptors: C.4. [**Computer Systems Organization**]: Performance of Systems—*Design studies; reliability, availability, and serviceability*

General Terms: Algorithms, Design, Management, Performance, Reliability

Additional Key Words and Phrases: Resource management, storage systems, performance guarrante, low priority work, idleness, continuous data histogram, foreground jobs, background jobs, idle periods

ACM Reference Format:

Mi, N., Riska, A., Zhang, Q., Smirni, E., and Riedel, E. 2009. Efficient management of idleness in storage systems. ACM Trans. Storage 5, 2, Article 4 (June 2009), 25 pages.
DOI = 10.1145/1534912.1534913 http://doi.acm.org/10.1145/1534912.1534913

1. INTRODUCTION

As computer systems operate 24 hours a day, 7 days a week, maintenance tasks are commonly scheduled during idle times [Golding et al. 1995]. These tasks are considered *background tasks* and aim at improving reliability and availability [Abd-El-Malek et al. 2005; Bachmat and Schindler 2002; Merchant and Yu 1994; Schwarz et al. 2004], at enhancing performance [Thereska et al. 2004; Huang et al. 2005; Litzkow et al. 1988; Venkataramani et al. 2002], and at preserving power [Helmbold et al. 2000; Colarelli and Grunwald 2002]. Completion of background tasks is critical to system operation, yet their priority is not as high as that of foreground jobs, that is, regular jobs submitted by the system users.[1] Consequently, scheduling of background activities should not compromise the performance of foreground tasks.

Background tasks that are instantaneously preemptable minimally affect foreground performance. This is not the case if the background tasks are nonpreemptive. Nonpreemptive background tasks are common in storage systems. For example, at the disk level, the head seeking portion of the request service process is noninstantaneously preemptable. Storage-level background tasks such as detecting media errors via background media scans [Schwarz et al. 2004; Iliadis et al. 2008; Mi et al. 2008], RAID rebuilds in case of disk array failures [Sivathanu et al. 2004; Merchant and Yu 1994], and disk-level data mirroring for faster data access [Huang et al. 2005] are examples of nonpreemptable background tasks. Other examples of nonpreemptable background tasks can be found in mobile or archival systems, where components (such as disks) are shut off to conserve power [Douglis et al. 1995; Helmbold et al. 2000; Colarelli and Grunwald 2002]. The nonpreemptive nature of such background tasks makes their execution challenging if delays on foreground jobs are to be kept at a minimum.

Efforts on utilizing idle times to improve performance or reliability are often *system specific* and are evaluated in the context of a specific feature based on prototyping and measurements [Golding et al. 1995; Abd-El-Malek et al. 2005; Thereska et al. 2004; Helmbold et al. 2000] or analytic models [Bachmat and Schindler 2002; Merchant and Yu 1994; Muntz and Lui 1990; Niu et al. 2003; Osogami et al. 2005]. In addition to the aforesaid system-specific solutions,

[1]In this article, we use the terms "task" and "job" interchangeably.

there are also efforts to evaluate the general concept of managing idle times, by viewing idleness as an additional resource [Douceur and Bolosky 1999; Eggert and Touch 2005]. From the theoretical perspective, the performance of systems with foreground and background jobs (also viewed as systems with jobs of high and low priority) has been extensively analyzed via queuing theory (see Takagi [1991] and references within).

Motivated by storage system examples, our focus is on the general problem of serving nonpreemptive background jobs during idle times. In storage systems, as in computer systems in general, idleness depends on system workload (see Section 6 for more details). For example, in general-purpose servers (including Web servers and file servers), variability and burstiness dominate workload and idleness characteristics. On the other hand, video streaming servers are expected to work under more deterministic workloads, and consequently the idleness will reflect that characteristic. Here, idleness is considered an additional system resource, but its management is driven by the performance trade-off between maintaining desired levels of foreground performance while maximizing the completions of nonpreemptive background tasks.

In contrast to previous work, which sustains foreground performance by letting the storage system to "idle wait" before a background job is scheduled, sometimes causing background work starvation, we propose to complement "idle waiting" with the "estimation" of the amount of background work to be served in any given idle interval. Such an approach does not compromise the foreground performance and avoids starvation (if any) among background jobs by allowing background jobs to be served in as many idle intervals as possible. As a result, the storage system is better utilized, while foreground performance targets are met.

The balance between foreground and background activities is achieved by monitoring the characteristics of foreground and background jobs, similarly to other works in the literature that focus on the same problem [Eggert and Touch 2005; Golding et al. 1995]. In addition, we also collect measurements of the empirical distribution of idle times. Resource management of idle times is now done in a dynamic way, using statistical information not only on the foreground and background job demands, but also on the idle intervals. All statistical information is collected online and is incorporated into the policies that manage idle times.

Detailed analysis of various systems with different statistical characteristics of foreground and background jobs and idle times shows that the effectiveness of idle waiting depends on the variability of the empirical distribution of idle times. In systems with low variability of idle times, idle waiting is not effective. The opposite holds for idle times of high variability. In both cases, the cumulative data histogram of idle times is used to dynamically determine the length of idle waiting.

In addition, we show how to take advantage of the burstiness (if any) in idle intervals to improve utilization of idle times. Specifically, if idle times exhibit burstiness, it becomes possible to predict the length of upcoming idle intervals. This becomes extremely effective, in particular when the background nonpreemption penalty on foreground performance is severe (e.g., spinning up disks

that were spun down to preserve power). If a sequence of idle interval lengths is positively correlated, then it implies that long (short) idle intervals are observed together. Consequently, if the current idle interval is long, then we can predict more accurately whether the next idle interval is going to be long as well. If the incoming interval is predicted to be long, then more background jobs can be scheduled for service. As a result, the overall system is better utilized while the slowdown of foreground jobs is still kept up to a predefined target. Validation of our methodology using actual disk drive traces shows that monitoring stochastic characteristics of idle times, in addition to the characteristics of foreground and background tasks, is an effective way to manage idleness.

This article is organized as follows. Section 2 presents related work. Section 3 outlines how stochastic characteristics of idle intervals can be used for background scheduling. We present a new methodology to manage idle intervals by exploiting their variability in Section 4. Section 5 describes how to exploit burstiness of idle times. We validate the proposed methodology in Section 6 using actual measurements from disk-level traces. Conclusions are given in Section 7.

2. RELATED WORK

Various studies have shown that in systems, periods of high utilization may be interleaved with long stretches of idleness [Litzkow et al. 1988; Golding et al. 1995; Eggert and Touch 2005; Riska and Riedel 2006]. A myriad of approaches have been proposed to best utilize idle times in order to enhance system performance, reliability, and availability. System idleness may be exploited locally (i.e., within the same system), or remotely (i.e., busy systems may offload part of their work in idle ones).

As it becomes more common for systems to operate 24/7, idle times offer the only time window to complete maintenance work [Golding et al. 1995; Huang et al. 2005; Thereska et al. 2004; Abd-El-Malek et al. 2005; Schwarz et al. 2004; Bachmat and Schindler 2002]. Consequently, the general problem of idle-time scheduling has recently regained attention [Golding et al. 1995; Douceur and Bolosky 1999; Eggert and Touch 2005] as a distinct problem within the larger and well studied problem of priority scheduling [Takagi 1991].

Utilization of remote idleness is often exploited in distributed or peer-to-peer systems and focuses on identifying idle remote systems to complete some work remotely. V-system [Theimer et al. 1985] and Condor [Litzkow et al. 1988] are examples of such systems. Other studies on utilizing remote idleness are presented in Osogami et al. [2005] and Lo et al. [2004].

On the analytic side, several models have been developed for analysis of systems where foreground/background jobs coexist, including vacation models [Niu et al. 2003; Thomasian and Nicola 1993; Xu and Alfa 2002] and queuing models of cycle stealing [Takagi 1991; Osogami et al. 2005].

The main performance pitfall of scheduling background tasks during idle times relates to cases where background jobs cannot be preempted instantaneously and foreground performance may be significantly affected. If tasks are nonpreemptable, effective scheduling of background tasks is more challenging.

Eggert and Touch [2005] focus on managing idle intervals under a wide range of characteristics for background and foreground tasks and first define the notion of the preemption interval or idle wait period that delays execution of background jobs in idle periods. This technique avoids using short idle intervals to schedule long background jobs. Efforts to adaptively determine the amount of time that the system should idle wait are proposed in Douglis et al. [1995] and Helmbold et al. [2000] for power saving in mobile devices by spinning down their disks.

The closest to the work presented here is the one presented in Eggert and Touch [2005]. In this article, we depart from previous work by presenting a methodology to sustain foreground performance while avoiding background starvation, by estimating the amount of work to be completed in any idle interval, in addition to the estimation of the idle wait. Furthermore, the estimation of the idle wait and per-interval background work is based not only on the characteristics and performance of the foreground and background jobs, but also on the characteristics of idle intervals. We identify when idle wait is effective and when it is not (i.e., it should be zero) based on the histogram of the observed idle times. More importantly, we propose ways to exploit burstiness in idle times (if burstiness exists) to best utilize pairs of long idle intervals, resulting in superior overall performance.

3. CHARACTERIZING IDLENESS

First we develop an understanding of the significance of the system idle times characteristics. Our goal is to develop policies that sustain system's foreground performance without starving background work. Via idle intervals characterization, we aim at estimating as accurately as possible how much background activity can be packed into an idle time period.

The stochastic characteristics of idle times are a result of the complex interaction of arrival and service processes in the system. Instead of deriving characteristics of idle times through analysis of the arrival and service processes, we concentrate on idle intervals themselves, which capture the interaction of the arrival and service processes. Idle intervals are viewed here as a separate stochastic process.

The characterization is based on two dimensions: variability and burstiness. Variability determines how well the mean of idle interval lengths describes the observed idle interval lengths. If variability is low then it is expected to observe idle intervals that have lengths close to the mean. If the variability is high, then it is expected to have the majority of idle intervals with lengths much shorter than the mean and a few of them much longer than the mean. On the other hand, burstiness in the sequence of idle intervals determines if the idle intervals in the near future will have the similar length as the currently observed idle intervals.

3.1 Independent Idle Intervals

First, we focus on independent idle intervals. If the sequence of idle intervals is independent, then the length of upcoming idle intervals does not depend

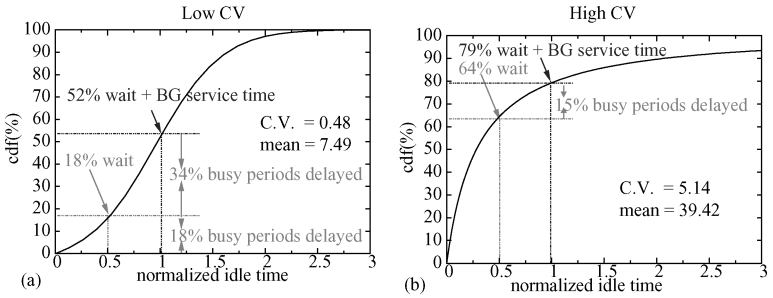

Fig. 1. Cumulative density function of idle times with low C.V. and high C.V. (left and right plot, respectively). The x-axis gives idle times normalized by their mean.

on the length of the current idle interval. Therefore, any information about the length of upcoming idle intervals is contained in its empirical distribution. After computing the first two moments of the observed idle times, namely mean and variance, the Coefficient of Variation (C.V.) is calculated. The C.V. indicates the existence of tails in the empirical distribution of idle times.

Figure 1 depicts the Cumulative Density Function (CDF) of two stochastic processes, one with low C.V. (left plot) and the other one with high C.V. (right plot). Here, if the value of C.V. is less than 1.0, then the process is considered to have low C.V., otherwise, the process has high C.V. For simplicity, the x-axis in Figure 1 gives values that are normalized by the mean; for example, "0.5" on the x-axis corresponds to a value that is half of the mean of the empirical distribution.

Observing the long tail in the right plot is straightforward: The CDF line goes toward 100% with a much lower pace than the CDF of the left plot. The CDFs can therefore provide important information about the majority of upcoming intervals. For idle intervals with low C.V.s, the mean of the empirical distribution provides a good guess about the idle interval length. For idle intervals with high C.V.s, there is a large percentage of intervals that are much shorter than their mean and a small percentage of intervals that are much longer than their mean. This useful information on the anticipated length of future idle intervals is embedded on the CDF and proves tremendously useful for efficient background scheduling, particularly in determining the idle wait length and the amount of background work to be scheduled in any given idle interval.

The intuition behind idle waiting relates to the relative lengths of foreground and background jobs. Idle waiting depends on the expected service time of the background jobs and the length of interarrival times [Eggert and Touch 2005]. However, idle waiting is not equally effective for idle intervals of low C.V. or high C.V. We illustrate this via a simple example.

Following the assumptions in Eggert and Touch [2005], we set the idle wait to be equal to the expected background service time, which is set to be half of the expected idle intervals length. If this idle wait policy were used for the case where the idle intervals are of low variability (see Figure 1, left plot), then 18% of the idle intervals would not be utilized by background jobs, and

consequently 18% of the busy periods[2] would not be affected by background work either. If only one background job is scheduled after idle waiting elapses equal to the average background service time (i.e., a normalized idle time of 0.5), it is expected that 34% of all idle periods (52% of idle intervals with normalized idle time = 1, minus 18% of idle intervals with normalized idle time = 0.5, which equals to 34%) would be serving a background job, while a foreground job arrives (which causes undesired delays on foreground work). In contrast, if idle wait is zero and average background duration is half of the mean of idle intervals length, then only 18% of the busy periods in the system are affected if only one background job is served. Consequently, we conclude that no idle wait is necessary for systems with idle intervals of low variability.

The policy of idle waiting for the expected background service time affects foreground jobs very differently if idle intervals are of high variability. The high C.V. plot of Figure 1 shows that by idle waiting for a period equal to the expected background service time, 64% of all idle intervals are not used to serve background jobs. If the expected length of a background job is equal to half of the mean of the idle times and one background job is served in one idle interval, then only 15% of busy periods are affected by the background work. If idle intervals have high C.V., then idle waiting helps exploit the "longer" intervals at the tail of the distribution.

The preceding two examples of low and high C.V. highlight the disadvantage of a scheduling policy that uses a "fixed" idle waiting period as in Eggert and Touch [2005]. The idle waiting period should be adapted according to the characteristics of foreground/background service demands and idle times.

3.2 Bursty Idle Intervals

We now turn to bursty idle intervals. All discussion of the previous section applies here as well. In addition, burstiness in the sequence of idle times provides extra information which can be used for the prediction of the length of the upcoming idle intervals [Golding et al. 1995]. Burstiness in a sequence implies that among all the observed values in the sequence, the *order* of their occurrence is not random as it is in the independent case. In a sequence that is characterized as bursty, very large (multiple times larger than the mean) or very small (multiple times smaller than the mean) values are sampled close to one another.

Figure 2 shows this effect in a bursty sequence of observations. The first two plots in Figure 2 represent an independent sequence[3] and the last two plots represent a bursty sequence.[4] All values are coarsely partitioned in two classes, namely "small" and "large," where the partition point is determined

[2]In a system, every idle period is followed by a busy period that starts when a foreground job arrives and finds the system idle.

[3]The independent sequence of idle times comes from a single server queue with interarrival times drawn from an Erlang distribution with mean rate equal to 0.01 and C.V. equal to 0.33. Service times are exponentially distributed with mean service rate equal to 0.1.

[4]The bursty sequence represents the idle times in a single server queue with correlated interarrival times that are drawn from a 2-stage Markovian-Modulated Poisson Process (MMPP) with mean rate equal to 0.01, C.V. equal to 9. The service process is also correlated; it is drawn from an MMPP

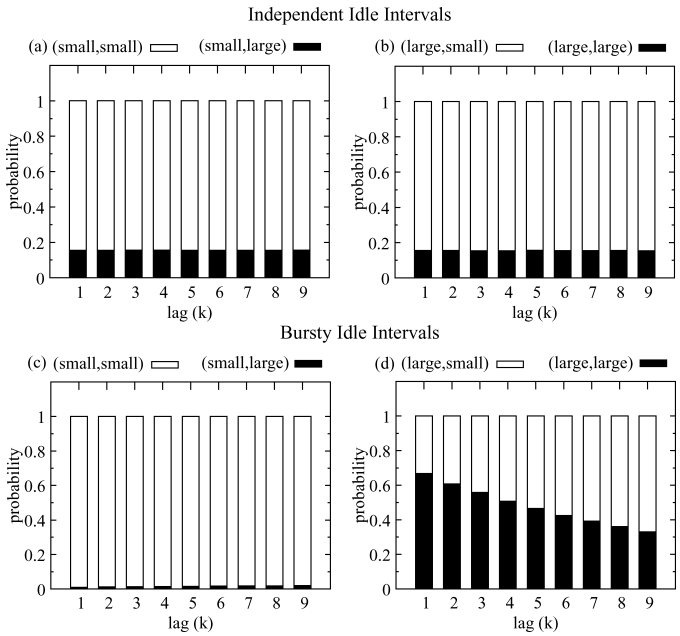

Fig. 2. Probabilities of pairs as a function of their relative distance (lags). Plots (a) and (b) give results for independent idle intervals and plots (c) and (d) for bursty idle intervals.

as $(C.V. + 1) \cdot mean$. In the independent sequence, 85% of all idle intervals belong in the small category, while for the bursty case 97% of all idle intervals belong in the small category. The group of small values represents the majority of observations in both sequences.

The x-axis in each plot represents the lags, that is, the distance in the number of observations between the two instances in the sample. For a given lag, ordered pairs of observations are constructed. Let $(X, Y)_i$ denote the probability that an interval in category Y follows an interval in category X with i lags apart, where X, Y are either "large" or "small". Figures 2(a) and 2(c) give the probability of occurrence of (small,small)$_i$ (white part of the bar) and (small,large)$_i$ (black part of the bar) pairs as a function of the lag i, $1 \leq i \leq 9$, that is, their relative distance in the sequence. Figures 2(b) and 2(d) give the probability of occurrence of (large,small)$_i$ (white part of the bar) and (large,large)$_i$ (black part of the bar) pairs. Note that the summation of probabilities of occurrence of (small,small)$_i$ and (small,large)$_i$ (or (large, small)$_i$ and (large,large)$_i$) pairs is one.

In the independent case (see Figures 2(a) and 2(b)), the probability of the next observation (idle time) being small or large does not depend on the value of the current observation, as expected. However, in the dependent case (see Figures 2(c) and 2(d)) the effect is different. Figure 2(c) shows that if the current idle interval is small, then the probability to have a small idle interval in the next 9 lags is very high. Figure 2(d) equivalently shows how the current

process with mean rate equal to 1, C.V. equal to 4.5. The MMPP is a versatile process that can capture both variability and burstiness.

large idle interval determines with high probability (e.g., 0.65 at lag 1) that the next observed idle interval would be large as well. The probabilities drop as a function of the lag and reach 0.35 for 9 observations apart. This information, in addition to the one provided by the cumulative density function, can be used to improve scheduling of background activities by allowing to complete more background work (during long idle intervals) without imposing additional delays on foreground work.

4. BACKGROUND WORK DURING INDEPENDENT IDLE INTERVALS

Idle waiting is used as a technique to ensure that a desired level of foreground performance is sustained while serving nonpreemptive background work. Because idle waiting is a nonwork-conserving strategy, background jobs may suffer from starvation [Eggert and Touch 2005]. In our approach to scheduling background jobs during idle times, we avoid background starvation by coupling idle wait with the amount of work that can be effectively completed within an idle interval. These two scheduling parameters are determined using the foreground and background service demands as well as the distribution of idle times. We consider the following policies.

Mean-Based. This policy serves as a baseline comparison [Eggert and Touch 2005] and defines the idle wait as the mean service time of background jobs. After the idle wait elapses, the system serves background jobs until a foreground job arrives.

CDF-Based. Similar to the mean-based policy, this policy also deploys idle waiting. Different from the mean-based policy, the CDF-based policy continuously estimates the idle wait length using the empirical distribution (i.e., cumulative histogram) of idle intervals and the mean service time of background jobs. The CDF-based policy idle waits *only* if idle intervals have high C.V. More details are given in Sections 3 and 4.3.

CDF/w Estimates. This policy estimates the idle wait the same way as the CDF-based policy, but it is more conservative because it limits the number of background jobs to be served in an idle interval according to the equation

$$T \cdot \frac{90^{th} \text{ percentile of idle intervals} - \text{idle wait}}{\text{Average background service time}}, \tag{1}$$

where $0 < T \leq 1$ is a parameter that adjusts the estimated number of background jobs assuming that the interval is large (i.e., equal to the 90^{th} percentile of idle times). This parameter controls the performance degradation of foreground jobs. As T increases, foreground performance degrades. T is adjusted to reflect variability in the distribution of idle intervals, that is, T is close to 1 under idle intervals of high variability and less than 1 for low-variability intervals.

In all of the aforementioned policies, if a new foreground request finds a background job in service, it waits until that job completes.

4.1 Simulation Environment

The three policies are evaluated via the simulation of a single server queue. We assume that there is no limit on the waiting queue capacity and the service process is First-Come First-Serve (FCFS). Consistently with Eggert and Touch [2005], we also assume that there are *always* background jobs waiting for service. This is the case in storage systems where background media scans happen continuously to ensure that any existing disk latent errors are detected and recovered before the user accesses the data [Bairavasundaram et al. 2007]. We discuss other scenarios where the amount of background work is not infinite as part of our future work in Section 7. Slowdown of foreground jobs caused by background activity is computed as the ratio of foreground response time when background jobs are served to foreground response time when no background jobs are served. We consider a 5–10% slowdown of foreground jobs due to background jobs to be acceptable and run our experiments with a slowdown threshold of 7%.

The service times of background jobs are exponentially distributed. We expect this to be a realistic assumption, because disk-level service times have variability (i.e., measured via the C.V.) close to that of the exponential distribution [Riska and Riedel 2006]. The background service times are adjusted so that two different systems are simulated: (a) one system where both foreground and background jobs have the same mean service time (dubbed also as "short foreground – short background" system) and (b) one system where the average background service time is 7 times longer[5] than the average foreground service time (dubbed also as "short foreground – long background").

Foreground interarrival times are drawn from an Erlang distribution, resulting in idle intervals of low variability. Drawing foreground interarrival times from a Lognormal distribution results in a system with high variability in its idle intervals. For both systems, the mean interarrival times are adjusted so we evaluate system utilizations due to foreground jobs only, equal to 10%, 30%, and 70%, representing a system under low, medium, and high foreground load, respectively. All simulations are done with a 1 million sample space of foreground jobs and results are reported with 98% confidence intervals.

We first use synthetic workloads to quantitatively evaluate the policies under controlled systems with different levels of variability or burstiness. In Section 6, two real disk-level traces are used for evaluation.

4.2 Idle Intervals with Low Variability

Results of the experiments with low variability in idle intervals are given in Figure 3. The first row of graphs corresponds to the system with short foreground – short background jobs, the second row corresponds to the system with short foreground – long background jobs. Four performance metrics are presented: (a) the number of completed background jobs in millions (first column), (b) the overall system utilization (second column), (c) the background-caused slowdown of foreground jobs (third column), and (d) the foreground response

[5]The results with other background service time ranges are qualitatively the same. They are not reported here due to lack of space.

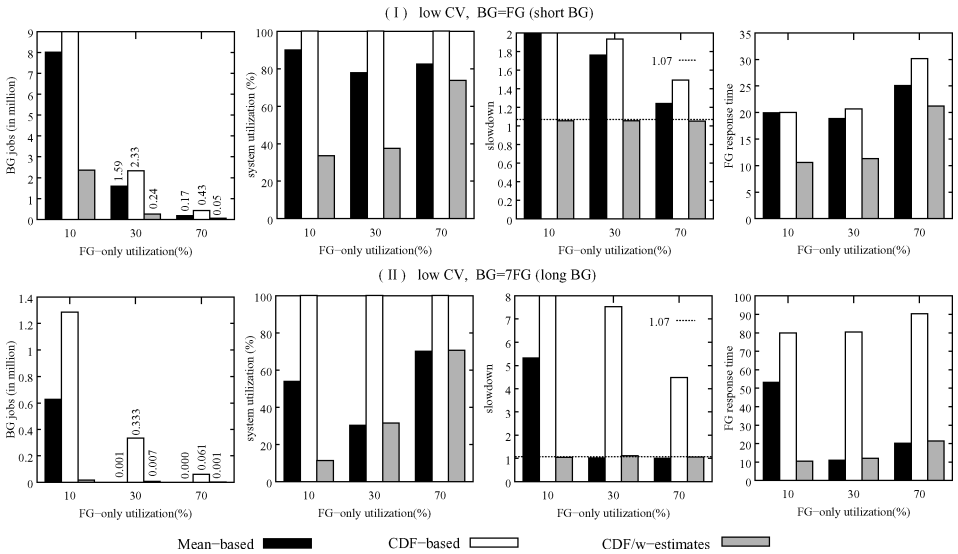

Fig. 3. Overall system performance measured by the number of completed background jobs (in millions), overall system utilization, slowdown of the foreground jobs attributed to background activity (the horizontal line corresponds to 7% slowdown), and the absolute foreground response time. The idle intervals are independent and with low variability. Three foreground system utilizations are evaluated, namely, 10%, 30%, and 70%. Foreground utilization is controlled by changing the foreground arrival rate and fixing its service time. The first row of graphs shows the case when the background jobs are "short", that is, as long as foreground jobs, and the second row shows the case when the background jobs are "long", that is, 7 times longer than foreground jobs.

times (fourth column). The last two metrics capture, respectively, the relative and the absolute background-caused degradation in foreground performance. Successful policies should increase the system utilization while the slowdown of foreground jobs is kept up to the predefined target of 7%. Results shown in Figure 3 can be summarized as follows.

—The mean-based and CDF-based policies are very aggressive in the number of background jobs that they serve (first column), which results in high system utilization (second column). The penalty, though, on foreground jobs is significant (third and fourth columns). Note that because there is always an infinite supply of background jobs, if the system serves background jobs as much as possible until foreground jobs arrive, then the overall system utilization reaches 100%.

—The CDF/w-estimates policy consistently meets the performance target of foreground jobs, across both experiments with short and long background jobs. This is due to the parameter T in Eq. (1), which determines how many background jobs to serve such that the effect on foreground performance is contained within the predetermined limits.

—Under low foreground-only utilizations (i.e., 10%) there is more room to exploit idle times and serve large quantities of background jobs while limiting the effect on foreground slowdown. For example, with CDF/w-estimates, the overall system utilization for short background jobs increases to 35% from

the initial 10% and 2.3 million background jobs are served, which is about 10 times and 46 times more than the number of background jobs served under the foreground-only utilization of 30% and 70%, respectively.

—As foreground utilization in the system increases, the relative impact of background activity on foreground jobs reduces. The reason is that response times of foreground jobs are already dominated by waiting in the queue due to the high foreground load. As a result, waiting because of background work is not as noticeable. In low utilizations, foreground response time is dominated by the service time rather than the waiting time in the queue. Any background-caused delay is immediately observed because it may be the only wait that the foreground jobs experience. As a result, background work can be scheduled effectively even when the foreground system utilization is high.

—There is a significant difference in relative policy performance if background jobs are short or long. First, the mean-based policy performs poorly under long background jobs. This is because, in the case of low variability in the idle times and high utilization, most idle times are short, that is, shorter than service times of long background jobs. If the idle wait is equal to the average background service time, then the majority of idle intervals are not used for servicing any background activity. Note that the number of long background tasks completed under the mean-based policy is only 1000 for a foreground utilization of 30% and none for a foreground utilization of 70%. By adjusting T in Eq. (1), the CDF/w-estimates policy remains flexible, avoids background work starvation, and maintains the foreground performance targets. For example, even under the case of long background jobs and medium or high utilization, the respective numbers of background jobs completed are 7000 for a foreground utilization of 30% and 1000 for a foreground utilization of 70%.

—As expected, absolute foreground response time (shown in column four of Figure 3) increases with foreground utilization, even if the delays due to background jobs are limited. In this article, the focus is to achieve foreground performance targets measured by slowdown (a relative measure) rather than response time (an absolute measure). If the latter were the case, and foreground performance under 70% utilization would be the performance target, then all three policies meet that target if the foreground utilization is 10% or 30%.

For idle intervals with low variability, the three policies use idle intervals differently. The mean-based policy "consumes" the beginning of an idle interval via the idle wait and background jobs are served at the end of the interval. The CDF-based does not wait idle and serves background jobs as long as there is no waiting foreground job, utilizing the system 100%. The CDF/w-estimates policy schedules background activity at the beginning of the idle interval and not at the end (as the mean-based one) by estimating the number of background jobs to be served in any idle interval. This proves effective and strikes a good balance between the performances of foreground and background jobs.

4.3 Idle Intervals with High Variability

If idle intervals have high variability, then policies that worked well under low-variability conditions cease to be effective. The long tail in the distribution of

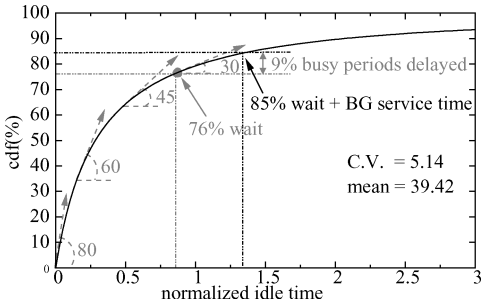

Fig. 4. Relation between the slope of the CDF and the length of idle wait when the distribution has high C.V.

idle times suggests that delaying background jobs is promising as now only long idle intervals are used for background jobs. No jobs are scheduled in idle intervals that are too short to fit a single background job.

Determining the length of idle wait is done dynamically by constructing online the cumulative histogram of idle times. The idle wait is defined by the CDF as the point in the histogram where the sharp increasing portion of the body ends and the slow increasing part of the tail starts (see Figure 4).

Changes in the histogram shape are detected by inspecting the slope of both portions of the CDF curve. When the slope decreases to a predefined angle, for example, 30 degree[6] in this case, then the desired point, that separates the body from the tail of the histogram, is found. The number of jobs to be served in each interval is then computed using Eq. (1).

The predefined slope angle that determines the separation point between the body and the tail of the histogram defines how aggressive the usage of idle intervals is, that is, the higher the slope, the smaller the idle wait. In order to contain the slowdown of foreground jobs to a minimum, the angle should be set such that the CDF's slope is small and the usage of idle times is conservative.

Figure 5 shows the results for two experiments: short foreground – short background (first row) and short foreground – long background (second row) under different foreground-only utilization levels. Similar to Figure 3, the following metrics are reported: the number of completed background jobs, the overall system utilization, the relative slowdown in foreground response time, and the absolute foreground response time. The performance target of foreground job slowdown remains 7% in these experiments as well.

The figure illustrates that the high variability in idle times offers better opportunities to utilize idle intervals. Especially for the first experiment with short background jobs, system utilization significantly increases. Results from these experiments are summarized as follows.

—The mean-based policy utilizes the system best, but at the expense of higher delays for foreground jobs. For long background jobs, foreground jobs slow

[6]Here, the predefined slope angle of 30 degrees is chosen such that the usage of idle times is neither aggressive nor conservative.

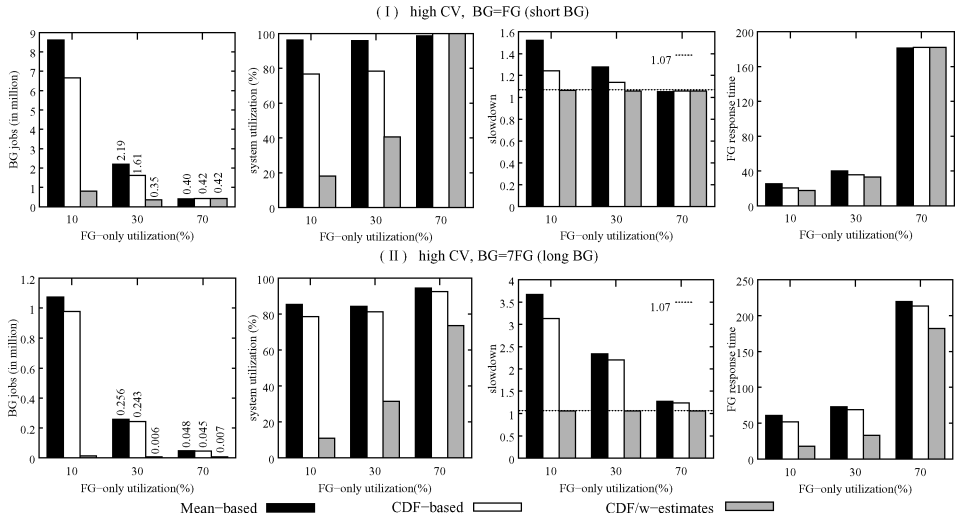

Fig. 5. Overall system performance measured by number of completed background jobs in millions, overall system utilization, slowdown of the foreground jobs attributed to background activity (the horizontal line corresponds to 7% slowdown), and the absolute foreground response time. The idle intervals are independent and with high variability. Three foreground system utilizations are evaluated, namely 10%, 30%, and 70%. Foreground utilization is controlled by changing the foreground arrival rate and fixing its service time. The first row of graphs shows the case when the background jobs are "short", that is, as long as foreground jobs, and the second row shows the case when the background jobs are "long", that is, 7 times longer than foreground jobs.

down up to 3.75 times for low utilization and 2.5 times for medium utilization (see third column in Figure 5(II)).

—The CDF/w-estimates policy is always below or right at the 7% slowdown target (see dotted line in the third column) at the expense of scheduling less background jobs and lower utilization levels (see first and second columns in Figure 5).

—The system is utilized as much as 100% with minimal performance degradation of foreground jobs when background jobs are short and the foreground utilization is high (e.g., 70% foreground utilization bars in Figure 5(I)). The CDF/w-estimates policy becomes aggressive here by selecting a higher slope and by estimating the number of background jobs to be served using $T = 1$ in Eq. (1) (see rightmost set of bars in the plots of Figure 5(I)).

—Recall that no long background jobs were served under the mean-based policy for idle intervals with low variability and medium/high utilization (see first plot in Figure 3(II)). Under highly variable idle times, the result is different (see first plot in Figure 5(II)). This is due to the existence of some very long idle times that enables the completion of long background jobs even if idle waiting is long.

If idle intervals have high C.V., then the mean-based policy, the CDF-based, and the CDF/w-estimates policy (different from the case of idle times with low C.V.) operate similarly; that is, they all idle wait in the beginning of an interval

and utilize its end. For medium to high utilization and short background jobs, the CDF-based policies estimate an idle wait that is similar to the static one used by the mean-based policy. For long background job, the number of estimated background jobs to be served in an idle interval for the CDF/w-estimates policy is high, similar to the number of other two policies that are oblivious of such estimation.

The results presented here show that there are cases where the utilization of idle times can be aggressive without affecting the performance of foreground jobs. Generally, to sustain foreground performance it is essential to idle wait before starting any type of background work. The necessity of estimating the amount of background work to complete in an idle interval increases as the length of the average service time of background jobs increases relative to the average foreground service time.

The following summarizes a comparison of the results in Figures 3 and 5.

—The number of completed background job decreases faster as foreground utilization increases for low-variability than high-variability idle times (first column in each figure).

—Overall system utilization is better under high-variability than low-variability idle times (second column in each figure).

—Foreground slowdowns are higher under low-variability than high-variability idle times (third column in each figure), because foreground response times (fourth column in each figure) are smaller (and more sensitive to additional delays) under low-variability than high-variability idle times.

4.4 Tail of the Response Time Distribution for Foreground Jobs

We have shown that the CDF/w-estimates policy consistently maintains foreground slowdown less than 7% while serving as many background jobs as possible. The figures of the two previous subsections give the average slowdowns of foreground jobs. Here, we analyze the distribution of foreground response times, particularly the corresponding tail.

In order to focus on the tail in the distribution, Figure 6 depicts the Complementary Cumulative Distribution Function (CCDF) of the foreground response times under a 30% foreground-only utilization. Here we only compare the tails of foreground response time distributions with and without background jobs when the CDF/w-estimates policy is used for the utilization of idleness.

The figure shows that the impact on the tail of the response time distribution depends on the length of the background job. For short background jobs, irrespectively of the variability of idle intervals, the foreground response time distribution with background jobs follows the distribution of the foreground response time without background jobs. The slight difference in average foreground response times exists throughout the distribution. Short background jobs, in general, delay the foreground jobs for a short period of time only. In this case, there are many background jobs that are scheduled, so there is a large portion of foreground jobs that are slightly delayed.

In the case of long background jobs, the behavior is different. Although the CDF/w-estimates policy schedules only a few large background jobs to contain

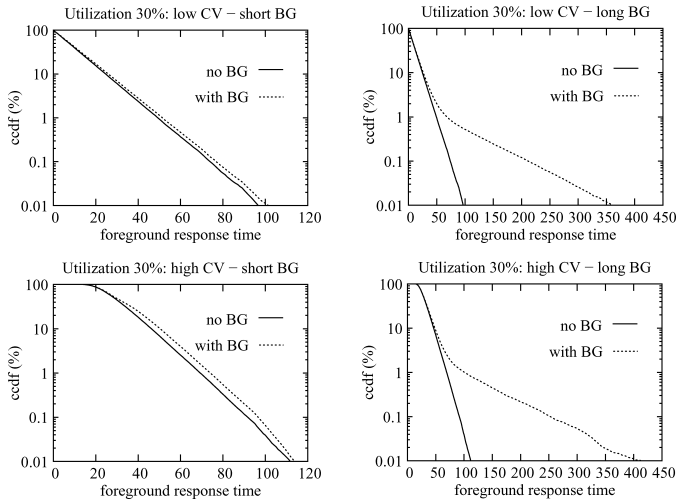

Fig. 6. Tail of the foreground response time with and without background work when the idle intervals are independent. The dashed lines in the graph correspond to results for the CDF-w/estimates policy. The foreground-only utilization is 30%.

delays to a few foreground jobs only (less than 2% of all foreground jobs), the tail of the foreground response time distribution is much longer than when no background jobs are served. The long tail is a result of the significant delays caused to foreground jobs by long background ones.

5. BACKGROUND ACTIVITY IN BURSTY IDLE INTERVALS

The policies presented in the previous section are based on the cumulative histogram of the empirical distribution of idle times. Here, the focus is on idle intervals that, in addition to being highly variable, are also bursty.

Section 3.2 presents an analysis of bursty stochastic processes and shows that burstiness enables prediction of the near future based on the near past. If a sequence of observations is positively bursty, then it implies that long observations (i.e., several times larger than the mean) are clustered together in the sequence and short ones (i.e., several time smaller than the mean) are clustered together as well. Figure 2(d) shows exactly this: In a bursty sequence, if the current observation has a large value, then it is with high probability that the next observation is also large. This property can be used to manage system idleness more efficiently by exploiting long intervals aggressively.

To detect burstiness in idle times, a similar structure as the one depicted in Figure 2 is constructed online. First, observations of idle times are classified as small or large. Consistent with Section 3.2, the range of idle times is partitioned in "small" and "large" at the $(C.V. + 1) \cdot mean$ point. Then, every pair of idle times is classified in the appropriate category (i.e., (small,small), (large,small), (small,large), or (large,large)) and the corresponding probability is calculated. Pairs do not include consecutive observations only, but also those that are separated by up to 9 observations (lags). Once these conditional probabilities are constructed, they are used to predict more accurately whether the next idle

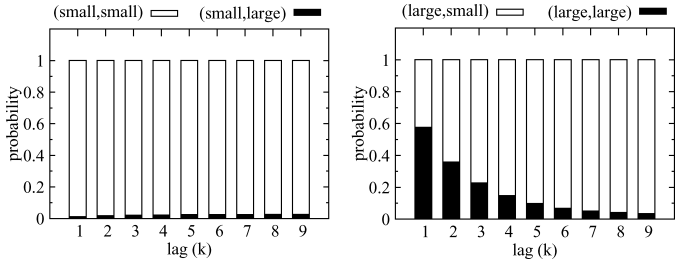

Fig. 7. The probabilities of occurrence of (small,small), (small,large), (large,small), and (large,large) pairs, for the bursty idle times used in the experiments of Section 5.

interval is short or long. The CDF/w-estimates policy is augmented into the Bursty+CDF/w-estimates policy as follows:

—If the current interval belongs to the "large" category, then the next interval is predicted to be "large" with probability ρ. Note that the probability of the occurrence of (large,large) pairs is not equal to 1, which implies that there will be mispredicted "large" intervals. The probability ρ controls the number of misspredicted "large" intervals.

—If the next idle interval is predicted to be "large," instead of using Eq. (1) to estimate the number of background jobs to be served in that interval, the following equation is used:

$$T \cdot \frac{(C.V. + 1) \cdot mean - \text{idle wait}}{\text{Average background service time}}, \tag{2}$$

which implies that the length of the incoming idle interval is at least the "large" value $(C.V. + 1) \cdot mean$.

Exploiting the long intervals (i.e., those longer than $(C.V. + 1) \cdot mean$) in a bursty sequence allows to increase overall system utilization, because during those intervals more background work may be scheduled without affecting foreground performance. Benefits are different for short and long background jobs.

For short background jobs. Stringent foreground slowdowns are achieved without radically reducing the number of completed background jobs,

For long background jobs. A given amount of background work can now be completed with less degradation on foreground performance, resulting in shorter tails in foreground response times.

For the experiments in this section, the interarrival and service processes of foreground jobs are bursty processes.[7] The service times of background jobs are exponentially distributed. The results in idle times with probability of (small,small), (small,large), (large,small), and (large,large) pairs are shown in Figure 7. The evaluation of improvements due to the use of conditional probabilities in the CDF/w-estimates policy is done via two sets of experiments:

[7]Burstiness is achieved by using an MMPP process.

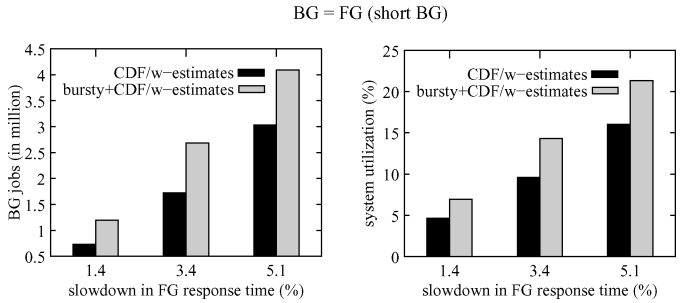

Fig. 8. Number of completed background jobs and overall system utilization under bursty idle intervals. Three different foreground slowdowns are considered, namely 1.4%, 3.4%, and 5.1%. The more stringent the requirements on foreground slowdown, the higher the relative improvement achieved by the Bursty+CDF/w-estimates policy.

one with short and the other with long background jobs. The sample space of foreground jobs is one million.

Figure 8 presents the number of completed background jobs and the overall system utilization as a function of the foreground slowdown, when short background jobs are served. There are more background jobs completed with Bursty+CDF/w-estimates than with CDF/w-estimates. The relative performance gap between the two policies increases as foreground slowdown decreases. If the requirements on foreground slowdown are relaxed, then the difference between the two policies diminishes. In general, overall system utilization improves with Bursty+CDF/w-estimates.

In a system that serves long background jobs, background jobs are chosen to be 300 times longer than the foreground service times, on the average. An example of such a scenario is spinning down disks to conserve power. Spinning them back up and ready for work is orders of magnitude larger than serving a single request. This extreme case is difficult to address; scheduling a background job in the wrong interval may have a tremendous impact on the tails of foreground jobs.

Figure 9 presents the foreground job slowdowns for the CDF/w-estimates and the Bursty+CDF/w-estimates policies when background jobs are long. The dotted horizontal line represents the performance target of 7% average slowdown for foreground jobs. The Bursty+CDF/w-estimates policy attempts to detect pairs of long idle intervals and utilize them by serving there the background jobs because then the probability to affect foreground is low. We select three levels of completed background work, which are: (1) high: with 7455 completed background jobs, the background work is more than two times the foreground work; (2) medium: with 1816 completed background jobs where the background work is more than half of the foreground work; and (3) low: with 1252 completed background jobs, here the background work is only a third of foreground work. These three levels are a result of different values of the parameter T used in Eq. (1).

Figure 9 shows the foreground job slowdown for each of the aforesaid three levels. For high, medium, and small amounts of completed background work,

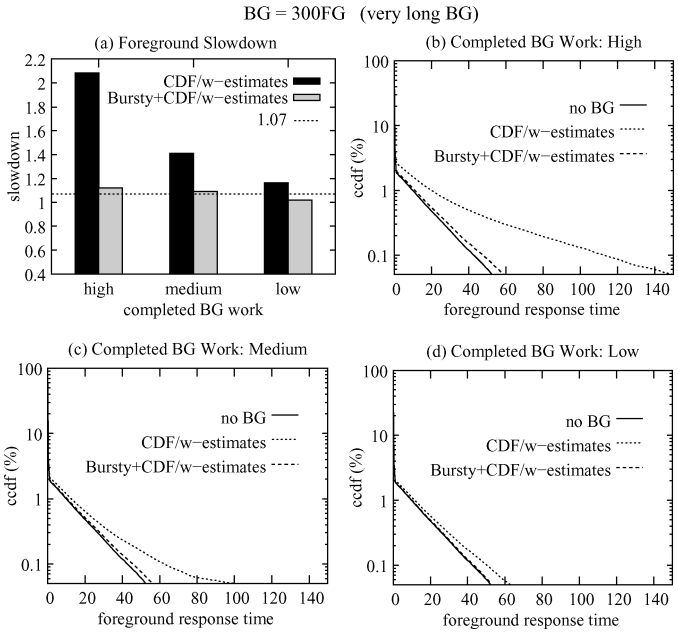

Fig. 9. Foreground performance in a system with bursty idle intervals when the amount of background work completed is high, medium, and low. Plot (a) gives the foreground slowdown for each amount of background work and plots (b)–(d) give the respective CCDFs of foreground response time distributions.

the foreground slowdown under CDF/w-estimates is, respectively, 110%, 40%, and 17%. The Bursty+CDF/w-estimates policy reaches foreground slowdowns of 12%, 9%, and 2%, respectively.

Experiments in Section 4.4 showed that long background jobs affect the tail of the foreground response time distribution. The tail of the foreground response time distribution under the long background jobs is given in Figure 9 with the plots of the CCDFs of the foreground response times. The figure also plots the CCDFs of foreground jobs with no background activity (labeled "no BG") as a baseline comparison. When there is no background activity, the tail of the foreground response time is not long; for example, only 0.05% of foreground jobs have response time larger than 52. Only 1% of jobs have response time larger than 10 when service time is 1 on the average.

The CDF/w-estimates policy, being oblivious to burstiness, affects significantly both average foreground response time as well as its distribution tail. For example, when the completed background work is high, 0.05% of foreground jobs have response time larger than 150, making the tail almost 3 times longer from the cases when there is no background activity. As the amount of completed background work decreases, the tail of the foreground response time distribution shortens and approaches the baseline tail.

Figure 9 also plots the tail of the foreground response time distribution under the Bursty+CDF/w-estimates policy. This policy utilizes mostly long idle intervals. Therefore the number of delayed foreground jobs by long background ones

Table I. Trace Characteristics

Trace	Mean Arrival	Mean Service	Mean Idle	CV Idle	Mean BG Service
T1	62.64	5.50	190.08	6.41	5.00
					50.00
					300.00
T2	252.29	5.50	731.34	3.90	300.00

The unit of measurement is ms.

under Bursty+CDF/w-estimates is reduced. As a result, not only is the slowdown of foreground work substantially smaller than under CDF/w-estimates, but also the tail of the foreground response time distribution is close to the baseline tail.

The results presented in this section show that if idle intervals are bursty (i.e., the series of idle intervals contains information on the order of observations) we can predict the occurrences of long idle intervals, which in turn can be used to efficiently schedule large quantities of background work without affecting foreground jobs.

6. VALIDATION VIA DISK-LEVEL TRACES

The work presented in this article is motivated by background work in storage systems. In general, storage systems deploy a variety of background activities, including media scans, verification of data written on the disk, and data movement for faster access [Bachmat and Schindler 2002; Huang et al. 2005; Golding et al. 1995; Mi et al. 2008; Riska and Riedel 2008]. We validate the results of Section 4.3 and Section 5 using traces that are measured in actual storage systems.

Traces used in this section are measured at different disks of a 40-disk storage system of an in-the-field email server. The traces record, in milliseconds, both arrival and departure times for each foreground request in the system and allow for *exact* computation of idle and busy times at the disk level. The main statistical characteristics of the traces relevant for background work scheduling are given in Table I.

The main observation from the statistics in Table I is the substantial difference between the mean of foreground interarrival times and the mean length of idle intervals. Results in Table I further confirm that neither the foreground arrival nor the service process can provide enough information for background scheduling, and instead we should focus on idle times. Consequently, instead of using foreground arrivals to guide idle time management (as in Eggert and Touch [2005]), we suggest to monitor and estimate idle time characteristics and use them to guide background scheduling.

The traces have variable idle times and can be used to validate the results of Section 4.3, which deals with scheduling background work in highly variable idle intervals. We also check for burstiness and present in Figure 10 the probability of pairs of "large" idle intervals of up to 9 observations (lags) apart. The maximum probability of pairs of long idle intervals is only 0.2 for trace T1 and about 0.5 for trace T2. Consequently, trace T1 is viewed as a trace

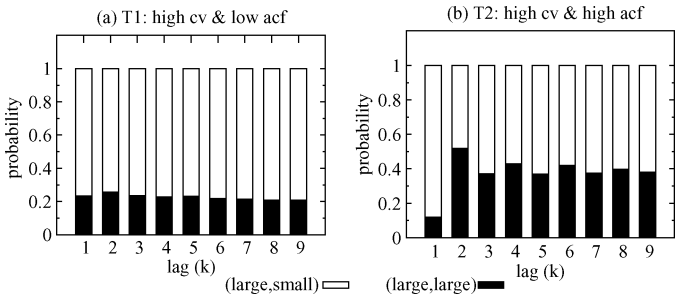

Fig. 10. Probabilities of (large,small) and (large,large) pairs for traces T1 and T2.

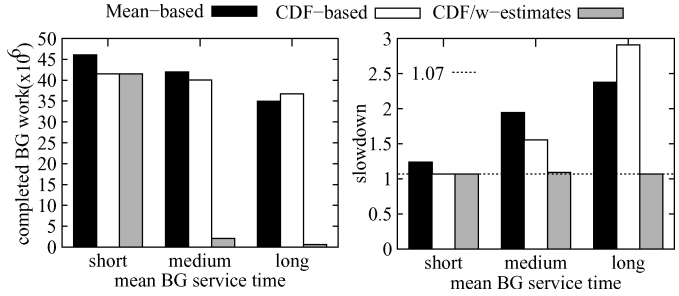

Fig. 11. Number of completed BG jobs and foreground slowdown for trace T1. Three background service demands are chosen that correspond to short, medium, and long background jobs.

with weak dependency (i.e., the observations are nearly independent). Trace T2 has stronger dependence among the observations, although the probability of (large,large) pairs is not as high as for the synthetic trace in Section 5. Trace T2 is clearly a trace with bursty idle intervals. Trace T1 is used to validate the results of Section 4.3 and trace T2 is used to validate results of Section 5.

According to the discussion earlier on this section, we do not deal with identifying why idle intervals are bursty or not, although the fact that burstiness exists in many processes associated with disk drives [Riska and Riedel 2006] implies also idle time burstiness. As shown by the results for Trace T2 in Figure 10, burstiness in idle times exists and taking it into consideration for managing idle time utilization as in Section 5 yields realistic benefits.

In the experiments with trace T1, all three policies are evaluated, namely mean-based, CDF-based, and CDF/w-estimates. The mean background service time is chosen to be 5ms, 50ms, and 300ms (see last column of Table I), and represents the cases of short, medium, and long background jobs, when compared to the mean foreground service time of 5.5ms. These service times of background jobs are all exponentially distributed. Similar demands of disk background activities are write verification (short), moving large chunks of data (medium), and flushing the write cache (long). More details on such storage system background tasks can be found in Golding et al. [1995].

Figure 11 plots results for trace T1. The CDF/w-estimates policy, consistently with results in Section 4.3, outperforms the other two policies when it comes to meeting foreground performance requirements. In the experiments with trace

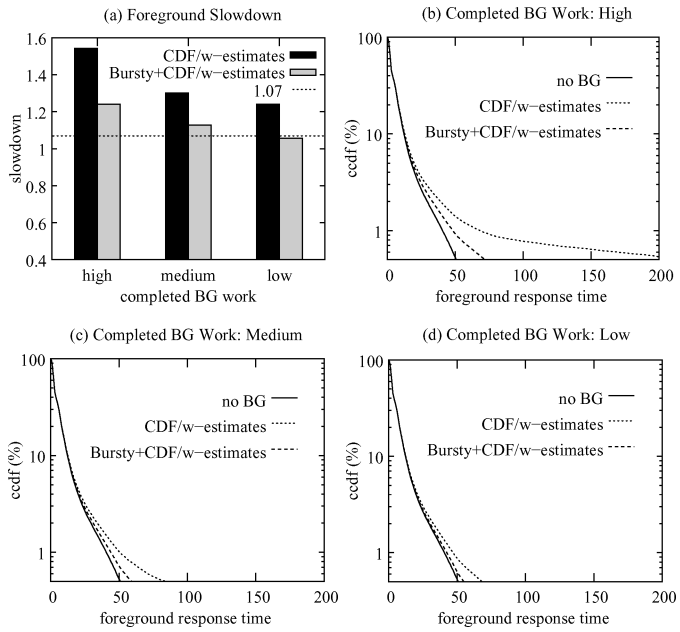

Fig. 12. Average slowdown of foreground jobs and the tail of response time distribution for three levels of completed background work, for trace T2.

T2, background service time is set to be 300ms. Experiments are conducted only with long background jobs and not short ones because with this set of experiments, the emphasis is on exploiting burstiness of idle times to schedule the most challenging long background jobs. Consistently with the results in Section 5, benefits are higher for long rather than short background jobs. Here only the CDF/w-estimates and the Bursty-CDF/w-estimates policies are evaluated (as in Section 5). Figure 12 presents average slowdown of foreground jobs for three levels of completed background work, namely high (83,998 jobs), medium (47,702 jobs), and low (23,322 jobs). By exploiting burstiness, both average foreground slowdown and the tail of the foreground response time distribution are improved. These trace-based experiments confirm our analysis in previous sections using synthetic workloads.

7. CONCLUSIONS AND DIRECTIONS FOR FUTURE WORK

Utilization of idleness in storage systems is regaining wide interest because various advanced techniques that enhance performance and reliability generate additional low-priority work (i.e., background tasks) to be served during idle times rather than compete for resources with user activity (i.e., foreground jobs). The goal is to minimize any degradation on foreground performance that may be inflicted by serving noninstantaneously-preemptive background tasks while not starving background tasks for service either.

This article focuses on the general concept of effective management of idle times. The work evaluates the effectiveness of idle wait, the default strategy that serves background tasks in nonwork-conserving fashion by letting the

storage system idle until a predefined period of time elapses. We show that idle wait-based strategies can be further enriched by an estimation of the amount of background work to be served in the idle interval. This estimation ensures that foreground performance is not compromised and background jobs are not starved.

We show that monitoring the stochastic characteristics of idle times is as important as monitoring the characteristics of foreground and background jobs. In particular, if idle intervals have low variability, then idle waiting is not effective. However, if idle times are highly variable, then idle waiting remains effective for scheduling background jobs without delaying foreground ones. For idle intervals with high variability, we propose to compute the length of the idle wait dynamically using the cumulative histogram of the observed idle times.

Apart from managing effectively idle intervals by distinguishing between low and high variability, the article identifies burstiness as a source of additional information to improve idle time utilization. The analysis shows that if burstiness exists in the observed idle intervals, then it can be used to predict the length of the upcoming idle intervals. Predicting that the next idle interval is long given that the current interval is also long is of particular interest, because scheduling of background jobs can become more aggressive. As a result, more background work completes with less delays in foreground jobs and tremendously shorter tails in the foreground response time distribution.

Throughout the article, we evaluate the proposed policies via synthetic workloads and measured disk drive traces. In the disk drive traces, the main characteristics of idle times are high variability and weak/strong burstiness. These characteristics are exploited for idle-time management. Trace-driven validation confirms that monitoring stochastic characteristics of idle times, in addition to the stochastic characteristics of foreground and background tasks, is an effective way to manage storage system idleness for overall high system performance.

Currently, we are working on refining the proposed approach and improving its adaptivity as system conditions change. Our focus is on self-adjusting various parameters in the proposed background scheduling policy. We are striving for a policy that is free of any input from offline analysis. Furthermore, we are extending the proposed policy to account for background work that is finite, that is, where background work is not always present. Examples of finite background tasks include verification of data written on disk and intradisk mirroring. Finite background jobs must be scheduled more aggressively, because foreground performance is affected less than with infinite background work. In such a case, background jobs should be scheduled as early as possible while the slowdown of foreground jobs is still kept up to the predefined target. Another aspect of the problem that we are exploring has to do with background work that does not always have strictly less priority than the foreground work. Examples of such background activities include disk cache flushing and RAID rebuild. Such activities can be deferred in background, but not indefinitely, that is, there is a deadline associated with their completion. We are currently working to refine the policies presented in this article to account for the aforementioned conditions.

REFERENCES

ABD-EL-MALEK, M., GANGER, G. R., GOODSON, G. R., REITER, M. K., AND WYLIE, J. J. 2005. Lazy verification in fault-tolerant distributed storage systems. In *Proceedings of the 24th IEEE Symposium on Reliable Distributed Systems (SRDS)*.

BACHMAT, E. AND SCHINDLER, J. 2002. Analysis of methods for scheduling low priority disk drive tasks. In *Proceedings of the ACM Conference on Measurements and Modeling of Computer Systems (SIGMETRICS)*. ACM Press. 55–65.

BAIRAVASUNDARAM, L. N., GOODSON, G. R., PASUPATHY, S., AND SCHINDLER, J. 2007. An analysis of latent sector errors in disk drives. In *Proceedings of the ACM SIGMETRICS Conference*. 289–300.

COLARELLI, D. AND GRUNWALD, D. 2002. Massive arrays of idle disks for storage archives. In *Proceeding of the SuperComputing Conferences*. 1–11.

DOUCEUR, J. R. AND BOLOSKY, W. J. 1999. Progress-Based regulation of low-importance processes. In *Proceedings of 17th ACM Symposium on Operating Systems Principles (SOSP'99)*. ACM Press. 247–260.

DOUGLIS, F., KRISHNAN, P., AND BERSHAD, B. N. 1995. Adaptive disk spin-down policies for mobile computers. In *Proceedings of the 2nd USENIX Symposium on Mobile and Location-Independent Computing*. 121–137.

EGGERT, L. AND TOUCH, J. D. 2005. Idletime scheduling with preemption intervals. In *Proceedings of the 20th ACM Symposium on Operating Systems Principles (SOSP'05)*. ACM Press. 249–262.

GOLDING, R., BOSCH, P., STAELIN, C., SULLIVAN, T., AND WILKES, J. 1995. Idleness is not sloth. In *Proceedings of the Winter'95 USENIX Conference.* 201–222.

HELMBOLD, D. P., LONG, D. D. E., SCONYERS, T. L., AND SHERROD, B. 2000. Adaptive disk spin-down for mobile computers. *Mobile Netw. Appl 5*, 4, 285–297.

HUANG, H., HUNG, W., AND SHIN, K. G. 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)*. ACM Press. 263–276.

ILIADIS, I., HAAS, R., HU, X.-Y., AND ELEFTHERIOU, E. 2008. Disk scrubbing versus intra-disk redundancy for high-reliability raid storage systems. In *Proceedings of the ACM SIGMETRICS Conference* 241–252.

LITZKOW, M. J., LIVNY, M., AND MUTKA, M. W. 1988. Condor - A hunter of idle workstations. In *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*. 104–111.

LO, V. M., ZAPPALA, D., ZHOU, D., LIU, Y., AND ZHAO, S. 2004. Cluster computing on the fly: P2P scheduling of idle cycles in the Internet. In *Proceedings of the International Workshop on Peer-to-Peer Systems (IPTPS)*. 227–236.

MERCHANT, A. AND YU, P. S. 1994. An analytic model of reconstruction time in mirrored disks. *Perform. Eval. J. 20,* 1-3, 115–129.

MI, N., RISKA, A., SMIRNI, E., AND RIEDEL, E. 2008. Enhancing data availability in disk drives through background activities. In *Proceedings of the Symposium on the Dependability of Systems and Networks (DSN)*. 492–501.

MUNTZ, R. R. AND LUI, J. C. S. 1990. Performance analysis of disk arrays under failures. In *International Conference on Very Large Databases (VLDB)*. 162–173.

NIU, Z., SHU, T., AND TAKAHASHI, Y. 2003. A vacation queue with setup and close-down times and batch markovian arrival processes. *Perform. Eval. 54,* 3, 225–248.

OSOGAMI, T., HARCHOL-BALTER, M., AND SCHELLER-WOLF, A. 2005. Analysis of cycle stealing with switching times and thresholds. *Perform. Eval. J. 61*, 4, 347–369.

RISKA, A. AND RIEDEL, E. 2006. Disk drive level workload characterization. In *Proceedings of the USENIX Annual Technical Conference*. 97–103.

RISKA, A. AND RIEDEL, E. 2008. Idle read after write - IRAW. In *Proceedings of the USENIX Annual Technical Conference*. 43–56.

SCHWARZ, T. J. E., XIN, Q., MILLER, E. L., LONG, D. D. E., HOSPODOR, A., AND NG, S. 2004. Disk scrubbing in large archival storage systems. In *Proceedings of the International Symposium on Modeling and Simulation of Computer and Communications Systems (MASCOTS)*. IEEE Press.

SIVATHANU, M., PRABHAKARAN, V., ARPACI-DUSSEAU, A. C., AND ARPACI-DUSSEAU, R. H. 2004. Improving storage system availability with D-GRAID. In *Proceedings of the 3rd USENIX Symposium on File and Storage Technologies (FAST'04)*.

TAKAGI, H. 1991. *Queuing Analysis Volume 1: Vacations and Priority Systems*. North-Holland, New York.

THEIMER, M. M., LANTZ, K. A., AND CHERITON, D. R. 1985. Preemptable remote execution facilities for the v-system. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*. 2–12.

THERESKA, E., SCHINDLER, J., BUCY, J., SALMON, B., LUMB, C. R., AND GANGER, G. R. 2004. A framework for building unobtrusive disk maintenance applications. In *Proceedings of the 3rd USENIX Conference on File and Storage Technologies (FAST)*.

THOMASIAN, A. AND NICOLA, V. F. 1993. Performance evaluation of a threshold policy for scheduling readers and writers. *IEEE Trans. Comput. 42*, 1, 83–98.

VENKATARAMANI, A., KOKKU, R., AND DAHLIN, M. 2002. TCP nice: A mechanism for background transfers. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI)*. 329–343.

XU, E. AND ALFA, A. S. 2002. A vacation model for the non-saturated readers and writers system with a threshold policy. *Perform. Eval. 50,* 4, 233–244.

Received July 2008; revised August 2008; accepted January 2009